Designing losses for data-free training of normalizing flows on Boltzmann distributions

Loris Felardos, Jérôme Hénin, Guillaume Charpiat

Designing losses for data-free training of normalizing flows on Boltzmann distributions

Loris Felardos[1,2]
loris.felardos.212@use.startmail.com

Jérôme Hénin[2]
jerome.henin@cnrs.fr

Guillaume Charpiat[1]
guillaume.charpiat@inria.fr

[1] Université Paris-Saclay, CNRS, Inria, Laboratoire interdisciplinaire des sciences du numérique, Orsay, France
[2] Université Paris Cité, CNRS, Laboratoire de Biochimie Théorique UPR 9080, Paris, France

Abstract

Generating a Boltzmann distribution in high dimension has recently been achieved with Normalizing Flows, which enable fast and exact computation of the generated density, and thus unbiased estimation of expectations. However, current implementations rely on accurate training data, which typically comes from computationally expensive simulations. There is therefore a clear incentive to train models with incomplete or no data by relying solely on the target density, which can be obtained from a physical energy model (up to a constant factor). For that purpose, we analyze the properties of standard losses based on Kullback-Leibler divergences. We showcase their limitations, in particular a strong propensity for mode collapse during optimization on high-dimensional distributions. We then propose strategies to alleviate these issues, most importantly a new loss function well-grounded in theory and with suitable optimization properties. Using as a benchmark the generation of 3D molecular configurations, we show on several tasks that, for the first time, imperfect pre-trained models can be further optimized in the absence of training data.

1 Introduction

Application context. In statistical physics, the properties of materials and molecular systems are expressed as expectations over probability distributions of microscopic configurations, which are determined by macroscopic, thermodynamic parameters. Such expectations can be estimated numerically by Monte Carlo averaging using samples from physically relevant distributions, particularly the Boltzmann distribution characterizing systems at equilibrium with a thermostat. The Boltzmann distribution over configurations x is characterized by the density p_B, which is related to the potential energy U_B by:

$$p_B(x) = \frac{1}{\mathcal{Z}_B} \cdot e^{-\beta U_B(x)} \tag{1}$$

where $\beta = 1/k_B T$ is the inverse temperature, and $\mathcal{Z}_B$ is a normalization factor known as the partition function. Though there are usually closed form expressions or robust numerical methods to estimate $\tilde{p}_B := \mathcal{Z}_B\, p_B$, there is no direct method to sample it. In practice, sampling is commonly performed with stochastic simulations of physical systems, however p_B is typically high-dimensional and

Preprint. Under review.

multimodal, so that simulations are plagued by long autocorrelation times. Sampling with generative models, which produce i.i.d. samples, is a potential avenue to overcome these limitations.

Normalizing Flows for Boltzmann distributions. Flow-based models (often just called normalizing flows) are a valuable type of architecture for this purpose ([1], [2], [3] and [4] for an overview), which is invertible and yields not only samples x but also the probability density $p_G(x)$ of the generated distribution. This in turns allows for unbiased estimation of expectations with respect to the ground-truth Boltzmann distribution via reweighting:

$$\mathop{\mathbb{E}}_{x\sim p_B}[f(x)] = \mathop{\mathbb{E}}_{x\sim p_G}\left[\frac{p_B(x)}{p_G(x)}\, f(x)\right] \tag{2}$$

for any function f, assuming that p_G is nonzero over the support of f. This is the case of Boltzmann generators [5], which are based on Normalizing Flows with affine coupling layers [6], trained to generate a known Boltzmann distribution.

Designing more robust and expressive normalizing flow architectures is an active field of research, with innovations such as rank-one perturbations to train fully connected layers [7], Augmented Normalizing Flows [8], Stochastic Normalizing Flows [9], Smooth Normalizing Flows [10] and base distribution resampling [11].

Towards data-free training. In principle, a loss function based on a well-chosen KL divergence should allow for the training of normalizing flows in the absence of data, merely based on the knowledge of the target Boltzmann distribution up to a constant factor [4]. However, there are no claims of successful numerical experiments in the literature, suggesting that this approach may be impractical for so-far undocumented reasons. Thus, it remains that in practice, Boltzmann generators must be trained using accurate reference data, which makes them applicable to systems that have already been sampled by other means, rather than standalone substitutes to simulations for studying new, unknown systems. Generally speaking, training generative models on high-dimensional distributions is difficult because it puts a high demand on the space to be covered during training; training them in the absence of complete reference data is to date an unsolved problem. There are two requirements for success: proper convergence (which implies stability of the generated distribution near its target), and exploration of the ground-truth distribution. Here we focus on stability and propose the very first data-free loss leading to stable training. We discuss possible approaches for an exploration strategy in the discussion (section 6).

Contributions and overview. In this work, we analyze the properties of loss functions based on Kullback-Leibler divergences, and showcase their limitations, in particular their lack of robustness with respect to discretization, with a general tendency towards mode drop that makes data-free training unstable. We then introduce a loss function that exhibits stable refinement training in the absence of data, after an initial data-dependent pre-training. We assess all losses and training strategies on a toy model (a high-dimensional double-well potential) and two molecular systems. We further discuss the sensitivity of normalizing flow training to degrees of freedom with broad probability distributions in the output, and propose strategies to avoid these effects at the level of the training criterion, without added architectural constraints such as equivariance or invariance.

2 Optimizing Kullback-Leibler Divergences

KL divergence defined in z-space. The goal of training a Normalizing Flow $G = F^{-1}$ is to obtain a one-to-one mapping between a known base distribution $q_\mathcal{N}$ (typically Gaussian) and a target distribution p_B, such that the pushforward measure p_G of $q_\mathcal{N}$ by G is similar to p_B.

$$\overbrace{z_\mathcal{N} \sim q_\mathcal{N}}^{\text{Gaussian distribution}} \xrightarrow{\boldsymbol{G}} \overbrace{x_G = G(z_\mathcal{N}) \sim p_G}^{\text{generated distribution}}$$
$$z_F = F(x_B) \sim q_F \xleftarrow{\boldsymbol{F}} \underbrace{x_B \sim p_B}_{\text{target distribution}}$$

Since normalizing flows are bijective, the conventional way of achieving this is by providing x_B samples from p_B (usually from a dataset) to the inverse function F and then minimizing the KL

divergence between the pushforward measure q_F of p_B by F and the known $q_\mathcal{N}$.

$$KL(q_F||q_\mathcal{N}) = \int q_F(z) \log \frac{q_F(z)}{q_\mathcal{N}(z)} \, dz \tag{3a}$$

$$= \log \mathcal{Z}_\mathcal{N} - S_B + \mathop{\mathbb{E}}_{x_B \sim p_B} \left[\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_B)) - \log \left| \det \left(\frac{\partial F(x_B)}{\partial x_B} \right) \right| \right] \tag{3b}$$

When leveraging the principle of Stochastic Gradient Descent, this gives rise to the following standard and data-*dependent* loss function, with $\mathbf{x}_B$ a mini-batch of x_B points sampled from p_B (appendix A):

$$\mathcal{L}_{KLz}(\mathbf{x}_B) = \sum_{i=1}^{n} \left[\frac{1}{n} \cdot \left[\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_{B,i})) - \log \left| \det \left(\frac{\partial F(x_{B,i})}{\partial x_{B,i}} \right) \right| \right] \right] \tag{4}$$

KL divergence defined in x-space. Another loss can be derived in an almost identical fashion by defining a KL divergence in x-space instead (appendix B):

$$KL(p_G||p_B) = \int p_G(x) \log \frac{p_G(x)}{p_B(x)} \, dx \tag{5a}$$

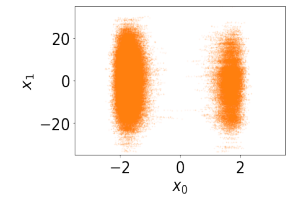
$$= \log \mathcal{Z}_B - S_\mathcal{N} + \mathop{\mathbb{E}}_{z_\mathcal{N} \sim q_\mathcal{N}} \left[\beta U_B(G(z_\mathcal{N})) - \log \left| \det \left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}} \right) \right| \right] \tag{5b}$$

This leads to the following data-*free* loss function (with $\mathbf{z}_\mathcal{N}$ a mini-batch of $z_\mathcal{N}$ points)::

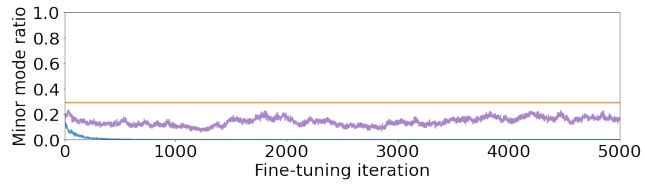
$$\mathcal{L}_{KLx}(\mathbf{z}_\mathcal{N}) = \sum_{i=1}^{n} \left[\frac{1}{n} \cdot \left[\beta U_B(G(z_{\mathcal{N},i})) - \log \left| \det \left(\frac{\partial G(z_{\mathcal{N},i})}{\partial z_{\mathcal{N},i}} \right) \right| \right] \right] \tag{6}$$

Comparing $\mathcal{L}_{KLx}$ with $\mathcal{L}_{KLz}$. When optimizing over mini-batches, these two loss functions behave very differently. $\mathcal{L}_{KLz}$ is known to be very stable and leads to good performance [4] while $\mathcal{L}_{KLx}$ is more erratic and often leads to mode collapse. To illustrate this, we pre-train a model on a simple dataset with $\mathcal{L}_{KLz}$ and then fine-tune it with $\mathcal{L}_{KLx}$. This is the general experimental setup used in this work. Poor pre-trainings are allowed as long as they do not miss an entire mode of the target distribution so as to analyze whether the fine-tunings manage to refine p_G successfully. See section 6 on possible strategies to remove this data-dependent pre-training in the future.

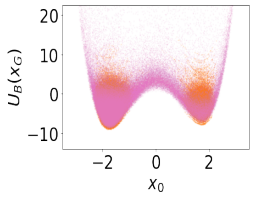
The dataset is a simple double well in 12 dimensions similar to those used in previous works[5, 12] (figure 1a), where the first dimension is bimodal and the 11 other dimensions are independent and



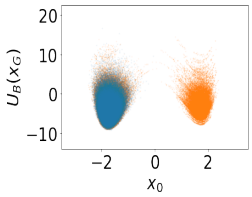
(a) 2D Projection of the dataset: `Double Well 12D`



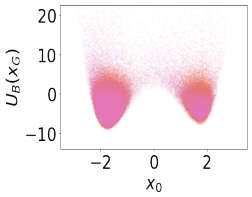
(b) Percentage of generated samples $x_G \sim p_G$ in the minor mode during fine-tunings compared to the real ratio from p_B (in orange).



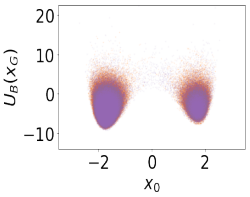
(c) Partial pre-training.



(d) Fine-tuning of 1c.



(e) Complete pre-training.



(f) Fine-tuning of 1e.

Figure 1: Results of two fine-tunings with $\mathcal{L}_{KLx}$ after pre-trainings of different lengths with $\mathcal{L}_{KLz}$. Data from p_B (i.e. the dataset) is represented in orange. Both pre-training results are represented in pink. Fine-tuning results after the partial pre-training are represented in blue (note the total collapse to the major mode). Fine-tuning results after the complete pre-training are represented in purple. Figures 1c to 1f all represent the potential energy U_B of generated samples x_G (in ordinates) as a function of the multi-modal dimension (in abscissa).

Gaussian with a standard deviation of 10. The lack of normalization is intended to exhibit how difficult this task already is for $\mathcal{L}_{KLx}$. Even with a partial pre-training that already samples the bottom of each mode (figure 1c), it is incapable of keeping both modes and collapses to the main one (figure 1d). A complete pre-training with $\mathcal{L}_{KLz}$ (figure 1e) results in a better fine-tuning (figure 1f) but is still not sufficient to completely stabilize $\mathcal{L}_{KLx}$ as shown in figure 1b when looking at the ratio between the modes over time. Note that this failure is *not* due to the poor normalization of the target distribution, which only exacerbates this undesirable behavior, since $\mathcal{L}_{KLx}$ also fails on more complex datasets with good normalization (appendix C).

Making $\mathcal{L}_{KLz}$ data-free. The standard loss $\mathcal{L}_{KLz}$ cannot be used in a data-free setting since it relies on samples from p_B, but it can be modified to use samples from p_G instead by leveraging importance sampling (appendix D):

$$\mathcal{L}^{\text{df}}_{KLz}(\mathbf{x}^{\ddagger}_G) = \sum_{i=1}^{n} \frac{1}{n} \cdot \left[\left(\frac{\tilde{p}_B(x^{\ddagger}_G)}{p_G(x^{\ddagger}_G)} \right)^{\ddagger} \cdot \left(\frac{1}{2\sigma^2} U_{\mathcal{N}}(F(x^{\ddagger}_{G,i})) - \log \left| \det \left(\frac{\partial F(x^{\ddagger}_{G,i})}{\partial x^{\ddagger}_{G,i}} \right) \right| \right) \right] \tag{7}$$

with $\ddagger$ the symbol used to denote the "detach" operator that makes the term constant with respect to gradient descent: $\mathbf{x}^{\ddagger}_G \sim p^{\ddagger}_G$ is therefore a detached mini-batch of size n.

This results in a loss that is significantly more stable than $\mathcal{L}_{KLx}$ and works perfectly on `Double well 12D` (data not shown). It also achieves good performance on more complex target distributions like that of `Butane` (figure 2). The configurations of the butane molecule have three main modes that can easily be visualized when projecting onto the values of the dihedral angle ϕ of its carbon chain (in red, figure 2a).

The potential energy function U_B of physical systems in the absence of external fields is invariant by collective rotation and translation. When the generative model is expressed in Cartesian coordinates and is not equivariant with respect to these external degrees of freedom, it is necessary to add a loss term that discourages translations and rotations, essentially acting as an alignment penalty. This penalty is weighted by a scalar denoted λ_{align}. To showcase the different behaviors of $\mathcal{L}_{KLx}$ and $\mathcal{L}^{\text{df}}_{KLz}$, a model is pre-trained with $\lambda_{\text{align}} = 10$ and then fine-tuned with $\lambda_{\text{align}} = 0$, essentially asking the generated density to expand infinitely in the translational degrees of freedom and to cover all possible rotations.

$\mathcal{L}_{KLx}$ makes p_G continuously expand translationally (figure 2c) but at the expense of losing the minor modes (figures 2b and 2d), resulting in an explosion of the energy of generation U_G (figure 2e). $\mathcal{L}^{\text{df}}_{KLz}$ on the other hand, does *not* explore significantly (figure 2g) but remains very stable by keeping all the modes (figure 2i) and producing samples that stay at low energy levels (figures 2h and 2j).

While these results describe the extreme case of degrees of freedom distributed uniformly over $\mathbb{R}$, they exemplify the importance of removing unnecessary degrees of freedom for better performance, especially those whose broad distribution considerably expands the support of the target density. For translations and rotations, this can be achieved by always using $\lambda_{\text{align}} > 0$. Of note, the degrees of freedom of hydrogen atoms (which are permutation invariant within groups like $-CH3$ for example) are also ignored here. The model is only asked to generate the positions of the carbon atoms, and another module places the hydrogen atoms deterministically near their energy minimum. This introduces a bias and changes the target ratio between the modes (from the solid to the dashed line in figure 2b, see appendix E) but does not explain why $\mathcal{L}^{\text{df}}_{KLz}$ does not converge to the expected ("dashed") ratio. $\mathcal{L}^{\text{df}}_{KLz}$ is also shown to be unstable on more complex datasets (i.e. `Dialanine`, figure 3i) and a better loss is developed in section 4 to counteract this problem.

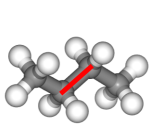
Since divergences are not symmetric, one might also wonder what happens when swapping the two distributions within the KL divergences, but an important result from the literature [4] already shows that the minimizations of $KL(q_{\mathcal{N}}||q_F)$ and $KL(p_G||p_B)$ are equivalent, as well as the minimizations of $KL(p_B||p_G)$ and $KL(q_F||q_{\mathcal{N}})$. $KL(p_B||p_G)$ is known to often lead to mode-drop in x-space [13], whereas $KL(q_{\mathcal{N}}||q_F)$ tends to avoid that behavior (in our case, it may cause mode collapse in z-space but this is not an impediment since the Gaussian target distribution has only one mode). Note also that combining both data-free losses ($\mathcal{L}^{\text{df}}_{KLz}$ and $\mathcal{L}_{KLx}$) is not sufficient to get proper ratios since $\mathcal{L}_{KLx}$ tend to dominate and the fine-tuning still results in a mode-collapse.

Optimization pitfalls due to discretization over minibatches. In Appendix I, we show that in general the optimization of Kullback-Leibler divergences with respect to a distribution suffers from severe issues when discretized over minibatches without proper normalization. This is due to the fact that the properties of KL heavily rely on a global unit mass constraint (for Gibbs inequality to hold), which hinders its estimation or optimization in practice. We show how to build more suitable estimators of the gradient of the Kullback-Leibler divergence, as well as how to minimize their variance via a *stabilizing trick*.

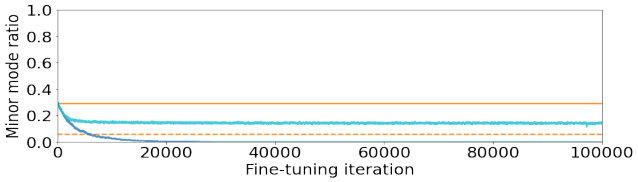
3 Desirable Properties for a Loss Function

3.1 Estimator variance as a loss

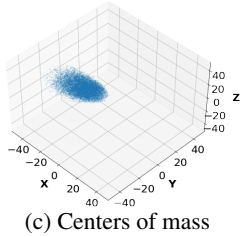
With normalizing flows, one can compute exactly the probability with which one generates any given point. As a consequence, one can correct the sampler based on the trained generator with importance sampling, i.e. by associating each sample x with a weight $\frac{p_B(x)}{p_G(x)}$. Expectations are then taken with respect to $\frac{p_B}{p_G}p_G$, which *exactly* matches the target p_B, regardless of p_G (provided that it has positive density everywhere p_B does). However, if importance sampling weights are closer to 1, the produced distribution will converge faster towards p_B, that is, fewer samples will need to be generated. The question here is how to design a loss to train p_G in such a context where the reweighted output distribution is always perfect.



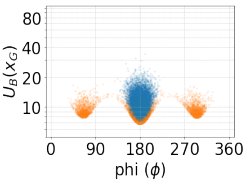
(a) A configuration of `Butane`



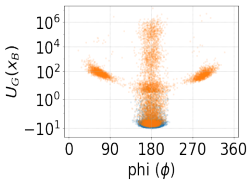
(b) Percentage of generated samples $x_G \sim p_G$ in the minor modes.



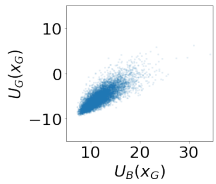
(c) Centers of mass



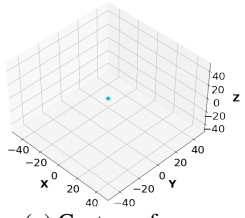
(d) U_B energies



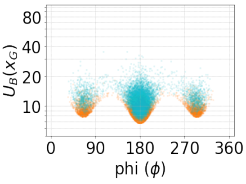
(e) U_G energies



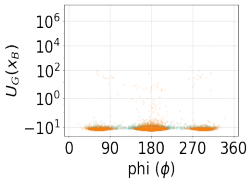
(f) Correlations



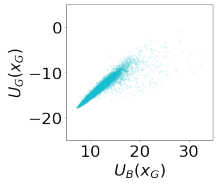
(g) Centers of mass



(h) U_B energies



(i) U_G energies



(j) Correlations

Figure 2: Results of two fine-tunings with $\mathcal{L}_{KLx}$ (second row) and $\mathcal{L}^{\text{df}}_{KLz}$ (third row) after the same pre-training with $\mathcal{L}_{KLz}$ on `Butane`. In all sub-figures, data from p_B (i.e. the dataset) is represented in orange, fine-tuning results with $\mathcal{L}_{KLx}$ are represented in blue, fine-tuning results with $\mathcal{L}^{\text{df}}_{KLz}$ are represented in cyan.
- Figures 2c and 2g represent the centers of mass of generated samples $x_G \sim p_G$.
- Figures 2d and 2h represent the potential energy U_B of generated samples $x_G \sim p_G$ (in blue or cyan) vs. samples from the dataset $x_B \sim p_B$ (in orange). Note that in both cases the energy of the hydrogens is minimized (either by the model or manually). These figures visualize whether or not $p_G \subset p_B$.
- Figures 2e and 2i represent the energy of generation U_G of samples from the dataset $x_B \sim p_B$ according to each pre-trained model (in orange) vs generated samples $x_G \sim p_G$ (in blue or cyan). These figures visualize whether or not $p_B \subset p_G$.
- Figures 2f and 2j represent the correlations between the energy of generation U_G and the potential energy U_B of generated samples $x_G \sim p_G$.

An important application of our generator G is often to estimate integral quantities of the form $\mathbb{E}_{p_B}[f]$ for some given function f. For instance, a classic use case in practice is to compute the free energy difference ΔF_{BC} between the state being sampled (with energy U_B) and an alternate state (with energy U_C). Then:

$$e^{-\beta \Delta F_{BC}} := \mathop{\mathbb{E}}_{x \sim p_B}[f] \qquad \text{with} \qquad f(x) = e^{-\beta(U_C(x) - U_B(x))} \tag{8}$$

Let us denote by Q the true value of the quantity to estimate:

$$Q := \mathop{\mathbb{E}}_{p_B}[f] := \int_{x \in \mathcal{X}} p_B(x) f(x) dx = \mathop{\mathbb{E}}_{p_G}\left[\frac{p_B}{p_G} \cdot f\right] \tag{9}$$

The latter equality holds under the assumption that p_G is never 0 where p_B is not. For any p_G, the following quantity $\widehat{Q}$ is an unbiased estimator of Q:

$$\widehat{Q} := \frac{1}{n} \sum_{x_i \in m} f(x_i) \frac{p_B(x_i)}{p_G(x_i)} \tag{10}$$

where $m = (x_1, \dots, x_N)$ is a large set of points sampled according to p_G. That is, when averaging over all possible mini-batches, $\widehat{Q}$ becomes Q (i.e. $\mathbb{E}_m[\widehat{Q}] = Q$). Yet, for some distributions p_G, the estimate $\widehat{Q}$ may converge faster than others, in terms of number of samples required to reach a given accuracy. The quality of a generator p_G can thus be quantified through the expected error when estimating Q with n points. This can be shown to be proportional to the variance of $\widehat{Q}$, which can then be turned into a training loss (see appendix F for a proof):

$$\mathcal{L}_f(p_G) = \mathop{\mathbb{E}}_{x \sim p_G}\left[\frac{p_B^2(x)}{p_G^2(x)} \cdot f^2(x)\right] \tag{11}$$

If the function f is not fixed and can be any bounded function over the space $\mathcal{X}$ of points x, then one can deduce the following optimization criterion:

$$\mathcal{L}(p_G) = \mathop{\mathbb{E}}_{x \sim p_G}\left[\frac{p_B^2(x)}{p_G^2(x)}\right] = \mathop{\mathbb{E}}_{x \sim p_B}\left[\frac{p_B(x)}{p_G(x)}\right] = e^{RN_2(p_B||p_G)} = \mathop{\text{var}}_{x \sim p_G}\left[\frac{p_B}{p_G}\right] + 1 \tag{12}$$

where $RN_2(p_B||p_G)$ is the Rényi divergence of order 2. This formula looks very similar to the KL divergence, without the $\log$, thus penalizing high ratios $\frac{p_B}{p_G}$ more strongly. In practice, one knows how to compute $\tilde{p}_B(x) := \mathcal{Z}_B p_B(x)$ but not $p_B(x)$ directly. Fortunately, a model p_G trained with $\mathcal{L}$ will yield by definition a good estimator of $\mathcal{Z}_B = \mathbb{E}_{p_B}[\mathcal{Z}_B] = \mathbb{E}_{p_G}\left[\frac{\tilde{p}_B}{p_G}\right]$.

Another justification for this loss is that one aims to find $p_G \propto \tilde{p}_B$, and therefore to make the ratio $\frac{\tilde{p}_B}{p_G}$ constant over $\mathcal{X}$. Without knowing the value of the target constant, this can still be achieved by minimizing the variance of the ratio over $\mathcal{X}$, which is precisely the loss $\mathcal{L}$.

Thus we arrive at $\mathcal{L}(p_G) = \text{var}_{x \sim p_G}\left[\frac{p_B}{p_G}\right]$ as a principled loss to minimize the variance of estimators of expectations over the Boltzmann distribution.

3.2 Practical Recommendations

Beyond the theoretical points considered in section 3.1, there are a few practical considerations that need to be addressed.

Degrees of freedom:

- As illustrated in section 2, avoiding unnecessary symmetries within the target distribution is often beneficial to ease the training. Hydrogen atoms for instance are permutation invariant within $-CH3$ groups and thus multiply by 6 the total number of modes for each group. Since the position of hydrogen atoms is often irrelevant for downstream applications they can often be ignored. In this work we choose the simplest method which consist in placing the hydrogens

deterministically near their energy minimum at the cost of intruducing a bias that changes the ratio between modes. Better options exist such as adjusting U_B (to encourage having only one permutation possible), or placing hydrogen atoms stochastically but with a model that does not care about mode collapse.
- More importantly, extremely flat degrees of freedom should be removed if possible. When it comes to translations and rotations, several approaches are available. One could add an alignment penalty to the potential energy U_B (as described in section 2), but it is also possible to generate configurations in internal coordinates directly (thereby removing 6 degrees of freedom).

Numerical instabilities:

- The loss $\mathcal{L}^{\mathrm{df}}_{KLz}$ may suffer from training instabilities due to the use of importance sampling weights that have a high variance and therefore often focuses most of the gradient onto just a few points of each mini-batch. Such weights should be avoided if possible during the design of new loss functions.
- The potential energy term U_B is also at risk of introducing training instabilities since it can be very sensitive to small changes in the position of the atoms. The strategy followed in this work is to cap each term of the energy function individually, so that their gradient never exceeds a given threshold. This approach is much more fine-grained than using a global capping, directly on U_B.

Minimizing vs. maximizing the energy terms:

- The term U_B should probably never be increased explicitly through gradient descent (which is equivalent to saying that p_B should never be decreased). Although some training objectives that do this may *seem* to be principled in the context of an integral over the whole space, they usually fail once converted into loss functions used on discrete mini-batches.
- In the same spirit, it is often a preferable to avoid decreasing p_G directly. Indeed, decreasing p_G at a given point implies moving the mass somewhere else, but since the direction where to move this mass is not specified, it could go anywhere without actually getting any closer to p_B. Since p_G is a probability distribution, increasing it anywhere implies that some other region of the space will become less probable to compensate (i.e. p_G cannot increase everywhere). In the case where p_G is never decreased explicitly (maybe by masking the troublesome points) the training is much smoother since the probability mass is always pushed where it is most needed.

4 A data-free L^2 loss

Building on $\mathrm{var}_{x\sim p_G}\left[\frac{p_B}{p_G}\right]$ (from equation 12), we replace ratios $\frac{p_B(x)}{p_G(x)}$ with log-ratios

$$r(x) = \log \frac{p_B(x)}{p_G(x)} \tag{13}$$

for numerical reasons, as normalizing flows actually compute log-probabilities and the exponentiation leads to instability. We also note that:

$$\underset{p_G}{\mathrm{var}}[r] = \underset{p_G}{\mathbb{E}}\left[\left(r - \underset{p_G}{\mathbb{E}}[r]\right)^2\right] \tag{14}$$

This formulation with differences between log-ratios has the advantage of making $\mathcal{Z}_B$ cancel out from the computations in practice. To avoid decreasing $p_G(x)$ explicitly at any point x, as mentioned in Section 3.2, we modify the loss as follows by masking $(r-K)$. As a consequence, r (and therefore U_G) can only be minimized (whereas $U_B(x^\ddagger_G)$ is not differentiated with respect to θ). The masked L^2 loss with detached means is therefore defined as:

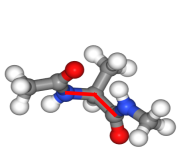
$$\mathcal{L}_{L^2_+}(\mathbf{x}^\ddagger_G) = \sum_{i=1}^{n}\left[\frac{1}{n}\cdot\left[\left(r(x^\ddagger_{G,i}) - K^\ddagger\right)^2_+\right]\right] \tag{15}$$

where $a^2_+ = a^2$ if $a > 0$ and 0 otherwise, and where $K^\ddagger = \left[\sum_{j=1}^{n}\left[\frac{1}{n}\cdot r(x^\ddagger_{G,j})\right]\right]^\ddagger$ is not differentiated (so as to ensure that it is never increased). Note that in the continuous limit: $\mathbb{E}_{p_G}[r] = -KL(p_G||p_B) \leqslant 0$.

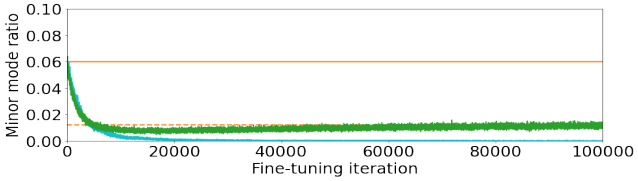
One can prove that, in spite of the non-differentiation of $K^{\ddagger}$, a pseudo-gradient descent on this loss *will* converge towards p_B, provided the model is expressive enough and that the initial p_G is non-zero on the support of p_B, for an adequate choice of inner product (appendix G).

This loss has common features with log-variance loss of Richter et al. [14], yet the mask applied in the present loss is critical for stability, just as well as detaching K (see the ablation study in appendix H).

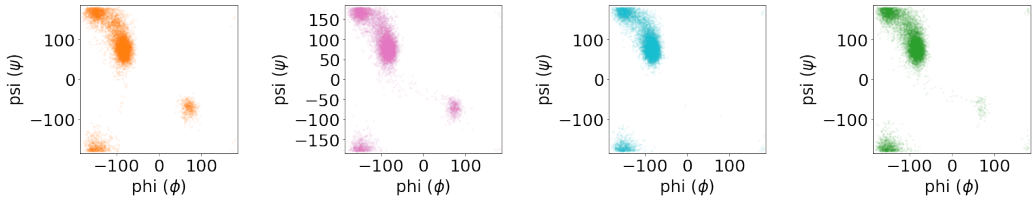
The conformational distribution of dialanine is often projected onto its two main dihedral angles ϕ and ψ for visualization (figures 3a and 3c). The "real" distribution p_B has about $\approx 6\%$ of its mass in the minor mode (the one where $\phi > 0$) but when taking into account the minimization of the energy of hydrogen atoms, this ratio drops to $\approx 1.21\%$ (dashed line in figure 3b, see appendix E).



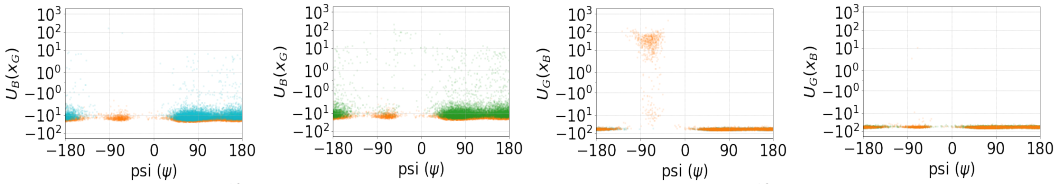
(a) A configuration of dialanine



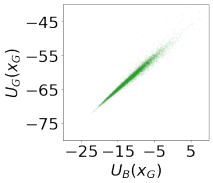
(b) Percentage of generated samples $x_G \sim p_G$ in the minor modes.



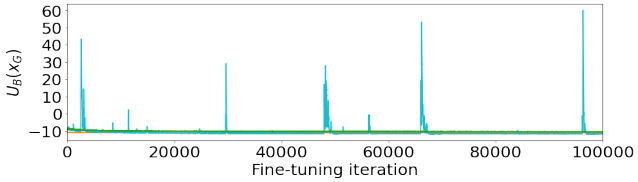
(c) Projections on (ϕ, ψ) dihedral angles. Ground truth target (orange), model after data-dependent pre-training (pink), models fine-tuned with $\mathcal{L}^{\text{df}}_{KLz}$ (cyan) and $\mathcal{L}_{L^2_+}$ (green)



(d) U_B energies with $\mathcal{L}^{\text{df}}_{KLz}$ (e) U_B energies with $\mathcal{L}_{L^2_+}$ (f) U_G energies with $\mathcal{L}^{\text{df}}_{KLz}$ (g) U_G energies with $\mathcal{L}_{L^2_+}$



(h) Correlations



(i) U_B energy of generated samples $x_G \sim p_G$ during fine-tuning.

Figure 3: Results of two fine-tunings (with $\mathcal{L}^{\text{df}}_{KLz}$ and $\mathcal{L}_{L^2_+}$) after the same pre-training with $\mathcal{L}_{KLz}$ on `Dialanine`. In all sub-figures, data from p_B (i.e. the dataset) is colored in orange, pre-training results are colored in pink, fine-tuning results with $\mathcal{L}^{\text{df}}_{KLz}$ are colored in cyan, fine-tuning results with $\mathcal{L}_{L^2_+}$ are colored in green.

- Figure 3b represents the percentage of generated samples $x_G \sim p_G$ in the minor modes during fine-tuning. The solid orange line corresponds to the "real" ratio from p_B, whereas the dashed orange line corresponds to the same ratio from p_B but with the energy minimized with repect to hydrogen atom coordinates (appendix E).
- Figure 3c contains 2D projections of the ground truth dataset and generated data.
- Figures 3d and 3e represent the potential energy U_B of generated samples $x_G \sim p_G$ (in cyan or green) vs. samples from the dataset $x_B \sim p_B$ (in orange). Note that in both cases the energy of the hydrogens is minimized (either by the model or manually). These figures visualize whether or not $p_G \subset p_B$.
- Figures 3f and 3g represent the energy of generation U_G of samples from the dataset $x_B \sim p_B$ (in orange) vs generated samples $x_G \sim p_G$ (in cyan or green). These figures visualize whether or not $p_B \subset p_G$.
- Figures 3h represents the correlations between the energy of generation U_G and the potential energy U_B of generated samples $x_G \sim p_G$.
- Figure 3i represents the potential energy U_B of generated samples $x_G \sim p_G$ during fine-tuning. Note the instability of $\mathcal{L}^{\text{df}}_{KLz}$ compared to the stability of $\mathcal{L}_{L^2_+}$.

This means that the result of the pre-training with the data-dependent $\mathcal{L}_{KLz}$ produces a ratio ($\approx 6\%$, figure 3c) that is different from the one expected at the end of the fine-tuning ($\approx 1.21\%$). It is clear that $\mathcal{L}^{\mathrm{df}}_{KLz}$ completely loses the minor mode (figures 3c and 3f) whereas $\mathcal{L}_{L^2_+}$ does not (figures 3c and 3g) and converges to the expected correct ratio of $\approx 1.21\%$. The bias induced by the deterministic placement of the hydrogen atoms does not change the main point that $\mathcal{L}_{L^2_+}$ converged to the ratio it was supposed to produce. Another thing to notice is that $\mathcal{L}^{\mathrm{df}}_{KLz}$ has unstable U_B energies during fine-tuning whereas the $\mathcal{L}_{L^2_+}$ does not (figure 3i). In addition to those clear qualitative improvements, and unlike $\mathcal{L}^{\mathrm{df}}_{KLz}$, $\mathcal{L}_{L^2_+}$ does not rely on numerically unstable importance sampling weights.

Note that the accuracy on this test is limited by the choice of generating deterministic hydrogen atom positions: this can be lifted by using a conditional normalizing flow to generate a Boltzmann distribution of hydrogen atom positions conditioned on the set of heavy atom positions generated by the main model.

5 Technical details

Data and pretraining. For all datasets (i.e. `Double Well 12D`, `Butane` and `Dialanine`), the data has been generated by Metropolis-Hastings simulations with Parallel Tempering [15–17]). The potential energy U_B of the molecules of butane and dialanine is evaluated according to the CHARMM36m force field [18]. The energy function used in the simulations also uses an alignment penalty with $\lambda_{\mathrm{align}} = 10$. The data-dependent pre-training is performed with L_{KLz}. The number of iterations was a 10th of the one used for fine-tunings (i.e. 10000 iterations).

Alignment Penalty. The alignment penalty is an L2 distance between generated coordinates and their image after a roto-translational alignment to some reference. The alignment may only be partial since we cap the maximum allowed rotation by $\pi/3$ to ease the training.

Model Architecture. Only two model architectures are used. One for `Double Well 12D` and one for molecular datasets (i.e. `Butane` and `Dialanine`). The architecture used in `Double Well 12D` experiments is a simple stack of 8×4 Coupling Blocks (as described in section [6]). Every Coupling Block uses an internal feed-forward sub-network M composed of two layers with an internal feature size of 64 separated by a CELU non-linearity [19]. The architecture used in `Butane` and `Dialanine` experiments is quite similar except for two changes:

- The stack is made deeper (24×4 Coupling Blocks) and wider (internal sub-networks M have a layer size of 256),
- and an additional feed-forward network is used to generate the position of the hydrogen atoms. It has 3 layers separated by CELUs and a hidden size of 512.

Error estimation. No error bars are provided, but empirically, each experiment proved to be entirely reproducible over dozens of runs.

Resources. Every experiment has been performed on a single machine with two GPUs GeForce RTX™ 2070. Each experiment on `Double Well 12D` takes about 40m, whereas the experiments on `Butane` and `Dialanine` take between 7 to 10 hours.

6 Conclusion and Perspectives

In this contribution, we have explored the conditions necessary for training or refining flow-based models based on an explicitly known target density, rather than pre-determined samples from a Markov-chain simulation. We have found that several losses that may seem appropriate in theory lead to numerical failures in a discrete setting. In particular, we have documented a major instability issue when optimizing the KL divergence $KL(p_G||p_B)$ between the generated and target distributions. We note that loss functions whose minimization amounts to decreasing the probability of a sample point (lowering either p_G or p_B) push the model to spread local mass in improbable directions, resulting in instability. Based on an estimator variance minimization approach, we have derived a stable data-free loss based on L^2 distances between log-distributions, with the important condition that a mask must

be applied to follow the criterion stated above. This loss is the first one to exhibit stable data-free optimization on the dialanine molecule task.

While this allows for stable optimization of a correctly trained model, lifting the requirement for complete reference data will require a training protocol able to explore the target space to discover new modes. We envision two families of approaches to that effect:

- Keeping the current paradigm of a generator fully trained on a single system, training could be initiated based on a limited and/or biased set of data, for example from high-temperature simulations, then extended using the properties of normalizing flows themselves [20, 21], enhanced-sampling simulations[22], or hybrid approaches [23].
- Alternately, the cost of complete training for every new target could be reduced by transferring information between systems using curriculum learning. In the case of molecular targets, this would require a generalizing model, e.g. one based on graph convolutions [24–26].

The novel masked L^2_+ loss has demonstrated remarkable stability on the `Dialanine` test case, which is a good benchmark for small molecules of pharmacological interest, and a smaller proof of concept for proteins. It remains to be seen how it will scale to larger systems, yet previous work has shown the normalizing flow approach to scale to larger molecules in presence of a training dataset [5].

Acknowledgments and Disclosure of Funding

Funding to LF was provided by Inria through IPL HPC-BigData. We are grateful to Bruno Raffin for leading the consortium that created and supported this project. We also thank Victor Berger and Cyril Furtlehner for fruitful discussions.

References

[1] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217 – 233, 2010. doi: cms/1266935020.

[2] E. Tabak and Turner Cristina. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66, 02 2013. doi: 10.1002/cpa.21423.

[3] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2015. URL `https://arxiv.org/abs/1505.05770`.

[4] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2019. URL `https://arxiv.org/abs/1912.02762`.

[5] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), September 2019. doi: 10.1126/science.aaw1147. URL `https://doi.org/10.1126/science.aaw1147`.

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2016. URL `https://arxiv.org/abs/1605.08803`.

[7] Andreas Krämer, Jonas Köhler, and Frank Noé. Training invertible linear layers through rank-one perturbations, 2020. URL `https://arxiv.org/abs/2010.07033`.

[8] Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models, 2020. URL `https://arxiv.org/abs/2002.07101`.

[9] Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows, 2020. URL `https://arxiv.org/abs/2002.06707`.

[10] Jonas Köhler, Andreas Krämer, and Frank Noé. Smooth normalizing flows, 2021. URL `https://arxiv.org/abs/2110.00351`.

[11] Vincent Stimper, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Resampling base distributions of normalizing flows, 2021. URL `https://arxiv.org/abs/2110.15828`.

[12] Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, and José Miguel Hernández-Lobato. Bootstrap your flow, 2021. URL `https://arxiv.org/abs/2111.11510`.

[13] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Adaptive computation and machine learning series. MIT, Cambridge, MA, 2012. ISBN 9780262018029 0262018020. URL https://www.worldcat.org/title/machine-learning-a-probabilistic-perspective/oclc/781277861?referer=br&ht=edition.

[14] Lorenz Richter, Ayman Boustati, Nikolas Nüsken, Francisco J. R. Ruiz, and Ömer Deniz Akyildiz. Vargrad: A low-variance gradient estimator for variational inference. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[15] Robert Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57:2607–2609, 12 1986. doi: 10.1103/PhysRevLett.57.2607.

[16] Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314(1-2):141–151, November 1999. ISSN 0009-2614. doi: 10.1016/S0009-2614(99)01123-9. URL https://www.sciencedirect.com/science/article/pii/S0009261499011239.

[17] David J. Earl and Michael W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, November 2005. ISSN 1463-9084. doi: 10.1039/B509983H. URL https://pubs.rsc.org/en/content/articlelanding/2005/cp/b509983h.

[18] Jing Huang, Sarah Rauscher, Grzegorz Nawrocki, Ting Ran, Michael Feig, Bert L de Groot, Helmut Grubmüller, and Alexander D MacKerell. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nature Methods*, 14(1):71–73, November 2016. doi: 10.1038/nmeth.4067. URL https://doi.org/10.1038/nmeth.4067.

[19] Jonathan T. Barron. Continuously differentiable exponential linear units. *CoRR*, abs/1704.07483, 2017. URL http://arxiv.org/abs/1704.07483.

[20] Manuel Dibak, Leon Klein, and Frank Noé. Temperature-steerable flows, 2020. URL https://arxiv.org/abs/2012.00429.

[21] Manuel Dibak, Leon Klein, and Frank Noé. Temperature steerable flows and boltzmann generators, 2021. URL https://arxiv.org/abs/2108.01590.

[22] Jérôme Hénin, Tony Lelièvre, Michael R. Shirts, Omar Valsson, and Lucie Delemotte. Enhanced sampling methods for molecular dynamics simulations, 2022. URL https://arxiv.org/abs/2202.04164.

[23] Marylou Gabrié, Grant M. Rotskoff, and Eric Vanden-Eijnden. Adaptive monte carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10), mar 2022. doi: 10.1073/pnas.2109420119. URL https://doi.org/10.1073%2Fpnas.2109420119.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017. URL https://arxiv.org/abs/1710.10903.

[25] Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions, 2017. URL https://arxiv.org/abs/1706.08566.

[26] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows, 2019. URL https://arxiv.org/abs/1905.13177.

[27] Felix Dangel, Frederik Kunstner, and Philipp Hennig. BackPACK: Packing more into backprop. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlrF24twB.

A Derivation of $KL(q_F||q_{\mathcal{N}})$

$$
\begin{aligned}
KL(q_F||q_{\mathcal{N}}) &= \int q_F(z) \log \frac{q_F(z)}{q_{\mathcal{N}}(z)}\, dz && \text{(16a)}\\
&= \int q_F(z) \log \mathcal{Z}_{\mathcal{N}}\, dz + \int q_F(z) \log \frac{q_F(z)}{\tilde{q}_{\mathcal{N}}(z)}\, dz && \text{(16b)}\\
&= \log \mathcal{Z}_{\mathcal{N}} + \int q_F(z) \log \frac{q_F(z)}{\tilde{q}_{\mathcal{N}}(z)}\, dz && \text{(16c)}\\
&= \log \mathcal{Z}_{\mathcal{N}} + \int p_B(x) \log \frac{p_B(x) \cdot \left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}}{\tilde{q}_{\mathcal{N}}(F(x))}\, dx && \text{(16d)}\\
&= \log \mathcal{Z}_{\mathcal{N}} - S_B + \int p_B(x) \log \frac{\left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}}{\tilde{q}_{\mathcal{N}}(F(x))}\, dx && \text{(16e)}\\
&= \log \mathcal{Z}_{\mathcal{N}} - S_B + \int p_B(x) \log \frac{\left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}}{e^{-\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x))}}\, dx && \text{(16f)}\\
&= \log \mathcal{Z}_{\mathcal{N}} - S_B + \mathop{\mathbb{E}}_{x_B \sim p_B}\left[\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x_B)) + \log\left|\det\left(\frac{\partial F(x_B)}{\partial x_B}\right)\right|^{-1}\right] && \text{(16g)}\\
&= \log \mathcal{Z}_{\mathcal{N}} - S_B + \mathop{\mathbb{E}}_{x_B \sim p_B}\left[\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x_B)) - \log\left|\det\left(\frac{\partial F(x_B)}{\partial x_B}\right)\right|\right] && \text{(16h)}
\end{aligned}
$$

with:

- (16a) by definition of the KL divergence
- (16b) by using $q_{\mathcal{N}} = \frac{1}{\mathcal{Z}_{\mathcal{N}}}\tilde{q}_{\mathcal{N}}$
- (16c) by using $\int q_F(z)dz = 1$ (probabilities sum to one)
- (16d) by substitution of $q_F(z)$ by the change of variable formula:

$$
\begin{aligned}
q_F(z)\, dz &= p_B(F^{-1}(z)) \cdot \left|\det\left(\frac{\partial F^{-1}(z)}{\partial z}\right)\right|\, dz\\
&= p_B(x) \cdot \left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}\, dz\\
&= p_B(x)\, dx
\end{aligned}
\qquad (17)
$$

- (16e) by definition of the entropy: $S_B = S(p_B) = -\int p_B(x) \log p_B(x) dx$
- (16f) by using: $\tilde{q}_{\mathcal{N}}(z) = e^{-\frac{1}{2\sigma^2}U_{\mathcal{N}}(x)}$
- (16g) by definition of expectation: $\mathop{\mathbb{E}}_{x_B \sim p_B}\big[f(x_B)\big] = \int p_B(x) f(x) dx$

B Derivation of $KL(p_G||p_B)$

$$
\begin{aligned}
KL(p_G||p_B) &= \int p_G(x) \log \frac{p_G(x)}{p_B(x)}\, dx && \text{(18a)}\\
&= \int p_G(x) \log \mathcal{Z}_B\, dx + \int p_G(x) \log \frac{p_G(x)}{\tilde{p}_B(x)}\, dx && \text{(18b)}\\
&= \log \mathcal{Z}_B + \int p_G(x) \log \frac{p_G(x)}{\tilde{p}_B(x)}\, dx && \text{(18c)}\\
&= \log \mathcal{Z}_B + \int q_{\mathcal{N}}(z) \log \frac{q_{\mathcal{N}}(z) \cdot \left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}{\tilde{p}_B(G(z))}\, dz && \text{(18d)}\\
&= \log \mathcal{Z}_B - S_{\mathcal{N}} + \int q_{\mathcal{N}}(z) \log \frac{\left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}{\tilde{p}_B(G(z))}\, dz && \text{(18e)}\\
&= \log \mathcal{Z}_B - S_{\mathcal{N}} + \int q_{\mathcal{N}}(z) \log \frac{\left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}{e^{-\beta U_B(G(z))}}\, dz && \text{(18f)}\\
&= \log \mathcal{Z}_B - S_{\mathcal{N}} + \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\beta U_B(G(z_{\mathcal{N}})) + \log \left|\det \left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|^{-1}\right] && \text{(18g)}\\
&= \log \mathcal{Z}_B - S_{\mathcal{N}} + \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\beta U_B(G(z_{\mathcal{N}})) - \log \left|\det \left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|\right] && \text{(18h)}
\end{aligned}
$$

with:

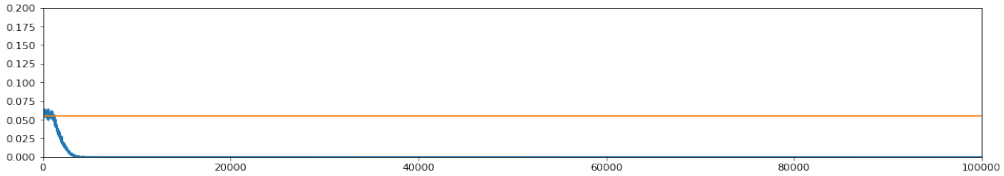
- (18a) by definition of the KL divergence
- (18b) by using $p_B = \tilde{p}_B / \mathcal{Z}_B$
- (18c) by using $\int p_G(x)dx = 1$ (probabilities sum to one)
- (18d) by using the change of variable formula:

$$
\begin{aligned}
p_G(x)\, dx &= q_{\mathcal{N}}(G^{-1}(x)) \cdot \left|\det \left(\frac{\partial G^{-1}(x)}{\partial x}\right)\right| dx\\
&= q_{\mathcal{N}}(z) \cdot \left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1} dx\\
&= q_{\mathcal{N}}(z)\, dz
\end{aligned}
\qquad (19)
$$

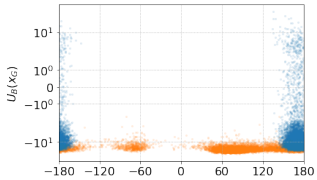
- (18e) by definition of the entropy: $S_{\mathcal{N}} = S(q_{\mathcal{N}}) = -\int q_{\mathcal{N}}(x) \log q_{\mathcal{N}}(x) dx$
- (18f) by using: $\tilde{p}_B(x) = e^{-\beta U_B(x)}$
- (18g) by definition of expectation: $\mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}}\big[f(z_{\mathcal{N}})\big] = \int q_{\mathcal{N}}(z) f(z) dz$

C Optimizing $\mathcal{L}_{KLx}$ leads to mode collapse on `Dialanine`

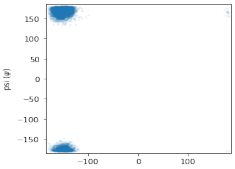
Although most generated samples have low energy, not all of them do (figure 4b) and they only represent a subset of the target distribution (figure 4d), since during training minor modes are progressively lost (figure 4a), until a single mode remains in the 2D projection (figure 4c).



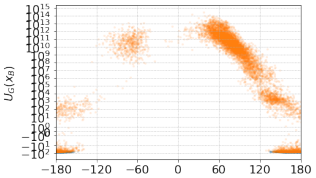
(a) Percentage of generated samples $x_G \sim p_G$ in the minor mode during fine-tuning with $\mathcal{L}_{KLx}$.



(b) U_B energies



(c) p_G projection on (ϕ, ψ)



(d) U_G energies

Figure 4: Results of the fine-tuning with $\mathcal{L}_{KLx}$ on `Dialanine`. See the captions of figure 3 of the main paper for more details.

D Derivation of $KL(q_F||q_\mathcal{N})$ with Importance Sampling

$$\nabla_\theta KL(q_F||q_\mathcal{N}) = \nabla_\theta \left[\mathop{\mathbb{E}}_{x_B \sim p_B} \left[\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_B)) - \log \left| \det \left(\frac{\partial F(x_B)}{\partial x_B} \right) \right| \right] \right] \tag{20a}$$

$$= \nabla_\theta \left[\int p_B^\ddagger(x) \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x)) - \log \left| \det \left(\frac{\partial F(x)}{\partial x} \right) \right| \right) dx \right] \tag{20b}$$

$$= \nabla_\theta \left[\int p_G^\ddagger(x) \left(\frac{p_B(x)}{p_G(x)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x)) - \log \left| \det \left(\frac{\partial F(x)}{\partial x} \right) \right| \right) dx \right] \tag{20c}$$

$$= \nabla_\theta \left[\mathop{\mathbb{E}}_{x_G^\ddagger \sim p_G^\ddagger} \left[\left(\frac{p_B(x_G^\ddagger)}{p_G(x_G^\ddagger)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_G^\ddagger)) - \log \left| \det \left(\frac{\partial F(x_G^\ddagger)}{\partial x_G^\ddagger} \right) \right| \right) \right] \right] \tag{20d}$$

$$= \frac{1}{\mathcal{Z}_B} \cdot \nabla_\theta \left[\mathop{\mathbb{E}}_{x_G^\ddagger \sim p_G^\ddagger} \left[\left(\frac{\tilde{p}_B(x_G^\ddagger)}{p_G(x_G^\ddagger)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_G^\ddagger)) - \log \left| \det \left(\frac{\partial F(x_G^\ddagger)}{\partial x_G^\ddagger} \right) \right| \right) \right] \right] \tag{20e}$$

$$= \frac{1}{\mathcal{Z}_B} \cdot \mathop{\mathbb{E}}_{x_G^\ddagger \sim p_G^\ddagger} \nabla_\theta \left[\left(\frac{\tilde{p}_B(x_G^\ddagger)}{p_G(x_G^\ddagger)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_G^\ddagger)) - \log \left| \det \left(\frac{\partial F(x_G^\ddagger)}{\partial x_G^\ddagger} \right) \right| \right) \right] \tag{20f}$$

with:

- (20a) by taking the gradient of equation 16h.
- (20b) by definition of expectation: $\mathop{\mathbb{E}}_{x_B \sim p_B} [f(x_B)] = \int p_B(x) f(x) dx$. Note the subtle replacement of p_B with $p_B^\ddagger$ which is allowed inside the gradient operator ∇_θ since p_B is not a function of θ.
- (20c) by importance sampling.
- (20d) by definition of expectation.
- (20e) by definition of $\tilde{p}_B = \mathcal{Z}_B p_B$.
- (20f) by noticing that, although $p_G^\ddagger$ is a function of θ, it is not a *differentiated* function of θ. Since it is detached, it is treated as a constant by the gradient operator and the expectation can be sampled in the context of Stochastic Gradient Descent.

E Analysis of the bias when generating deterministic, minimum-energy hydrogen coordinates

In our two-stage architecture, the normalizing flow generator G outputs only heavy atom coordinates x^C, while hydrogen atoms are added at minimum-energy positions x^H by an auxiliary neural network denoted by h. Thus, all-atom coordinates are generated as $\{x^C, x^H\} = h(x^C) = h(G(z))$, and the reverse operation is $z = F(\bar{h}(x^C, x^H))$, noting $\bar{h}$ the operation of stripping H coordinates from an all-atom configuration.

As a result, whereas the generator G is bijective, the complete pipeline $h \circ G$ is not: while $F \circ \bar{h} \circ h \circ G$ is identity in the latent space, $h \circ G \circ F \circ \bar{h} = h \circ \bar{h}$ corresponds to energy minimization with respect to hydrogen atom coordinates, i.e. the *projection* of complete atomic coordinates onto the minimum-energy-hydrogen manifold.

In the spirit of a coarse-graining (CG) approach, the desirable target for the generated distribution p_G^C of heavy atom coordinates is the marginal p_B^C of the target p_B with respect to those coordinates:

$$p_B^C(x^C) = \int p_B(x^C, x^H) dx^H \tag{21}$$

We characterize convergence on the dialanine example by computing the predicted probability of the minor mode $\mathcal{M}$ of dialanine (known to biochemists as the C7ax conformation). The Boltzmann probability of this mode is:

$$P_B(\mathcal{M}) = \int_{\mathcal{M}} p_B(x) dx \tag{22}$$

$$= \int_{\mathcal{M}} p_B(x^C, x^H) dx^C dx^H \tag{23}$$

Now we use the fact that $\mathcal{M}$ is defined solely based on the values of x^C, so that it can be written $\mathcal{M} = \mathcal{M}^C \times \mathbb{R}^{3N_H}$, with $\mathcal{M}^C$ a set of heavy atom coordinates, and N_H the number of hydrogen atoms.

$$P_B(\mathcal{M}) = \int_{\mathcal{M}^C} \left[\int p_B(x^C, x^H) dx^H \right] dx^C \tag{24}$$

$$= \int_{\mathcal{M}^C} p_B^C(x^C) dx^C \tag{25}$$

However, in practice, the optimization of G minimizes the divergence between p_G^C and an auxiliary distribution $\bar{p}_B$ defined by:

$$\bar{p}_B(x^C) = p_B(h(x^C)) \tag{26}$$

Thus any difference between p_B^C and $\bar{p}_B$ introduces a bias in the generation of heavy atom coordinates. Furthermore, the probability p_G of generation of an all-atom configuration x is:

$$p_G(x) = p_G^C(\bar{h}(x))\ \delta(x - h \circ \bar{h}(x)) \tag{27}$$

which is non-zero only on the minimum-energy-hydrogen manifold that is the image of $h \circ \bar{h}$.

Assuming perfect training ($p_G^C = p_{\bar{B}}$), we obtain:

$$p_G(x) = p_B(h \circ \bar{h}(x)) \delta(x - h \circ \bar{h}(x)) \tag{28}$$

The probability of the minor mode as generated by a perfectly trained network is thus:

$$\bar{P}_B(\mathcal{M}) = \int_{\mathcal{M}} p_G(x) dx \tag{29}$$

$$= \int_{\mathcal{M}} p_B(h \circ \bar{h}(x)) \delta(x - h \circ \bar{h}(x)) dx \tag{30}$$

$$\approx \int_{\mathcal{M}} p_B(h \circ \bar{h}(x)) dx \tag{31}$$

where the last step relies on the fact that the conditional Boltzmann distribution of hydrogen atom positions is peaked, that is, p_B is largest around minimal-energy hydrogen coordinates (where $x = h \circ \bar{h}(x)$).

This leads to an importance sampling estimator for this probability based on samples from the reference dataset:

$$\hat{\tilde{P}}_B(\mathcal{M}) := \frac{\sum_{x_B \sim p_B | x_B \in \mathcal{M}} \frac{\tilde{p}_B(h \circ \bar{h}(x_B))}{\tilde{p}_B(x_B)}}{\sum_{x_B \sim p_B} \frac{\tilde{p}_B(h \circ \bar{h}(x_B))}{\tilde{p}_B(x_B)}} \approx 1.21\% \tag{32}$$

which we use as a reference value in Figure 3b.

F Details and proofs for section 3.1 (Estimator variance as a loss)

F.1 Integral quantity of interest

Let us suppose that the use case of our generator is to estimate integral quantities of the form:

$$\mathop{\mathbb{E}}_{p_B}[f]$$

for some given function(s) f.

Let us denote by Q its true value.

$$Q := \mathop{\mathbb{E}}_{p_B}[f] := \int_{x \in X} f(x)\, p_B(x)\, dx \;=\; \mathop{\mathbb{E}}_{p_G}\left[f \frac{p_B}{p_G} \right]$$

This is exact provided that p_G is never 0 where p_B is not.

F.2 Estimation by sampling

In practice, one estimates Q by sampling:

$$Q \simeq \widehat{Q} := \frac{1}{N} \sum_{x_i \in m} f(x_i) \frac{p_B(x_i)}{p_G(x_i)}$$

where $m = (x_1, \ldots, x_N)$ is a mini-batch of points sampled according to p_G. Note however than we do not know p_B, but only $\tilde{p}_B = \mathcal{Z}_B p_B$. We will come back to this point later.

F.3 This estimator is unbiased

Whatever p_G, $\widehat{Q}$ is an approximation of Q, in that for very large mini-batches m, i.e. large N, the estimate $\widehat{Q}$ tends to Q. One then says that the estimator is *unbiased*. The convergence rate is typically in $O(1/\sqrt{N})$. Indeed:

$$\mathop{\mathbb{E}}_{m \sim p_G^N} [\, \widehat{Q}\,] = Q$$

where the expectation is taken over mini-batches of N independent samples, taken according to p_G. To prove this, see that even for just one sample ($N = 1$) one has:

$$\widehat{Q} = f(x_1) \frac{p_B(x_1)}{p_G(x_1)}$$

and thus:

$$\mathop{\mathbb{E}}_{x_1 \sim p_G} \left[\widehat{Q} \right] \;=\; \mathop{\mathbb{E}}_{x \sim p_G} \left[f(x) \frac{p_B(x)}{p_G(x)} \right] \;=\; \int_{x \in X} f(x)\, p_B(x)\, dx \;=:\; Q$$

For a mini-batch of arbitrary size N, one gets the average of N such quantities, each of which are Q on expectation, so one recovers Q again:

$$\begin{aligned}\mathop{\mathbb{E}}_{m\sim p_G^N}[\widehat{Q}] &= \mathop{\mathbb{E}}_{m\sim p_G^N}\left[\frac{1}{N}\sum_{x_i\in m} f(x_i)\frac{p_B(x_i)}{p_G(x_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathop{\mathbb{E}}_{x_i\sim p_G}\left[f(x_i)\frac{p_B(x_i)}{p_G(x_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathop{\mathbb{E}}_{x\sim p_G}\left[f(x)\frac{p_B(x)}{p_G(x)}\right]\end{aligned}$$

because points x_i are sampled independently;

$$= \mathop{\mathbb{E}}_{x\sim p_G}\left[\ f(x)\frac{p_B(x)}{p_G(x)}\ \right] =: Q$$

F.4 Variance of the estimator

Yet, for some distributions p_G, the estimate $\widehat{Q}$ may converge faster than for other ones, in terms of number of samples required to reach a given target accuracy. This is reflected in the *variance* of the estimator $\widehat{Q}$:

$$V = \mathop{\mathbb{E}}_{m\sim p_G^N}[\ (\widehat{Q}-Q)^2\ ]$$

that one would like to be as small as possible. Indeed the typical gap between an estimate $\widehat{Q}$ for a mini-batch and the real value Q can be expected to be of the order of magnitude of V (by definition).

Can we train p_G so as to minimize V?

F.5 Reducing variances over mini-batches to variances over single samples

For a given mini-batch size N, the variance over the choice of mini-batch m is:

$$\begin{aligned}V &= \mathop{\mathbb{E}}_{m\sim p_G^N}[(\widehat{Q}-Q)^2] \\ &= \mathop{\mathbb{E}}_{m\sim p_G^N}[\widehat{Q}^2] - Q^2\end{aligned}$$

As Q^2 is constant (does not depend on p_G), we aim at minimizing only:

$$\begin{aligned}\mathop{\mathbb{E}}_{m\sim p_G^N}[\widehat{Q}^2] &= \mathop{\mathbb{E}}_{m\sim p_G^N}\left[\left(\frac{1}{N}\sum_{x_i\in m} f(x_i)\frac{p_B(x_i)}{p_G(x_i)}\right)^2\right] \\ &= \frac{1}{N^2}\mathop{\mathbb{E}}_{m\sim p_G^N}\left[\sum_{x_i\in m} f^2(x_i)\frac{p_B^2(x_i)}{p_G^2(x_i)} + \sum_{x_i,x_j\in mi\neq j} f(x_i)\frac{p_B(x_i)}{p_G(x_i)}f(x_j)\frac{p_B(x_j)}{p_G(x_j)}\right]\end{aligned}$$

Note that points x_i and x_j are sampled independently, and all points are sampled identically (according to the same law), and thus:

$$\begin{aligned}\mathop{\mathbb{E}}_{m\sim p_G^N}[\widehat{Q}^2] &= \frac{1}{N^2}\sum_{i=1}^{N}\mathop{\mathbb{E}}_{x_i\sim p_G}\left[f^2(x_i)\frac{p_B^2(x_i)}{p_G^2(x_i)}\right] + \frac{1}{N^2}\sum_{i,j=1,i\neq j}^{N}\mathop{\mathbb{E}}_{x_i\sim p_G}\left[f(x_i)\frac{p_B(x_i)}{p_G(x_i)}\right]\mathop{\mathbb{E}}_{x_j\sim p_G}\left[f(x_j)\frac{p_B(x_j)}{p_G(x_j)}\right] \\ &= \frac{1}{N}\mathop{\mathbb{E}}_{x\sim p_G}\left[f^2(x)\frac{p_B^2(x)}{p_G^2(x)}\right] + \frac{N(N-1)}{N^2}\mathop{\mathbb{E}}_{x\sim p_G}\left[f(x)\frac{p_B(x)}{p_G(x)}\right]^2 \\ &= \frac{1}{N}\mathop{\mathbb{E}}_{x\sim p_G}\left[f^2(x)\frac{p_B^2(x)}{p_G^2(x)}\right] + \left(1-\frac{1}{N}\right)Q^2\end{aligned}$$

The variance (without forgetting any constant term) thus interestingly rewrites as:

$$V = \frac{1}{N}\left(\mathop{\mathbb{E}}_{x\sim p_G}\left[f^2(x)\frac{p_B^2(x)}{p_G^2(x)}\right] - Q^2\right)$$

which can be interpreted as: the variance of an estimator based on N samples is $\frac{1}{N}$ times the variance of the estimator based on a single sample. This implies that the variance behaves as $O(\frac{1}{N})$ and thus the typical error (standard deviation) is $O(\frac{1}{\sqrt{N}})$.

F.6 Optimizing the variance w.r.t. p_G

Based on the variance formula above, one can consider that the quality (or rather: expected error) of the generator G can be quantified as:

$$C(p_G) \;=\; \mathop{\mathbb{E}}_{x\,\sim\,p_G}\left[\; f^2(x)\frac{p_B^2(x)}{p_G^2(x)}\;\right]$$

and we would like to minimize it w.r.t. p_G.

If f can be any bounded function over the space X of points x, then one can deduce the following optimization criterion:

$$C(p_G) \;=\; \mathop{\mathbb{E}}_{x\,\sim\,p_G}\left[\; \frac{p_B^2(x)}{p_G^2(x)}\;\right]$$

Note that this resembles a KL divergence without the logarithm, and is also equal to:

$$C(p_G) = \mathop{\mathbb{E}}_{x\sim p_G}\left[e^{2\beta(U_G - U_B)}\right]\frac{1}{\mathcal{Z}_B^2}$$

This loss is also equal to:

$$C(p_G) \;=\; \mathop{\mathbb{E}}_{x\,\sim\,p_B}\left[\; \frac{p_B(x)}{p_G(x)}\;\right]$$

though this is not directly exploitable.

Note that if function f that needs to be integrated is known, it should be used explicitly in the criterion to optimize!

F.7 Special case: estimating free energy differences

An interesting and classic case in practice is to compute the free energy difference ΔF_{BC} between the sate being sampled (with energy U_B) and an alternate state defined by energy U_C. Then:

$$f(x) = e^{-\beta(U_C(x) - U_B(x))}$$

and

$$e^{-\beta\Delta F_{BC}} = \mathop{\mathbb{E}}_{x\sim p_B}[f]$$

G Proof of L2 loss pseudo-gradient descent convergence (section 4)

We study here the optimization properties of the masked L^2 loss, with detached means.

G.1 Notations

Given a dataset of points x_i, we denote by

$$r_i = \log\frac{p_B(x_i)}{p_G(x_i)}$$

the log-ratio of the target and generated densities at point x_i.

Differences of log-ratios satisfy:

$$r_i - r_j = \log \frac{p_B(x_i)}{p_G(x_i)} - \log \frac{p_B(x_i)}{p_G(x_i)} = \log \frac{\tilde{p}_B(x_i)}{p_G(x_i)} - \log \frac{\tilde{p}_B(x_i)}{p_G(x_i)}$$

where $\tilde{p}_B = (\log \mathcal{Z}_B)\, p_B$ is easily computable, which makes such differences easily computable, to the opposite of the log-ratios r_i themselves.

Let us note that:

$$\mathop{\mathbb{E}}_{x_j \sim p_B}[r_j] = KL(p_B||p_G)$$

and similarly:

$$\mathop{\mathbb{E}}_{x_i \sim p_G}[r_i] = -KL(p_G||p_B)$$

G.2 Pairwise L2 loss

G.2.1 Definition

The pairwise L^2 loss (simple version) is defined as:

$$L(p_G, p_B) = \mathop{\mathrm{var}}_{x_i \sim p_G}(r_i) = \mathop{\mathbb{E}}_{x_i \sim p_G}\left[\left(r_i - \mathop{\mathbb{E}}_{x_j \sim p_G}[r_j]\right)^2\right]$$

As a side remark, let us note that this is equal to:

$$L(p_G, p_B) = \frac{1}{2} \mathop{\mathbb{E}}_{x_i, x_j \sim p_G}\left[(r_i - r_j)^2\right]$$

but we will not use this property here. The proofs are the same as for the mixed sampling case (cf below).

G.2.2 Global minimum

This loss is also equal to:

$$L(p_G, p_B) = e^{D_2(p_G||p_B)}$$

where D_2 is the Rényi divergence of order 2, a measure of divergence between distributions. D_2 is a f-divergence and in particular it is jointly convex. As a consequence, the only minimum is the global one, reached at $p_G = p_B$.

G.3 Masked L2 loss with detached means

G.3.1 Definition

The masked L^2 loss (simple version) with detached means is defined as:

$$L_{MD}(p_G, p_B) = \mathop{\mathbb{E}}_{x_i \sim p_G}\left[\left(r_i - K^\ddagger\right)_+^2\right]$$

where $a_+^2 = a^2$ if $a > 0$ and 0 otherwise, and where $K = \mathbb{E}_{x_j \sim p_G}[r_j] = -KL(p_G||p_B) \leqslant 0$ is not differentiated (considered as a constant at every time step of the gradient descent); we say $K^\ddagger$ is *detached*, following PyTorch vocabulary.

This definition is motivated as follows:

- masking $(r_i - K)$ with $()_+$ to consider it only when positive has the consequence that r_i will be only asked to decrease. Since p_G is a probability distribution, this implies that some other r_j will increase to compensate (p_G cannot increase everywhere), but at least this will not be done by the gradient descent, hence not in the worst possible direction (make x_j as unlikely as possible, and this as fast as possible, i.e. make it as unrealistic as possible), but rather in the smoothest possible way (push the probability mass to regions where it is more needed).
- not detaching K would ask it to increase, and thus to decrease values of p_G at most points x_j.

G.3.2 Global minimum

This loss is non-negative, and 0 can be reached if all r_i are equal (note that 0-loss implies that no r_i is greater than the mean K, and consequently no r_i can be strictly less than the mean K as well, otherwise the mean would be lower than itself). As previously, this is the case if and only if p_G is proportional to $\tilde{p}_B$ and thus equal to p_B (see the end of the convergence proof below for more details).

G.3.3 Optimization by partial gradient descent

However, since only part of the loss is differentiated (K is considered a constant though changing with p_G), then strictly speaking the optimization process is not a gradient descent, as the loss is different at each time step. Therefore one needs to check that this optimization process does converge, and to the global minimum.

This task is hindered by the fact that the constraint that the total mass of p_G has to remain 1 is handled implicitly by the normalizing flow, and so the precise way the gradient w.r.t. p_G is replaced with a variation δp_G that preserves the total mass depends on the architecture and the neural network weights.

Total variation of log-ratios Let us study the variation of K induced by a (partial) gradient step:

$$\delta K = \delta \left(\mathop{\mathbb{E}}_{x \sim p_G} [r] \right) = \delta \left(\int p_G \, r \, dx \right) = \mathop{\mathbb{E}}_{x \sim p_G} [\delta r] + \int r \, \delta p_G \, dx$$

Note that:

$$p_G(x_i) = e^{\log p_G(x_i)} = e^{-r_i + \log p_B(x_i)} = e^{-r_i} p_B(x_i)$$

As $\mathbb{E}_{x \sim p_G}[1] = 1$ we have $\mathbb{E}_{x \sim p_B}[e^{-r(x)}] = 1$, and this for any p_G or equivalently for any associated r. As a consequence, the variation of $\mathbb{E}_{x \sim p_B}[e^{-r}]$ w.r.t. to any realizable change δr (p_B being fixed and p_G varying) is necessarily 0:

$$\delta \left(\mathop{\mathbb{E}}_{x \sim p_B} [e^{-r}] \right) = \mathop{\mathbb{E}}_{x \sim p_B} [e^{-r} \delta r] = 0$$

which rewrites as

$$\mathop{\mathbb{E}}_{x \sim p_G} [\delta r] = 0$$

Consequently δK can be simplified as:

$$\delta K = \int r \, \delta p_G \, dx$$

and rewritten as:

$$\delta K = \int (r - K) \, \delta p_G \, dx$$

since $K \int \delta p_G = 0$ as p_G has conserved total mass = 1.

Now, note that the gradient descent step is asking to decrease all r that are greater than K. Since $r = \log p_B / p_G$, this means increasing p_G for such points where $r > K$. So $\delta p_G > 0$ when $r - K > 0$.

For points where $r < K$, p_G is not asked to change, but the conservation of mass makes that (at least some of) such points will have their probabilities decreased, to produce the extra mass needed by the previous points above ($r > K$). Thus $\delta p_G \leqslant 0$ where $r < K$.

In the end, for all points, $(r - K)\delta p_G \geqslant 0$ and consequently:

$$\delta K \geqslant 0$$

See Section G.3.4 below for more details about this proof.

Consequences Therefore, K is increasing with time. This is good news, as $K = -KL(p_G||p_B)$: this means that with this optimization process, p_G is getting closer to p_B.

As K is actually strictly increasing as long as p_G is not p_B, we can conclude that the optimization process will lead to the desired global optimum.

Another way to see this is that K is an increasing, upper-bounded value (bounded by 0) and this will converge. When K converges (possibly to a non-0 value), then the training criterion becomes stable with time and the optimization process becomes a real gradient descent w.r.t. r_i. Therefore a local minimum of this loss (for fixed limit K) will be reached. Now, the gradient of this fixed-K loss is 0 when all r_i are either equal to or less than K. If at such a minimum one r_i was strictly less than K, we would get that the average of all r_i (according to p_G) would be strictly less than K, while it is precisely K. Therefore all r are equal and the global minimum is reached.

G.3.4 More details on the proof

We explain here in more details the links between the signs of $\delta p_G(x)$ and of $(r(x) - K)$, that is, that their product is never negative.

To see this, we need to detail how $\delta p_G(x)$ is obtained. We are optimizing the following criterion :

$$L_{MD}(p_G, p_B) = \mathop{\mathbb{E}}_{x_i \sim p_G} \left[\left(r_i - K^{\ddagger} \right)_+^2 \right]$$

with $K = \mathbb{E}_{x_j \sim p_G}[r_j] = -KL(p_G||p_B)$, where:

- the sampling distribution p_G over which the expectation is performed is not differentiated, i.e. the minibatch points $x_G = (x_i)$ are detached; this is similar to what is done with VarGrad [14] ;
- the log ratios $r_i = \log \frac{p_B(x_i)}{p_G(x_i)}$ are differentiated, with respect to the parameters of the modeled distribution p_G, and this will induce a desired variation for p_G ;
- K is not differentiated ;
- only points for which $r_i > K$ are actually taken into account in the criterion.

These two last points differ from VarGrad [14]; practice shows that they are required for the optimization process to go well. Interestingly, this pseudo gradient descent can be proven theoretically to converge towards the right minimum. We will prove that $K = -KL(p_G||p_B)$ increases with time, and tends to 0, and thus p_G gets closer to p_B at each step and finally converges to the target.

The hypotheses for this theoretical study are only that:

- the support of p_G includes the one of p_B ;
- the neural network is expressive enough (for the pseudo gradient direction to be followed).

Desired variations Let us first note that the log ratio is $r(x) = \log \frac{p_B(x)}{p_G(x)}$, i.e. $p_G = e^{-r} p_B$. As a consequence, possible variations of r or p_G are linked as follows:

$$\delta r = -\frac{1}{p_G} \delta p_G$$

$$\delta p_G = -p_G \, \delta r$$

The (opposite of the) derivative of $L_{MD}(p_G, p_B)$ with respect to r yields the desired variation: $\delta r_{\text{desired}}(x) = -2(r(x) - K)_+ \, p_G(x)$ having taken into account that only some parts of the criterion are differentiated as explained above. This translates into a desired distribution variation : $\delta p_{G_{\text{desired}}} = 2(r - K)_+ \, p_G^2$ This is 0 for points x such that $r(x) \leqslant K$ and positive otherwise.

Constrained variations However p_G is a probability distribution and is constrained to sum up to 1. How a practical training step projects the desired probability variation $\delta p_{G_{\text{desired}}}$ onto a realizable variation $\delta p_{G_{\text{realizable}}}$ depends on the normalizing flow architecture, its weights, and its expressivity, in

a complex manner. If the neural network is expressive enough though, the desired variation $\delta p_{G_{\text{desired}}}$ is realizable up to the mass constraint. We will study here this ideal case, where the network is sufficiently expressive, and reason in the functional space of possible functions p_G, forgetting about the network (that will be able to express the realizable variation $\delta p_{G_{\text{realizable}}}$ anyway).

There are several ways to project $\delta p_{G_{\text{desired}}}$ onto a realizable variation $\delta p_{G_{\text{realizable}}}$ that satisfies the mass constraint. We do as follows:

For points x such that $r(x) > K$:

$$\delta p_{G_{\text{realizable}}}(x) = \delta p_{G_{\text{desired}}}(x)$$

and for other points:

$$\delta p_{G_{\text{realizable}}}(x) = -\mu$$

where μ is the following constant (i.e. not depending on x):

$$\mu \;=\; \frac{1}{|\Omega^-|} \int_{x \in \Omega} \delta p_{G_{\text{desired}}}(x)\, dx \;\geqslant\; 0$$

where Ω is the support of p_G and where Ω^- is the subset of Ω where $r(x) < K$. This construction is meant so that:

$$\int_{x \in \Omega} \delta p_{G_{\text{realizable}}}(x)\, dx \;=\; 0$$

which is the condition for p_G to remain a probability distribution (the mass is kept constant).

Note that this projection of the desired pseudo-gradient over the set of variations satisfying the mass constraint is not the standard orthogonal one.

Impact on the average K By construction, the projected gradient will decrease the criterion L_{MD} for fixed K and fixed sampling distribution. However p_G and consequently K evolve with time.

The variation of $K = \mathbb{E}_{x_j \sim p_G}[r_j]$ is:

$$\delta K = \delta \left(\int p_G\, r \right) = \int r\, \delta p_G + \int p_G\, \delta r$$

As explained earlier, the last term is 0: $\int p_G\, \delta r = \int -\delta p_G = 0$ for any realizable variation δp_G.

The variation of K thus becomes:

$$\delta K = \int r\, \delta p_G = \int (r - K)\, \delta p_G = \int_{\Omega^+} (r - K) \delta p_G + \int_{\Omega^-} (r - K) \delta p_G$$

as $\int \delta p_G = 0$, and where Ω^+ is the subset of Ω where $r(x) > K$.

Considering for δp_G our pseudo gradient $\delta p_{G_{\text{realizable}}}$, we obtain:

$$\delta K = 2 \int_{\Omega^+} (r - K)^2 p_G^2 - \mu \int_{\Omega^-} (r - K)$$

where the fist term is non-negative, and $\mu \geqslant 0$ and $r - K < 0$ on Ω^-. Consequently $\delta K \geqslant 0$. Therefore K keeps increasing with time.

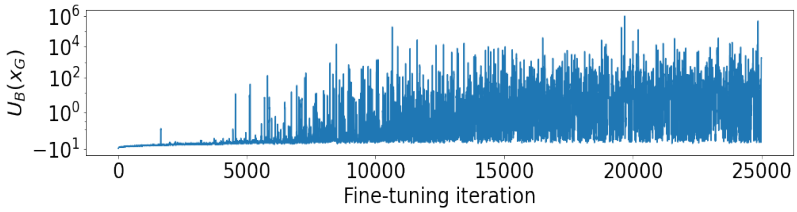
Moreover, $\delta K > 0$ as long as p_G is not proportional to p_B on the support of p_G. As K increases and is upper-bounded by 0, K converges. Convergence implies $\delta K = 0$, and therefore that p_G is proportional to p_B on the support of p_G. The hypothesis on the supports then implies that $p_G = p_B$.

H Not detaching K: experimental results

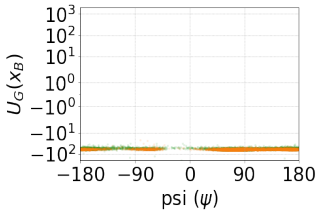
The masked L^2 loss with detached means is defined as:

$$\mathcal{L}_{L^2_+}(\mathbf{x}^\ddagger_G) = \sum_{i=1}^{n} \left[\frac{1}{n} \cdot \left[\left(r(x^\ddagger_{G,i}) - K^\ddagger \right)_+^2 \right] \right] \tag{33}$$

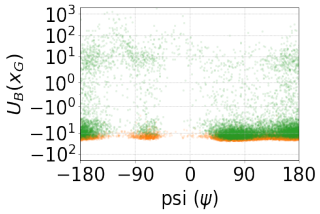
When K from equation 33 is not detached, the potential energy U_B of the generated samples x_G is highly unstable during fine-tuning (blue curve of figure 5a in this appendix), but when K is detached the potential energies remain stable (green curve of figure 3i of the main paper). Very similar results are obtained when the mask of equation 33 is omitted, thereby illustrating experimentally that VarGrad [14] does not allow for stable data-free fine-tuning.



(a) U_B energy of generated samples $x_G \sim p_G$ during fine-tuning.



(b) U_B energies with $\mathcal{L}_{L^2_+}$ with no detach of K



(c) U_G energies with $\mathcal{L}_{L^2_+}$ with no detach of K

Figure 5: Results of a fine-tunings on `Dialanine` with a slightly modified version of $\mathcal{L}_{L^2_+}$ where K (from equation H) is not detached. See caption of figure 3 of the main paper for more details.

I Optimization pitfalls induced by the discretization of distributions into minibatches

In this section we list various optimization pitfalls encountered during gradient descents over a "distances" or divergences between probability distributions, and bring practical or theoretical recommendations to avoid them.

Historically for us, the results below were strong motivations to search for new optimization criteria with better optimization properties. We did not include this part in the main paper for space reasons and because these considerations are not essential, though they help understand the theoretical context of our study. The proofs and details are deferred to the next section, for readability reasons.

I.1 Discretization issues with Kullback-Leibler and remedies

To motivate this study of discretization and normalization issues, we start with an intruiguing fact.

Optimization of naively-discretized Kullback-Leibler does not converge towards the target. The main property of Kullback-Leibler divergence, as a measure of "distance" between probability distributions, is Gibbs inequality: $KL(p_B||p_G) \geqslant 0$ for any p_B, p_G, with equality if and only if $p_B = p_G$. Without the constraint of being probabilities (unit total mass), Gibbs inequality does not hold anymore, and thus minimizing $KL(p_B||p_G)$ w.r.t. p_G in the space of all distributions leads to an unexpected behavior.

Proposition I.1 (Unconstrained-mass pitfall) *The gradient descent* $\frac{dp_G}{dt} = -\nabla_{p_G} KL(p_B||p_G)$, *starting from* $p_{G,0}$ *and without constraining* $\int_X p_G$ *to be 1, yields, for large times* t*:*

$$\forall x, \quad p_{G,t}(x) \simeq \sqrt{2t}\sqrt{p_B(x)}$$

Thus a lack of normalization will push p_G to get infinite mass, and even correcting $p_{G,t}$ by its total mass will not yield p_B, but $\sqrt{p_B}$. Clearly, $p_G = p_B$ is not the minimizer of $KL(p_B||p_G)$, and indeed with $p_{G,t} = \sqrt{2t}\sqrt{p_B}$ one gets $KL(p_B||\sqrt{2tp_B}) = \frac{1}{2}H(p_B) - \frac{1}{2}\log(2t) \quad \ll \quad 0 = KL(p_B||p_B)$ for high t.

This pitfall still stands even if one considers a parameterized model for p_G that always satisfies the constraint $\int_X p_G = 1$, such as the output of a normalizing flow, if the discretization is inadequate. Indeed the above also applies to continuous distributions discretized on a minibatch m of samples:

$$p_{B|m}(x) := \sum_{i \in m} \delta_{x=x_i} p_B(x_i) \qquad \text{and} \qquad p_{G|m}(x) := \sum_{i \in m} \delta_{x=x_i} p_G(x_i)$$

The total mass of these distributions is not 1, even if normalized by the number of samples (minibatch size). It can be arbitrarily high, as $p_G(x_i)$ is a probability density. As a consequence:

Corollary I.1 (Unproper-minibatch-normalization pitfall) *A gradient descent w.r.t. θ, parameters of p_G, to minimize $KL(p_{B|m}\,||\,p_{G|m})$, with always the same minimibatch m, will follow similar exploding dynamics.*

Variance over minibatches. Note that the followed gradient $-\nabla_{p_G} KL(p_B||p_G) = \frac{p_B}{p_G}$ is always positive, at all locations x, and thus at each time step, the density increases at sampled points of the current minibatch, and is implicitely reduced at other locations by the normalizing flow architecture. This positive pressure will cancel out when integrated over the whole space, as one cannot increase densities at all points simultaneously while keeping the total mass constant; in the end, pressures matter only relatively to the average one (densities at locations x with relatively weak pressure will decrease). In practice this induces a lot of variance, as one needs to wait for mini-batches to have covered the whole space for the average gradient to be informative, furthermore hoping that the resulting sampling will be sufficiently uniform. This slows down and may significantly hinder the optimization of quantities such as KL strongly relying on global quantities (unit mass).

Correct normalization. The problem above disappears with correct normalization over minibatches, ensuring that the distributions inside KL have unit mass:

$$p_B^{d|m}(x) := \frac{1}{\sum_{i\in m} p_B(x_i)} \sum_{i\in m} \delta_{x=x_i} p_B(x_i) \qquad \text{and} \qquad p_G^{d|m}(x) := \frac{1}{\sum_{i\in m} p_G(x_i)} \sum_{i\in m} \delta_{x=x_i} p_G(x_i)$$

As $p_B^{d|m}$ and $p_G^{d|m}$ are probability distributions over minibatch points, the divergence $KL(p_B^{d|m}\|p_G^{d|m})$ makes sense, as well as its gradient w.r.t. the parameters θ of the generative model p_G, as redistributing the mass within the minibatch, without pulling extra mass from non-sampled points. Therefore each minibatch gradient is informative, and convergence is much faster. On the opposite, the divergence $KL(p_B^{d|m}\|p_G)$ and the former discretization of the divergence $KL(p_B||p_G)$ over the minibatch both suffer from manipulating densities $p_G(x_i)$ that are not constrained to sum to 1 over the minibatch, leading to the previously detailed issues.

I.2 Reducing the variance of estimators induced by discretization

Losses vs. estimators of them by discretization. One important thing is not to confuse a quantity, such as $Q = KL(p_B||p_G)$, with estimators $\widehat{Q}$ of it, such as the approximations obtained by discretization over mini-batches of samples. Another important thing is not to confuse the gradient $\nabla\widehat{Q}$ of a good estimator of a quantity with a good estimator $\widehat{\nabla Q}$ of the gradient of that quantity. This is the latter one that we aim at finding, and that we study now.

We would like to estimate the gradient of $KL(p_B\|p_G)$ (or of another similar criterion) w.r.t. the generator parameters θ.

Such gradient is of the form:

$$A := \int_X g \; d\mu = \int_{x\in X} g(x) \; d\mu(x)$$

where for instance in the case of $\nabla_\theta KL(p_B\|p_G) = \int_X \frac{p_B}{p_G}\frac{dp_G}{d\theta}$, one could choose $g(x) = \frac{p_B(x)}{p_G(x)}\frac{dp_G(x)}{d\theta}$ and $d\mu = dx$, or $g = \frac{p_B}{p_G^2}\frac{dp_G}{d\theta}$ and $d\mu(x) = p_G(x)dx$, depending on the sampler. Yet, all we can do is to sample a minibatch m containing n samples, chosen i.i.d. according to $d\mu$:

$$\widehat{A} := \frac{1}{n}\sum_{i\in m} g(x_i)$$

This estimator is unbiased: $\mathbb{E}_m[\widehat{A}] = A$, i.e. on average over all possible minibatches, $\widehat{A}$ becomes A. The approximation error, or *estimator variance*, can be shown to be:

$$\mathop{\mathbb{E}}_m[(A-\widehat{A})^2] = \frac{1}{n}\left(\mathop{\mathbb{E}}_x\left[g^2\right] - A^2\right)$$

Stabilizing trick to reduce estimator variance. Note that adding $+\int_X K\frac{dp_G}{d\theta}$ to the gradient A, for some constant K, does not change it, as $\int_X \frac{dp_G}{d\theta} = \frac{d}{d\theta}\int_X p_G = \frac{d}{d\theta}(1) = 0$. A new expression of our target quantity A, with its associated unbiased estimator $\widehat{A}'$, is thus:

$$A = \int_X g = \int_X g + K\frac{dp_G}{d\theta} \qquad \text{and} \qquad \widehat{A}' := \frac{1}{n}\sum_{i\in m} g(x_i) + K\frac{dp_G(x_i)}{d\theta}$$

Minimizing the variance of the estimator $\widehat{A}'$ w.r.t. K yields $K^* = -\mathbb{E}\left[g\frac{dp_G}{d\theta}\right] / \mathbb{E}\left[\frac{dp_G}{d\theta}^2\right] \simeq -\sum_j g(x_j)\frac{dp_G(x_j)}{d\theta} / \sum_j (\frac{dp_G(x_j)}{d\theta})^2$ where the approximation is performed through running means over past minibatches.

This variance-reduced estimator of the gradient can easily be obtained by just adding $+K^*\int_X p_G$ to the optimization criterion before discretization over the minibatch. The corresponding gradient descent will be more robust to minibatch discretization.

Normalization mistakes are removed by the stabilizing trick (on average). Applying the trick above removes the normalization mistakes of the naive discretization. Indeed, on average, i.e. on expectation over minibatches, one can show that the total mass within a minibatch is preserved by a gradient step based on the trick-corrected gradient estimator. This was not the case with the naively-discretized KL gradient, which always asks for increasing the mass at each point of each minibatch.

J Proofs and details of the previous section

J.1 Reminder about KL divergence

The quantity:

$$KL(p||q) = \int_X p\log\frac{p}{q}$$

is called a "divergence" and has the following properties, when applied to 2 probability distributions p and q defined over a space X:

- $KL(p||q) > 0$ for any different p and q
- $KL(p||q) = 0$ if and only if $p = q$

This is known as Gibbs inequality and makes KL usable as a criterion to measure how far two probability distributions are from each other. KL is not a distance in the mathematical sense though. In particular, $KL(p||q)$ is not equal to $KL(q||p)$ in general.

Note 1: it is important that p and q be *probability* distributions, i.e. $p \geqslant 0$ and $\int_X p = 1$, and similarly for q. Without these constraints, Gibbs inequality is not true anymore, and thus minimizing $KL(p||q)$ w.r.t. q might lead to a solution q^* different from p.

Note 2: formulas containing the symbol $\int_X$ are generically true for any measure over X, and not necessarily just the Lebesgue measure. For instance, one can replace $\int_X$ with $\sum_i \delta_{x=x_i}$, i.e. consider a discrete set of points $\{x_i\}$, or a weighted set: $\sum_i w_i\delta_{x=x_i}$ with $w_i \geqslant 0$.

J.2 First pitfall: dynamics of minimizing $KL(p||q)$ w.r.t. q without constraining $\int_X q$ to be 1

Let us start from $q_{t=0} = q_0$ and minimize $KL(p||q_t)$ by gradient descent w.r.t. q_t directly (no intermediate parameterization), i.e. $\frac{dq}{dt} = -\nabla_q KL(p||q_t)$

To obtain the expression of the functional gradient $\nabla_q KL$, let us consider any infinitesimal variation δq of q : the quantity $KL(p||q) = \int_X p(x)\log\frac{p(x)}{q(x)}dx = -\int_X p(x)\log q(x)dx + \text{Constant}$ would change by:

$$\delta(KL(p||q))(\delta q) = -\int_X \frac{p(x)}{q(x)}\delta q(x)dx$$

As the (L^2) gradient $\nabla_q f(q)$ of a function f is defined as the unique distribution v such that $\int_X v(x)\,\delta q(x)dx = \delta(f(q))(\delta q) + o(\delta q) \quad \forall \delta q$, one has:

$$\nabla_q KL(p||q) = -\frac{p}{q}$$

Hence the dynamics rewrite:

$$\frac{dq}{dt} = \frac{p}{q}$$

in the sense that $\forall x,\ \frac{dq_t(x)}{dt} = \frac{p(x)}{q_t(x)}$. Let us note that this implies $\frac{dq}{dt} \geqslant 0\ \forall x, t$ and that as a consequence, $q(x)$ increases with time for all x, getting farther and farther away from the missing constraint $\int_X q = 1$. Indeed:

$$\frac{dq^2}{dt} = 2p$$

and hence:

$$q^2(t) = 2pt + q_0^2$$

$$q(t) = \sqrt{2pt + q_0^2}$$

Thus, for large times t,

$$q(t) \simeq \sqrt{2t}\sqrt{p}$$

in the sense that:

$$\forall x,\ \ q_t(x) \simeq \sqrt{2t}\sqrt{p(x)}$$

which shows that:

- there is a lack of normalization (q will get infinite mass),
- even correcting by the total mass of q will not yield p, but $\sqrt{p}$,
- $q = p$ is not the minimizer of $KL(p||q)$.

Indeed with $q_t = \sqrt{2t}\sqrt{p}$ one gets:

$$KL(p||\sqrt{2tp}) = \frac{1}{2}H(p) - \frac{1}{2}\log(2t) \quad \ll\ 0 = KL(p||p)$$

for $t > \frac{1}{2}e^{H(p)}$.

J.3 Extension to normalizing flows keeping full mass constant by design

The above also applies to distributions discretized on a minibatch m:

$$p(x) = p_{B|m}(x) := \sum_i \delta_{x=x_i} p_B(x_i)$$

and

$$q(x) = p_{G|m}(x) := \sum_i \delta_{x=x_i} p_G(x_i)$$

The total mass of these distributions is not 1. It can be arbitrarily high, as $p_G(x_i)$ is a density (and not a probability: it is not constrained to be less than 1). As a consequence, a gradient descent w.r.t. θ, parameters of p_G, to minimize $KL(p_{B|m}||q_{G|m})$, with always the same minimibatch m, will follow similar exploding dynamics.

J.4 Gradient of the estimator vs. estimator of the gradient

The gradient $\nabla_q^{L^2}$ with respect to the distribution q does not take into the fact that q should remain a probability distribution, i.e. sum up to 1. One should *project* this gradient onto the set of possible variations of q. This can be done for instance by considering $\nabla_q^{L^2} KL - \int_X \nabla_q^{L^2} KL$, i.e. removing its mean. This would definitely change the dynamics studied in the previous section.

In our implementation with "normalizing flows", the gradient $\nabla_\theta^{L^2}$ with respect to the parameters θ of the probability distribution q_θ does implicitely take into account the fact that q should remain a probability distribution, in that by construction all q_θ are probability distributions: with a "normalizing flow", there is no way to escape the manifold of distributions which sum up to 1.

Note the difference between:

- the gradient $\nabla_\theta^{L^2}$ of $KL(p^d \| q^d)$ between discretized distributions (which is the correct way according to the previous sections): $\sum_i \frac{d \log q(x_i)}{d\theta} \left(q^d(x_i) - p^d(x_i)\right)$, the first term coming from the derivative of the log of the normalizing factor $\frac{1}{\sum_i q(x_i)}$
- and the discretization of the gradient $\nabla_\theta^{L^2}$ of $KL(p^d \| q)$ (which is an incorrect way): $-\sum_i p^d(x_i) \frac{d \log q(x_i)}{d\theta}$.

The incorrect way misses a term, which acts as if it had a supplementary term $-\frac{1}{\sum_i q(x_i)} \sum_i \frac{dq(x_i)}{d\theta}$ which leads to be a positive additive term in a gradient *descent*: all $q(x_i)$ are increased.

This is similar to the actor-critic approach in reinforcement learning, where the comparison $\left(q^d(x_i) - p^d(x_i)\right)$ improves the dynamics of the training, pushing the probability flow in the right direction (the sign indicates whether more probability mass is needed or the opposite), while without the critic p^d the training would be far less stable and require much more time.

J.5 Stabilizing trick

First note that for any parameterized probability distribution $q = p_G = p_G^{(\theta)}$:

$$\forall \theta, \quad \int_X p_G = 1$$

and consequently:

$$\frac{d}{d\theta} \int_X p_G = \int_X \frac{dp_G}{d\theta} = (0, 0, 0 \dots 0)$$

with as many 0 as parameters θ. Thus any gradient formula of the form $\int_X \frac{dq}{d\theta} f = \int_{x \in X} \frac{dq}{d\theta}(x) f(x) dx$ satisfies:

$$\int_X \frac{dq}{d\theta} f = \int_X \frac{dq}{d\theta}(f + K) \quad \forall K \in \mathbb{R}^{|\Theta|}$$

for any additive constant K (which is a vector). Thus we can form many different estimators of $\int_X \frac{dq}{d\theta} f$ by picking a value for K and discretizing $\int_X \frac{dq}{d\theta}(f + K)$ over a minibatch. What we will do in next section is to note that our gradient writes in that form ($\int_X \frac{dq}{d\theta} f$), and choose within this family of estimators the one with the least variance.

Similarly,

$$\int_X \frac{d \log q}{d\theta} f = \int_X \frac{d \log q}{d\theta}(f + Kq) \quad \forall K \in \mathbb{R}^{|\Theta|}$$

Note: for readability purposes, we used the abusive notation $f + K$, which stands for $f . * (1, 1, 1 \dots) + K$, and the product with $\frac{dq}{d\theta}$ is done coefficient-wise for K. To be more precise:

- $f(x) \in \mathbb{R}$, so $\frac{dq}{d\theta} f$ reads $f(x) \frac{dq(x)}{d\theta}$: real $\times$ vector multiplication,
- $K \in \mathbb{R}^{|\Theta|}$ is a vector (as many coefficients as parameters θ) and $\frac{dq}{d\theta} K$ reads $\frac{dq}{d\theta} .* K = \left(\frac{dq(x)}{d\theta_j} K_j\right)_j$ which is a coefficient-wise multiplication, yielding a vector of same size.

Implementing the trick very simply as an addition to the loss Instead of manipulating the gradient, this trick can be implemented directly by changing the loss to be optimised. Indeed adding $+\int_X \frac{dq}{d\theta} K$ to the gradient amounts to adding $+K \int_X q$ to the optimization criterion (with *detached* K and *without* replacing $\int_X q$ by its expected value, 1).

J.6 Minimizing the variance of the estimator of the gradient

J.6.1 Set-up

Consider the case where one wants to minimize $KL(p\|q)$ (or another criterion) w.r.t. q. At some point during gradient computations we would like to compute a quantity of the form:

$$A := \int_X g = \int_{x \in X} g(x) dx$$

but all we can do is sample a minibatch m containing n samples, chosen i.i.d. and uniformly over X:

$$B := \frac{1}{n} \sum_{i \in m} g(x_i)$$

NB: this i.i.d. and uniform hypotheses will be important in the sequel. One could imagine that points are sampled on purpose far away from each other, or from another distribution. In which case, the proves below have to be adapted.

We know that on average over all possible minibatches, B becomes A:

$$\mathop{\mathbb{E}}_m[B] = A$$

and this can be shown by: $\mathbb{E}_m[B] = \frac{1}{n} \mathbb{E}_m \left[\sum_{i \in m} g\right] = \mathbb{E}_x \left[g(x)\right] = A$ as minibatch points are i.i.d. and uniformely sampled over X (see next section for details).

Yet for any minibatch, B is rarely exactly A. In particular if X is large, n is small, or g varies quickly, B is not likely to be exactly A. We thus want to study the approximation error:

$$\mathop{\mathbb{E}}_m[(A - B)^2]$$

in order to minimize it w.r.t. the parameter K above.

NB: computations below are done with integrals and samplers uniform over X.

J.6.2 Minimizing the variance

$$\mathop{\mathbb{E}}_m[(A - B)^2] = A^2 + \mathop{\mathbb{E}}_m \left[B^2\right] - 2A \mathop{\mathbb{E}}_m [B] = -A^2 + \mathop{\mathbb{E}}_m \left[B^2\right]$$

as A does not depend on m. The $-A^2$ term is constant (depends neither on m, nor K). Let us study the other term, in order to optimize it w.r.t. K later:

$$B^2 = \left(\frac{1}{n} \sum_{i \in m} g(x_i) \right)^2 = \frac{1}{n^2} \left(\sum_i g(x_i)^2 + \sum_{i \neq j} g(x_i) g(x_j) \right)$$

On average over minibatches, this yields:

$$\mathop{\mathbb{E}}_m \left[B^2\right] = \frac{1}{n^2} \mathop{\mathbb{E}}_m \left[\sum_i g(x_i)^2 \right] + \frac{1}{n^2} \mathop{\mathbb{E}}_m \left[\sum_{i \neq j} g(x_i) g(x_j) \right]$$

Useful properties of averages over minibatches. As minibatches are formed with samples randomly chosen, i.i.d. (that is, each sample is independently sampled from the other ones in the minibatch), $\mathbb{E}_m$ is actually $\mathbb{E}_{x_1 \sim \mathcal{S}} \mathbb{E}_{x_2 \sim \mathcal{S}} \ldots \mathbb{E}_{x_m \sim \mathcal{S}}$ where $\mathcal{S}$ is the sampling distribution of one point, and consequently formulas such as $\mathbb{E}_m \left[\sum_i f(x_i)\right]$ can be simplified as follows:

$$\mathop{\mathbb{E}}_m \left[\sum_i f(x_i) \right] = \sum_i \mathop{\mathbb{E}}_m \left[f(x_i) \right] = \sum_i \mathop{\mathbb{E}}_{x_i \sim \mathcal{S}} \left[f(x_i) \right] = n \mathop{\mathbb{E}}_{x \sim \mathcal{S}} \left[f(x) \right]$$

Similarily, formulas involving 2 variables symmetrically such as $\mathbb{E}_m \left[\sum_{i \neq j} f(x_i) f(x_j)\right]$ boil down as follows:

$$\mathop{\mathbb{E}}_{m}\left[\sum_{i \neq j} f(x_i) f(x_j)\right] = \sum_{i \neq j} \mathop{\mathbb{E}}_{m}\left[f(x_i) f(x_j)\right] = n(n-1) \mathop{\mathbb{E}}_{m}\left[f(x) f(x')\right]$$

$$= n(n-1) \mathop{\mathbb{E}}_{x \sim \mathcal{S}}\left[f(x)\right] \mathop{\mathbb{E}}_{x' \sim \mathcal{S}}\left[f(x')\right] = n(n-1) \left(\mathop{\mathbb{E}}_{x \sim \mathcal{S}}\left[f(x)\right]\right)^2$$

Back to our variance minimization. Our quantity of interest above thus becomes: $\mathbb{E}_m\left[B^2\right] = \frac{1}{n}\mathbb{E}_{x \sim \mathcal{S}}\left[g(x)^2\right] + \frac{n(n-1)}{n^2}\left(\mathbb{E}_{x \sim \mathcal{S}}\left[g(x)\right]\right)^2 = \frac{1}{n}\mathbb{E}\left[g(x)^2\right] + (1 - \frac{1}{n})A^2$ in our case where the sampling distribution is uniform.

Thus the approximation error is:

$$\mathop{\mathbb{E}}_{m}[(A-B)^2] = -\frac{1}{n}A^2 + \mathop{\mathbb{E}}_{m}\left[\frac{1}{n^2}\sum_i g(x_i)^2\right]$$

which we can estimate by sampling as:

$$\mathop{\mathbb{E}}_{m}[(A-B)^2] \approx -\frac{1}{n}A^2 + \frac{1}{n^2}\sum_i g(x_i)^2$$

using as many samples as possible (not the current minibatch considered at that gradient descent step, but a sliding average over past minibatches for instance). Using the current minibatch might lead to a biased estimator.

Let us develop the second term (the first one being constant). The target quantity A is the gradient of the optimization criterion, of the form

$$A = \int_X g = \int_X \frac{dq}{d\theta}(f + K)$$

thus

$$g(x_i) = \frac{dq(x_i)}{d\theta}(f(x_i) + K)$$

Note that g and K are vectors, but we can deal with each coordinate independently as they do not interact in these expressions. Let us focus on the j-th coordinate of g, i.e. the j-th parameter:

$$g_j(x_i) = \frac{dq(x_i)}{d\theta_j}(f(x_i) + K_j)$$

In order not to hamper the reading, we drop j in the next lines:

$$\sum_i g(x_i)^2 = \sum_i \frac{dq(x_i)}{d\theta}^2 (f(x_i)^2 + K^2 + 2Kf(x_i))$$

$$= K^2\left(\sum_i \frac{dq(x_i)}{d\theta}^2\right) + 2K\left(\sum_i \frac{dq(x_i)}{d\theta}^2 f(x_i)\right) + \left(\sum_i \frac{dq(x_i)}{d\theta}^2 f(x_i)^2\right)$$

Minimizing this w.r.t. K yields:

$$K = -\frac{\sum_i (\frac{dq(x_i)}{d\theta})^2 f(x_i)}{\sum_i (\frac{dq(x_i)}{d\theta})^2}$$

i.e.

$$K_j = -\frac{\sum_i (\frac{dq(x_i)}{d\theta_j})^2 f(x_i)}{\sum_i (\frac{dq(x_i)}{d\theta_j})^2}$$

This quantity can be efficiently computed in practice using libraries such as BackPACK[1] [27].

[1] `https://backpack.pt/`

J.6.3 Gradient estimation

Thus, given a minibatch, the estimation of a gradient A of the form $\int_X \frac{dq}{d\theta} f$, such as $\nabla_\theta KL(p||q_\theta) = -\int_X \frac{dq}{d\theta}\frac{p}{q}$, should rather be done with the formula:

$$A_j = \frac{1}{n}\sum_i \frac{dq(x_i)}{d\theta_j}\left(f(x_i) - \frac{\sum_k (\frac{dq(x_k)}{d\theta_j})^2 f(x_k)}{\sum_k (\frac{dq(x_k)}{d\theta_j})^2}\right) \in \mathbb{R}$$

$$= \frac{1}{n}\left(\sum_i \frac{dq(x_i)}{d\theta_j} f(x_i) - \left(\sum_i \frac{dq(x_i)}{d\theta_j}\right)\frac{\sum_k (\frac{dq(x_k)}{d\theta_j})^2 f(x_k)}{\sum_k (\frac{dq(x_k)}{d\theta_j})^2}\right)$$

The full gradient A with all coordinates is then estimated as:

$$\hat{A} = \frac{1}{n}\left(\sum_i \frac{dq}{d\theta} f - \left(\sum_i \frac{dq}{d\theta}\right)\frac{\sum_k (\frac{dq}{d\theta})^2 f}{\sum_k (\frac{dq}{d\theta})^2}\right)$$

using coefficient-wise squaring, multiplication and division between parameter-size vectors.

NB: as said above, the terms coming from K, i.e. the sums involving squares, should be computed with running means over minibatches, while the other sums are performed on the current minibatch.

Remember that $\sum_i \frac{dq}{d\theta}$ is a quantity which on average over possible minibatches is 0, but is not necessarily exactly 0 for any given minibatch. One could correct the deviation of $\sum_i \frac{dq}{d\theta}$ from 0 by removing the mean of $\frac{dq}{d\theta}$, which would lead to an expression of the form:

$$\frac{1}{n}\sum_i \left(\frac{dq}{d\theta} - \overline{\frac{dq}{d\theta}}\right) f = \frac{1}{n}\sum_i \frac{dq}{d\theta} f - \frac{1}{n^2}\left(\sum_i \frac{dq}{d\theta}\right)\left(\sum_i f\right)$$

but it turns out that with our stabilizing trick, a better correction can be found, by exploiting the correlation between f and $(\frac{dq}{d\theta})^2$.

Note also that it is not useful to use this trick on normalized discretized distribution optimization such as $KL(p^d||q^d)$. Indeed, $\sum_i \frac{dq^d}{d\theta}$ would be 0 always, so no correction would be brought.

J.6.4 Variance

The expected deviation between the true gradient A and the (optimized) minibatch estimation B is then (estimated over a minibatch):

$$\mathop{\mathbb{E}}_m[(A-B)^2] = -\frac{1}{n}A^2 + \frac{1}{n^2}\left[\left(\sum_i \frac{dq}{d\theta}^2 f^2\right) - \frac{\left(\sum_i (\frac{dq}{d\theta})^2 f\right)^2}{\sum_i (\frac{dq}{d\theta})^2}\right]$$

Note that the term between brackets is positive (or 0), according to Cauchy-Schwartz. It is 0 when (and only when) f is constant (in which case A is 0 also).
Note also that without optimization upon K, i.e. with $K = 0$, the negative term in the bracket disappears. Consequently, the variance reduction due to this stabilizing trick is $\frac{1}{n^2}\frac{\left(\sum_i (\frac{dq}{d\theta})^2 f\right)^2}{\sum_i (\frac{dq}{d\theta})^2}$.

J.7 Normalization mistakes are removed by the stabilizing trick (on average)

We show here that applying the stabilizing trick correctly (i.e. with running means to estimate K) does remove the normalization mistakes of the naive discretization. Indeed, on average, i.e. on expectation over minibatches, the total mass is preserved at sampled points, as follows.

Considering the mass (density) $q(x_i)$ at point x_i, the mass change during this time step, at point x_i, is $\frac{dq}{d\theta}(x_i) \cdot \varepsilon\, \delta\theta$ where ε is the learning rate and $\delta\theta$ is the parameter change, given by $\delta\theta = \sum_i \frac{dq}{d\theta} f + \sum_i \frac{dq}{d\theta} K$.

The global mass change over all sampled points is thus:

$$\varepsilon \left(\sum_i \frac{dq}{d\theta}(x_i) \right) \cdot \left(\sum_i \frac{dq}{d\theta} f + \sum_i \frac{dq}{d\theta} K \right) \quad \propto \quad \left(\sum_i \frac{dq}{d\theta} \right) \left(\sum_i \frac{dq}{d\theta} f \right) + \left(\sum_i \frac{dq}{d\theta} \right)^2 K$$

On average over minibatches, this becomes:

$$\mathbb{E}\left[\frac{dq}{d\theta}^2 f \right] + \mathbb{E}\left[\frac{dq}{d\theta}^2 \right] K$$

which can be enlightened by the value of K obtained in previous section: $K = -\frac{\mathbb{E}\left[\frac{dq}{d\theta}^2 f \right]}{\mathbb{E}\left[\frac{dq}{d\theta}^2 \right]}$. Consequently on average the mass is kept. This prevents the pathological dynamic behavior observed with naive discretization.