Data-free Generation of Molecular Configurations with Normalizing Flows

Loris Felardos

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Data-free Generation of Molecular Configurations with Normalizing Flows

Génération de configurations moléculaires avec des flux normalisants sans données

Présentée par :

Loris FELARDOS SAINT JEAN

Direction de thèse :

Bruno RAFFIN Directeur de recherche, INRIA, LIG, Université Grenoble Alpes	Directeur de thèse
Jérôme HÉNIN Chargé de Recherche, CNRS, IBPC, LBT	Co-directeur de thèse
Guillaume CHARPIAT Chargé de recherche, INRIA, LISN, Université Paris Saclay	Co-encadrant de thèse

Rapporteurs :

TONY LELIEVRE
Ingénieur HDR, École des Ponts ParisTech, CERMICS
ERIC VANDEN-EIJNDEN
Professeur, New York University, CIMS

Thèse soutenue publiquement le **2 décembre 2022**, devant le jury composé de :

TONY LELIEVRE Ingénieur HDR, École des Ponts ParisTech, CERMICS	Rapporteur
ERIC VANDEN-EIJNDEN Professeur, New York University, CIMS	Rapporteur
JEAN-PHILIP PIQUEMAL Professeur des Universités, Sorbonne Université, CNRS, LCT	Examinateur
ANDREW FERGUSON Professeur associé, University of Chicago	Examinateur
KIM THANG NGUYEN Professeur des Universités, LIG, Université Grenoble Alpes	Examinateur
BRUNO RAFFIN Directeur de recherche, INRIA, LIG, Université Grenoble Alpes	Directeur de thèse
JÉRÔME HÉNIN Chargé de recherche HDR, CNRS, IBPC, LBT	Co-directeur de thèse

Invités

GUILLAUME CHARPIAT Chargé de recherche, INRIA, LISN, Université Paris Saclay	Co-encadrant de thèse, Invité
DANILO REZENDE Senior Staff Research Scientist, Deepmind	Invité

Data-free Generation of Molecular Configurations with Normalizing Flows

Loris Felardos
Co-supervised by Guillaume Charpiat (INRIA, LISN, Université Paris-Saclay)
Co-supervised by Jérôme Hénin (CNRS, IBPC, LBT)
Directed by Bruno Raffin (INRIA, LIG, Université Grenoble Alpes)

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 of interest). However, current implementations rely on training data, which typically comes from computationally expensive simulations. There is therefore a clear incentive to train models in a *data-free* setting by only relying on the target density, which can be obtained from a physical energy model (up to a constant factor).

In this work, we start by analyzing the properties of the only data-free loss used in the literature and expose its limitations. It is based on a Kullback-Leibler divergence and shows a strong propensity for mode collapse during optimization on high-dimensional distributions. We then propose multiple guidelines to alleviate the issue and demonstrate the disproportionate impact that flat degrees of freedom in the target distribution may have on the quality of convergence. Another KL loss, which we make data-free, solves the collapse problem but is still brittle since it relies on numerically unstable importance sampling weights.

We then introduce a new loss function, well-grounded in theory and with suitable optimization properties (including a low computational cost and the absence of importance sampling weights). 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. This work is a fundamental step towards complete trainings that could be 100% data-free and we discuss the remaining conditions for how to achieve that.

Contents

0 Notation

General distributions and energies:

- $c = 3 \times N$: the total number of atomic coordinates of the molecule of interest (i.e. the dimensionality of the problem) with N the number of atoms.
- x: one molecular configuration, a vector of size c, with energy $U_B(x)$
- z: one embedding, a vector of size c, with energy $U_{\mathcal{N}}(z)$

Real distributions:

- $x_B \sim p_B$: one *real* molecular configuration (i.e. from the dataset)
  - $U_B(x)$ is the physically-grounded potential energy[1]
  - $p_B(x) = \frac{1}{\mathcal{Z}_B} \cdot e^{-\beta U_B(x)}$ is the Boltzmann distribution
  - $\tilde{p}_B = \mathcal{Z}_B p_B$ is the "unnormalized" Boltzmann distribution
  - $\mathcal{Z}_B = \int e^{-\beta U_B(x)} dx$ is the partition function
- $z_{\mathcal{N}} \sim q_{\mathcal{N}}$: one sample of the Gaussian
  - $q_{\mathcal{N}}(z) = \frac{1}{\mathcal{Z}_{\mathcal{N}}} \cdot e^{-\frac{1}{2\sigma^2} U_{\mathcal{N}}(z)} = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^c} \cdot e^{-\frac{1}{2\sigma^2} \sum_i^c z_i^2}$ is a c-dimensional multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ that is centered ($\boldsymbol{\mu} = \mathbf{0}$) and has a diagonal covariance matrix with equal entries ($\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2) = \sigma^2 \text{Id}$) usually with $\sigma = 1$ (see appendix A.1).
  - $\tilde{q}_{\mathcal{N}} = \mathcal{Z}_{\mathcal{N}} q_{\mathcal{N}}$ is the "unnormalized" Gaussian distribution
- Within plots, real data is colored in orange.

Generated distributions:

- $x_G = G(z_{\mathcal{N}}) \sim p_G$: one *generated* molecular configuration (with $G = F^{-1}$)
  - by definition, p_G is the pushforward measure of $q_{\mathcal{N}}$ by G
  - $G = F^{-1}$ is the model used during ***G****eneration*
  - $p_G(x) = p_G(G(z))$ is the generated x-distribution with: $p_G(G(z)) = q_{\mathcal{N}}(z) \cdot \left| \det \left(\frac{\partial G(z)}{\partial z} \right) \right|^{-1}$ by the change of variable formula
  - the energy of generation of x_G samples is denoted $U_G(x)$ and defined by $p_G(x) \coloneqq \frac{1}{\mathcal{Z}_G} \cdot e^{-U_G(x)}$ (to mimic the Boltzmann formula)
- $z_F = F(x_B) \sim q_F$: one *inferred* sample of the Gaussian (with $F = G^{-1}$)
  - by definition, q_F is the pushforward measure[2] of p_B by F
  - $F = G^{-1}$ is the model used during *in****F****erence*
  - $q_F(z) = q_F(F(x))$ is the inferred z-distribution with: $q_F(F(x)) = p_B(x) \cdot \left| \det \left(\frac{\partial F(x)}{\partial x} \right) \right|^{-1}$ by the change of variable formula
  - the energy of generation of z_F samples is denoted $U_F(z)$ and defined by $q_F(z) \coloneqq \frac{1}{\mathcal{Z}_F} \cdot e^{-U_F(z)}$ (to mimic the Gaussian formula)
- θ denotes the model parameters and is usually implicit: $G = G_\theta$ and $F = F_\theta$
- $J_G(z)$ denotes the Jacobian of the function G (i.e. the matrix of all its first-order partial derivatives at point z).

[1] Here, approximated by a molecular mechanics force field.

[2] Herein, we only describe regular measures that are associated with a density. When the context is clear, we sometimes refer to a measure using the corresponding density.

Batch Weights:

- $(\mathbf{x}_B \sim p_B^n) = \{x_{B,i} | x_{B,i} \sim p_B \text{ and } i \in [1, ..., n]\}$ is a mini-batch of x_B points
- $(\mathbf{z}_\mathcal{N} \sim q_\mathcal{N}^n) = \{z_{\mathcal{N},i} | z_{\mathcal{N},i} \sim q_\mathcal{N} \text{ and } i \in [1, ..., n]\}$ is a mini-batch of $z_\mathcal{N}$ points
- w_i: batch weight associated with element i of a mini-batch of size n

Summary Diagram:

$$\overbrace{z_\mathcal{N} \sim q_\mathcal{N}}^{\text{Gaussian distribution}} \xrightarrow{\boldsymbol{G}} \overbrace{x_G = G(z_\mathcal{N}) \sim p_G}^{\text{generated distribution}}$$

$$z_F = F(x_B) \sim q_F \xleftarrow{\boldsymbol{F}} \underbrace{x_B \sim p_B}_{\text{target distribution}}$$

Detach Operator:

- $[...]^\ddagger$: detach operator (acts as a no op during the forward pass but treats the contents of the brackets as a constant during back-propagation):

$$\begin{aligned}
\frac{\partial}{\partial x}\left[x \cdot \cos x\right] &= \cos x - x \cdot \sin x \\
\frac{\partial}{\partial x}\left[x \cdot [\cos x]^\ddagger\right] &= \cos x \\
\frac{\partial}{\partial x}\left[x^\ddagger \cdot \cos x\right] &= -x \cdot \sin x
\end{aligned}$$

Color Scheme:

- Simulated "p_B" data is colored in **orange**. This is the "real data" often used during model pre-trainings and/or to assess the quality of model generation.
- Generated "p_G" data sampled from pre-trained models is colored in **pink**.
- Generated "p_G" data sampled from models fine-tuned with variations of $\mathcal{L}_{KLx}$ or $\mathcal{L}_{RN_{1/2}}$ is colored in **blue**.
- Generated "p_G" data sampled from models fine-tuned with variations of $\mathcal{L}_{KLz}^{\text{dd}}$ is colored in **cyan**.
- Generated "p_G" data sampled from models fine-tuned with variations of $\mathcal{L}_{KLx} + \mathcal{L}_{KLz}^{\text{dd}}$ is colored in **purple**.
- Generated "p_G" data sampled from models fine-tuned with variations of $\mathcal{L}_{PL^2}$ is colored in **green**.
- Generated "p_G" data sampled from models fine-tuned with variations of $\mathcal{L}_{OT^\varepsilon}^{\text{dd}}$ is colored in **teal**.

Loss superscripts. If a loss does not have a superscript then it is data-free (e.g. $\mathcal{L}_{KLx}$ or $\mathcal{L}_{PL^2}$). To avoid confusion, some losses are labeled with the superscripts $^{\text{dd}}$ or $^{\text{df}}$ to denote data-dependent or data-free losses respectively (e.g. $\mathcal{L}_{KLz}^{\text{dd}}$ and $\mathcal{L}_{KLz}^{\text{df}}$)

Infoboxes:

"Infoboxes" like this one contain the most important observations and/or conclusions of this work.

1 Context: Molecules and Neural Networks

1.1 Boltzmann distribution and Molecular Dynamics

Being able to efficiently analyze the distributions of molecular configurations has been a long-standing objective of Statistical Mechanics. One successful method to estimate these distributions consists in reproducing the movements of individual atoms through Molecular Dynamics simulations. Given an initial 3D-configuration, one can numerically solve Newton's equations of motion to get an accurate approximation of the trajectory that the molecule will follow over some timescale. If the dynamics is sufficiently disordered, it is ergodic, which means that averages over the trajectory converge towards averages over the macroscopic distribution under the relevant conditions.

In an isolated system, the total energy (sum of the kinetic energy and potential energy U_B) is conserved, and the system tends to maximize its thermodynamic entropy. If the system is in contact with an external heat bath (a large system with thermal inertia), it exchanges energy until it reaches thermal equilibrium, which is a stable distribution of molecular configurations characterized by the temperature T of the heat bath. This distribution is called the *Boltzmann distribution* and is described by:

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

$$= \frac{1}{\mathcal{Z}_B} e^{-U_B(x)/k_B T} \tag{1b}$$

with:

- $p_B(x)$: the probability that the molecule is in a configuration x
- $U_B(x)$: the energy of the configuration x
- $\mathcal{Z}_B$: the partition function defined as $\mathcal{Z}_B = \int e^{-U_B(x)/k_B T} dx$
- β: the thermodynamic beta defined as $\beta = 1/k_B T$
- $k_B T$: the Boltzmann constant multiplied by the temperature

The potential energy term U_B can be estimated from particle interactions (cf. section 2.2 for how to achieve this), and its negative gradient $-\nabla U_B$ represents the forces that apply to all the particles of the system. Back to the dynamics perspective, Newton's second law states the following formula which implies conservation of energy:

$$M \odot \ddot{x} = -\nabla U_B(x) \tag{2}$$

with:

- M the masses of all the particles
- $\ddot{x}$ the acceleration of the particles
- $\odot$ an element-wise multiplication

This, however, represents an isolated system, not one at constant temperature, and does not produce the Boltzmann distribution. One also needs to model thermal coupling to an external heat bath. One way to represent such a "thermostat" is to introduce an implicit material present throughout the system, interacting with all particles though friction forces (modeled with Stokes' law) and random collisions. The full model, referred to as *Langevin dynamics* is described in the following equation:

$$M\ddot{x} = -\nabla U_B(x) - \gamma\dot{x} + \sqrt{2\gamma k_B T}R(t) \tag{3}$$

with:

- $\dot{x}$ the velocity of the particles
- γ the friction coefficient of the system (which multiplies the velocity $\dot{x}$ to form the friction force $-\gamma\dot{x}$)
- $R(t)$ a Gaussian process (with zero-mean and such that samples taken at two different time steps are completely uncorrelated with one another)

In this model, the temperature is given as a hyperparameter of the dynamics and can therefore be controlled like with a thermostat. At higher temperatures, the trajectory becomes noisier because of the relatively higher influence of $R(t)$. As a consequence the probability distribution over the possible configurations of the system becomes more uniform (as illustrated by the Boltzmann equation) and the entropy S_B rises:

$$S_B = S(p_B) = -k_B \int p_B(x) \log p_B(x)\ dx \tag{4}$$

1.2 Rationale behind using Neural Networks

The approximations made so far with the choice of the dynamics model and later of the energy function (see section 2) are sufficiently reliable to produce realistic trajectories when compared with quantum simulations operating at a much finer scale. In principle, molecular dynamics could even take advantage of considerable improvements in hardware and higher parallelism to predict protein structure by replicating the folding of polypeptide chains. In practice, such simulations run at a timescale in the order of the femtosecond (10^{-15}s) whereas slow protein movements can reach several minutes. As a result they remain notoriously slow and the field would be of limited practical utility without algorithmic improvements.

In addition to the high-dimensionality of the configuration space, another culprit behind the poor sampling efficiency is usually the presence of free energy barriers that separate the space into several more or less isolated regions. Various methods have been proposed over the years to address that problem [23]. For example, such methods may aim to:

- carefully balance the temperature of the simulation (e.g. Simulated annealing [31, 59], Parallel Tempering [16, 48, 49]),

- flatten the energy landscape along directions of interest (e.g. Umbrella Sampling [27, 53], Metadynamics [2, 14, 30], Adaptive Biasing Force (ABF) [8, 11, 22]),
- sample rare events more efficiently while keeping track of the relative probability of such events with importance sampling (e.g. Adaptive Multilevel Splitting method (AMS) [9, 10]),
- etc.

Those methods significantly improve the efficiency (and therefore the practical utility) of Molecular Dynamics but three significant problems remain:

- First, samples are highly correlated in time. In principle, one would want to sample configurations independently of previous ones.
- Second, analysis is difficult. Producing the correct distribution does not mean understanding it. A simple density estimation around a configuration for example can be too computationally expensive to do in practice.
- Third, there is no transfer of knowledge from one simulation to another. A new simulation needs to be run each time a new molecule is being studied.

The objective of this work is to address these problems by capturing the distribution of molecular configurations with an Artificial Neural Network. This would make the generation of samples completely uncorrelated, some new methods of analysis would become available (like precise density estimation) and this would open the door to transfer learning. The underlying strategy is to make use of existing energy calculators and carefully tuned force fields parameters from the literature to train the model without having to rely on lengthy simulations (i.e. without data).

1.3 Outline and contributions

During training, two fundamental quantities need to be computed: the potential energy U_B and the energy of generation U_G. In this work, the force-fields to compute U_B and the flow-based layers to compute U_G all come from the literature, as well as the losses $\mathcal{L}_{KLx}$ and $\mathcal{L}^{\text{dd}}_{KLz}$. The novel contributions of this work include:

- A principled clipping method of the individual terms of the potential energy U_B to avoid numerical instability issues. The associated code enables efficient force-field computations on GPU through batching with PyTorch (which is something that other frameworks like OpenMM do not support yet). See section 2.2.
- A simple method to deal with rotational and translational invariances in the target distribution without having to rely on data augmentation, internal coordinates nor special architectures. See section 2.2.
- A new proposition for how to remove the combinatorial complexity of the target distribution introduced by the presence of hydrogen atoms. See section 3.2.5 for the easiest approach and section 9.2 for the more general one.

- An analysis of the collapse behavior of $\mathcal{L}_{KLx}$ on multiple datasets and the circumstances that make the collapse more likely. See sections 4.2, 4.4 and 4.5.
- A modification of the data-dependent loss $\mathcal{L}^{\mathrm{dd}}_{KLz}$ into the data-free loss $\mathcal{L}^{\mathrm{df}}_{KLz}$ with importance sampling weights. This results in a clear improvement over $\mathcal{L}_{KLx}$ in training stability and resistance to mode collapse. See section 5.
- A new family of losses that improve upon $\mathcal{L}^{\mathrm{df}}_{KLz}$ qualitatively by relying on L^2 losses on the energy ratios of pairs of points. See section 6.
- A loss that specifically optimizes the quality of unbiased estimations of expections of interest in downstream applications. This loss is developed by using many conclusions from the previous sections, does not require unstable importance sampling weights and outperforms every other data-free loss in terms of quality of generation (even managing to retain a minor mode representing just about 1% of the target distribution of the configurations of dialanine). This is the most important contribution of this work. See section 7.
- An investigation of yet another data-free loss based on Optimal Transport. See section 8.
- A description of how to make flow-based models hierarchical, even when the topological structure of the target distribution is irregular or changing between samples. See section 9.2.
- A presentation of how to scale the current approach in the future with curriculum learning in order to achieve data-free trainings of a larger scale, that would not rely on any pre-training steps. This approche would leverage transfer learning by using a single model for multiple molecules. See section 9.

2 Preparing U_B, the molecular energy

2.1 A differentiable molecular energy

Most loss functions used in the following sections rely on optimizing $p_B = \frac{e^{-\beta U_B}}{\mathcal{Z}_B}$ and $p_G = \frac{e^{-U_G}}{\mathcal{Z}_G}$.[3] Since $\mathcal{Z}_B$ is generally unknown, but also for numerical stability, it is usually U_B and U_G that are used instead of p_B and p_G. This section analyzes how to compute U_B in a differentiable and numerically stable manner. The next section (section 3) focuses on U_G.

The simplest method to compute the energy of a given molecular configuration is to use the OpenMM[4] python library. An energy computed in this fashion is not differentiable within existing Deep Learning frameworks (i.e. PyTorch[5] or TensorFlow[6]) and is therefore not optimizable at first sight. But since the forces applied to the individual atoms are the negative of the energy gradient (e.g. $-\nabla U_B(x)$) one can use the forces provided by OpenMM directly and propagate them backward to get the same result.

In practice, using this approach during training introduces two difficulties. First, it makes it almost impossible to modify the energy function (which is essential both for numerical stability and debug, as discussed below). And second, since OpenMM does not support batch computations on GPU[7], all the molecular configurations x_G of each mini-batch need to be loaded *sequentially*, which causes a considerable overhead.

The solution is to reimplement the energy calculator in PyTorch by relying on the same two parameter files as OpenMM:

- A `.psf` (aka. the "structure" file) which is specific to each molecule and holds lists of every bond, angle, dihedral and improper torsion as well as information needed to generate the hydrogen bonds and the non-bonded list.
- And a `.prm` (aka. the "parameter" file) which is provided by CHARMM[8] (another molecular simulation program) and holds the set of force field parameters used to determine the structural potential energy (more details on force fields are provided in section 2.2).

When comparing the individual energy terms computed with PyTorch with those provided by OpenMM, the resulting worst absolute difference is on the order of 10^{-5}

[3] Note that here there is no factor multiplying U_G that would be controlling the temperature of the p_G distribution. This is because U_G is assumed to be a multiplicative function of $z_{\mathcal{N}}$ which already has such a temperature parameter in the form of its standard deviation.

[4] `https://openmm.org/`

[5] `https://pytorch.org/`

[6] `https://www.tensorflow.org/`

[7] `https://github.com/openmm/openmm/issues/1995`

[8] `https://www.charmm.org/`

with `float32` precision which is negligible and thus demonstrates the validity of the implementation.[9] Note that an even lower absolute difference could be achieved with `float64` precision.

As a consequence of using PyTorch directly, the measured speedup lies between a factor 5 and 200 depending on the size of the batch. But most importantly it is now possible to differentiate each term of the energy separately (which is essential for diagnostic) and to modify them freely (which is critical for numerical stability as discussed in section 2.3).

2.2 Terms of the energy function

Interatomic interactions, which are quantum mechanical in nature, do not have simple analytical functional forms but can be estimated in Molecular Mechanics with *force fields*. Those are functions, associated with sets of parameters, used to compute potential energy terms. The energy function $U_B(x)$ of a given configuration x can be computed as a sum of several such terms [36]:

$$\begin{aligned} U_B(x) = U_{\text{bonds}}(b) \\ &+ U_{\text{angles}}(\theta) + U_{\text{urey-bradley}}(u) \\ &+ U_{\text{dihedrals}}(\phi) + U_{\text{impropers}}(\omega) \\ &+ U_{\text{nb}}(r) + U_{\text{coulomb}}(r) \end{aligned} \tag{5}$$

with b, θ, u, ϕ, ω and r being functions of x corresponding to bonds, angles, etc.

More specifically:

- $U_{\text{bonds}}(b) = \sum_{\text{bonds}} k_b(b-b_0)^2$ accounts for bond stretches, where k_b is the bond force constant and $b - b_0$ is the distance from equilibrium.
- $U_{\text{angles}}(\theta) = \sum_{\text{angles}} k_\theta(\theta-\theta_0)^2$ accounts for angles between bonds, where k_θ is the angle force constant and $\theta - \theta_0$ is the angle distance from equilibrium between 3 bonded atoms.
- $U_{\text{urey-bradley}}(u) = \sum_{\text{ub}} k_u(u-u_0)^2$ accounts for angle bending using non-bonded 1-3 interactions, where k_u is the force constant and u the distance between atoms 1-3.
- $U_{\text{dihedrals}}(\phi) = \sum_{\text{dihedrals}} k_\phi\big[1+\cos(m\phi+\delta)\big]$ accounts for dihedrals (e.g. torsion angles), where k_ϕ is the force constant, m the multiplicity, ϕ the dihedral angle and δ the phase shift.
- $U_{\text{impropers}}(\omega) = \sum_{\text{impropers}} k_\omega(\omega-\omega_0)^2$ accounts for impropers (e.g. out of plane bending), where k_ω is the force constant and $\omega-\omega_0$ is the out of plane angle.

[9] The worst absolute difference observed on the gradients of the energy (i.e. the forces) is of the same order of magnitude.

- $U_{\text{nb}}(r) = \sum_{\text{nb}} \epsilon \left[\left(\frac{r_{\min}}{r} \right)^{12} - \left(\frac{r_{\min}}{r} \right)^{6} \right]$ accounts for nonbonded Van der Waals forces between atom pairs separated by at least three bonds, where ϵ is the Lennard-Jones well-depth and $r_{\min}$ is the distance at the Lennard-Jones minimum (see figure 1 for a chart of the Lennard-Jones potential).
- $U_{\text{coulomb}}(r) = \sum_{\text{coulomb}} \frac{q_i q_j}{\varepsilon_r r_{ij}}$ accounts for Coulomb potentials where q_i is the partial atomic charge, ε_r is the effective dielectric constant (around 78.5 for water at 25°C), and r_{ij} is the distance between atoms i and j.

Note also that the values of the parameters depend on the type of the interaction. For example, the bond equilibrium value b_0 may change depending on the bond type.

2.3 Numerical stability of the energy gradient

The description of the individual terms of the energy sheds light on an important problem in the computation of $U_B(x_G)$: numerical instability. Small deviations of Cartesian coordinates can increase the energy by many times $k_B T$, and more critically, its associated gradient passed on to the model. The problem is especially pronounced at initialization when the outputs of the generator are close to random, to the point that it makes training effectively impossible.

Three types of terms in particular are responsible for this:

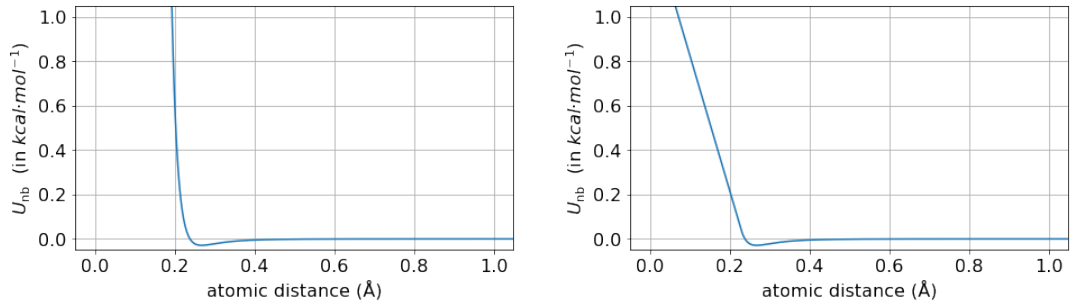
- $U_{\text{nb}}(x)$ because of its Lennard-Jones potential which requires a computation in the order of r^{-12} and thus blows up quickly when r goes to 0.
- $U_{\text{coulomb}}(x)$ which suffers from the same problem with r^{-1}.
- $U_{\text{bonds}}(x)$ and other harmonic potentials which grow quadratically with the deviation from the equilibrium.

The first idea that may come to mind to make the training more stable is to clip the gradient norm of the parameters of the model (or restrict it by using a log as in Noé et al. [39]). However this would require the manual tuning of the clipping value which may change with the model architecture. A better method consists in clipping *each term individually*. Not only this method is much more robust to the choice of the clipping value but it also provides a clear physical meaning to the operation, which is to soften the "hard" terms of the energy function. It also allows for clipping one specific term while preserving the information coming from all the other terms, which would be squashed by global clipping.

Two options remain. First, the value of the energy terms could be left unchanged and the gradients clipped only during the backward pass, or alternatively the function *itself* could be modified to become k-Lipschitz (with k the clipping value). The first method is slightly faster and simpler to implement, the second gives a better intuition of how much the clipping operation changes the energy function that the network tries to satisfy.

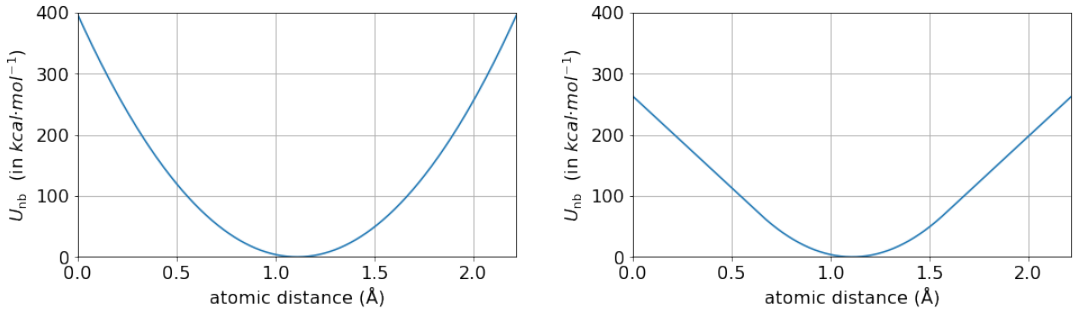
Focusing on the second method, the strategy consists in using affine functions to replace the sections of the potential where the derivative explodes in order to make the norm of the gradient bounded. For example, the clipping of the gradient of the Lennard-Jones potential is implemented as follows:

- The threshold value of r_{th} at which the gradient exceeds u'_{th} (e.g. the clipping value) is estimated. Given that $U'_{\text{nb}}(x) \approx -\frac{12\epsilon r_{\text{min}}^{12}}{r^{13}}$ when r is small, the clipping value verifies $|u'_{\text{th}}| \approx \frac{12\epsilon r_{\text{min}}^{12}}{r_{\text{th}}^{13}}$ and therefore $r_{\text{th}} \approx \left[\frac{12\epsilon r_{\text{min}}^{12}}{|u'_{\text{th}}|}\right]^{1/13}$.
- The linear approximation L_{nb} of U_{nb} at r_{th} is used to replace the Lennard-Jones potential for small r.
- The final function becomes: $\tilde{U}_{\text{nb}}(r) = \begin{cases} L_{\text{nb}}(r), & \text{if } r < r_{\text{th}} \\ U_{\text{nb}}(r), & \text{if } r \geqslant r_{\text{th}} \end{cases}$



(a) Before clipping. (b) After clipping

Figure 1: Lennard-Jones potential



(a) Before clipping. (b) After clipping

Figure 2: Harmonic bond potential

As a result, the Lennard-Jones potential is modified as in figure 1 (which uses the distance between two hydrogen atoms as an example), and the harmonic potentials are similarly transformed as in figure 2 (which uses a bond between a carbon and a hydrogen). Note that unrealistic clipping values have been chosen for better visualization.[10][11]

[10]Losses where U_B is not differentiated will be less sensitive to the gradient, but will benefit from the modified Lennard-Jones potential being bounded, as clashes between atoms are the type of singularity most likely to cause numerical issues.

[11]The clipped version of the energy function is used in every experiment performed in this work. But note that it is possible to reweight generated samples to compute expectations according to a different distribution if its energy function is known (see equation 9). So models trained with the

2.4 Handling rotations and translations

The potential energy $U_B(x)$ of a molecule x is both translationally and rotationally invariant. Although it makes perfect physical sense, this property can be a significant issue when it comes to generating molecular configurations.[12] Such invariances introduce degrees of freedom that unnecessarily complicate the task that the generative model has to solve.

There are several ways to deal with this issue:

- **Data Augmentation**. In this case, when a dataset is provided (usually during pre-training), the data needs to cover all possible rotations, for example by applying a different random rotation matrix to each data point sampled from the dataset. This approach is problematic for several reasons. It cannot handle translations on its own since an infinite amount of translations are possible[13], it is typically slower to learn since the target space is larger, and it makes fast density estimation (a fundamental feature of Normalizing Flows) impossible since it requires integration over all possible rotations.
- **Internal Coordinates**. Here, when encoding the data, the Cartesian coordinates are discarded in favor of internal coordinates. The obvious benefit is the complete removal of the redundant degrees of freedom. But in practice, internal coordinates are defined on specific domains: bond lengths are positive, angles live in $[0, \pi]$ and torsions are periodic in $[-\pi, \pi]$. These constraints are difficult to satisfy with flow-based models [29] (as used in this work) which need to be invertible (see section 3). Another issue is that small placement errors in the backbone of a protein may result in instabilities and even collisions between sections of the molecule that are far apart. Lastly, it is unclear how to design a single model that could be applied to different molecules in internal coordinates, whereas many equivariant architectures can do this trivially in Cartesian coordinates (see section 9).
- **Equivariant Architectures**. This method relies on models that are made translational and rotational equivariant by construction.[14] Such architectures include specific types of Graph Neural Networks [4, 44, 45] and can also be flow-based [28, 37, 43]. See section 9.
- **Translational and Rotational Penalties**. This is the simplest method, which is based on an additional term to the energy function that penalizes the rotations and translations of both simulated and generated configurations. The only requirement is a single configuration example (usually at an

clipped energy function are still useful to compute valuable quantities that depend on the orginal (non-clipped) energy function.

[12] One reason for that is described in section 5.2

[13] The model would actuallly learn the distribution of translations used in the data augmentation, which can only be different from the true uniform distribution over the whole non-bonded space.

[14] Note that the term "invariant" is used when the output *does not change* when the input changes, but "equivariant" means that it *does change in equivalent proportions* (for example, a small input translation leading to a small output translation).

energy minimum) to use as reference during alignments.[15] Unlike the three points above, this approach is a novel contribution.

Translational penalties (denoted $U_{\text{tran}}(x)$) are easy to implement by simply using some distance function d_{tran} between the generated molecule and its centered counterpart:

$$U_{\text{tran}}(x) = d_{\text{tran}}\left(x - x_{\text{centered}}\right) \tag{6}$$

Rotational penalties (denoted $U_{\text{rot}}(x)$) are quite similar in the sense that some distance function d_{rot} is used between the generated molecule and its rotated counterpart. But in this case several distances are possible, leading to different gradients:

$$U_{\text{rot}}(x) = d_{\text{rot}}\left(x - x_{\text{rotated}}\right) \tag{7}$$

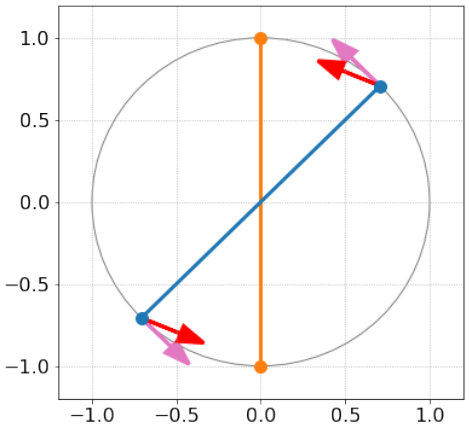


Figure 3: In orange: the target position. In blue: the current position. In red: a gradient that follows a simple euclidean interpolation. In purple: a gradient that follows a circular interpolation.

In figure 3, two possible gradients are represented that aim to rotate the blue segment on top of the orange one. The red gradient follows the shortest distance at the price of distorting the segment (essentially shortening it in this case), whereas the purple gradient follows the tangent of the circle, so as to leave the length of the blue segment unchanged during the transformation.

In this work, both translational and rotational penalties are combined into a single term which measures an L^2 distance between the generated configuration and a better aligned counterpart x_{aligned} (both translationally and rotationally).[16][17] Note that the x_{aligned} target may not be perfectly aligned since the magnitude of the maximum possible rotation with respect to x is capped at $\pi/3$ to avoid excessive distortions:

$$U_{\text{align}}(x) = d_{\text{align}}\left(x - x_{\text{aligned}}\right) = \lambda_{\text{align}} \cdot \left(x - x_{\text{aligned}}\right)^2 \tag{8}$$

with λ_{align} being a factor determining the balance between U_{align} and the rest of the terms of the energy U_B.[18]

[15] If such an example is unavailable, it may be best to let the model decide (for example by choosing the mode of a generated mini-batch).

[16] Essentially choosing the red arrow over the purple one in figure 3.

[17] Alignment is performed with `scipy`'s `align_vectors()` which is based on the Kabsch algorithm.

[18] In this work, a value of $\lambda_{\text{align}} = 10$ is used in every `Butane` and `Dialanine` experiment.

3 Preparing U_G, the energy of generation

3.1 Explicit p_G calculation for statistical physics applications

Now that the question of how to compute U_B in a differentiable manner is out of the way, the focus of this section is on how to do the same with U_G.

The task is to capture the target Boltzmann distribution p_B by training a generative model G with parameters θ. For a given molecule, a vector of Gaussian noise $z_{\mathcal{N}} \sim q_{\mathcal{N}} = \mathcal{N}(0, \mathrm{Id})$ is given to the generator which produces an output configuration $x_G = G(z_{\mathcal{N}}) \sim p_G \in \mathbb{R}^c$ of dimension $c = 3 \times N$ (e.g. the number of 3D-coordinates for all the atoms of the molecule).

$$\begin{aligned} z_{\mathcal{N}} \sim q_{\mathcal{N}} &\xrightarrow{\boldsymbol{G}} x_G = G(z_{\mathcal{N}}) \sim p_G \\ z_F = F(x_B) \sim q_F &\xleftarrow{\boldsymbol{F}} x_B \sim p_B \end{aligned}$$

If p_G can be computed, a very powerful feature becomes available: the unbiased estimation of expectations with respect to p_B, provided that the reweighting via $\frac{\tilde{p}_B}{p_G}$ remains reasonable (i.e. p_G must not be too close to zero where p_B is not):

$$\mathop{\mathbb{E}}_{x \sim p_B}[f(x)] = \frac{\mathop{\mathbb{E}}_{x \sim p_G}\left[\frac{\tilde{p}_B(x)}{p_G(x)} f(x)\right]}{\mathop{\mathbb{E}}_{x \sim p_G}\left[\frac{\tilde{p}_B(x)}{p_G(x)}\right]} \tag{9}$$

for any function f.

In addition, being able to compute p_G explicitly is very precious in the design of loss functions. An example of a frequently occurring term (see Section 4.1) involving p_G is the entropy[19] S_G:

$$S_G = -\int p_G(x) \cdot \log p_G(x)\, dx \tag{10a}$$

$$= -\int q_{\mathcal{N}}(z) \cdot \log \left[q_{\mathcal{N}}(z) \cdot \left| \det \left(\frac{\partial G(z)}{\partial z} \right) \right|^{-1} \right] dz \tag{10b}$$

$$= -\int q_{\mathcal{N}}(z) \cdot \log q_{\mathcal{N}}(z)\, dz + \int q_{\mathcal{N}}(z) \cdot \log \left| \det \left(\frac{\partial G(z)}{\partial z} \right) \right| dz \tag{10c}$$

$$= S_{\mathcal{N}} + \int q_{\mathcal{N}}(z) \cdot \log \left| \det \left(\frac{\partial G(z)}{\partial z} \right) \right| dz \tag{10d}$$

$$= S_{\mathcal{N}} + \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\log \left| \det \left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}} \right) \right| \right] \tag{10e}$$

[19] Note that equation 10e shows that the mean log Jacobian determinant measures the change of entropy introduced by the mapping into its output distribution, with respect to that of the input distribution.

with:

- (10a) note that S_G is the almost simplest integral that involves U_G and a p_G weight (used for sampling later) since $S_G = \int p_G(x)U_G(x)dx + \log \mathcal{Z}_G$
- (10b) by substitution of $p_G(x)$ by the change of variable formula:

$$\begin{aligned} p_G(x)\ dx &= q_{\mathcal{N}}(G^{-1}(x)) \cdot \left|\det\left(\frac{\partial G^{-1}(x)}{\partial x}\right)\right|\ dx \\ &= q_{\mathcal{N}}(z) \cdot \left|\det\left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}\ dx \\ &= q_{\mathcal{N}}(z)\ dz \end{aligned} \tag{11}$$

- (10c) note that this step may not be possible if we are not using S_G but some other, more general, integral of the form $\int p_G(x)f(U_G(x))dx$ with f a non-linear function.
- (10d) by definition of the entropy: $S_{\mathcal{N}} = S(q_{\mathcal{N}}) = -\int q_{\mathcal{N}}(x) \log q_{\mathcal{N}}(x)dx$
- (10e) by definition of expectations $\mathbb{E}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}}\left[f(z_{\mathcal{N}})\right] = \int q_{\mathcal{N}}(z)f(z)dz$

Having to compute p_G introduces two requirements over the model:

1. The model must be easily invertible for fast exact density estimation (but also for the computation of some useful loss terms described later).
2. The architecture must make the computation of the log of the Jacobian determinant (i.e. $\log|\det J|$) efficient.

Flow-based generative models (often just called Normalizing Flows) satisfy both requirements.[20] See Tabak and Vanden-Eijnden [51] as well as Papamakarios et al. [40] for an overview.

3.2 Simplest Normalizing Flow layers

Many layers have recently been developed that satisfy the requirements of Normalizing Flows, of which just a few are described below. To stay consistent with the previous notation z denotes the input of a layer and x its output. In this section, $J^{(l)}$ is used to denote the Jacobian of a layer l. If a layer is invertible, then a stack of such layers is invertible too, and if one can compute the Jacobian determinant of each layer, then the Jacobian determinant of the stack can be computed as well:

$$\log|\det J| = \sum_l \log\left|\det J^{(l)}\right| \tag{12}$$

3.2.1 Addition of a bias vector:

The addition of a bias vector b is an operation that is easily invertible (by simply subtracting b) and has a known Jacobian determinant of 1 (and therefore a log

[20] Note that the term "Normalizing Flow" refers to the *statistical method* leveraging the change of variable formula from equation 11, but may also refer to the associated Flow-based generative *models* as it is the case here.

Jacobian determinant of 0).

$$\begin{aligned} \text{Forward pass:} \quad & x = z + b \\ \text{Inverse pass:} \quad & z = x - b \\ \text{Jacobian Determinant:} \quad & \log\left|\det J^{(l)}\right| = 0 \end{aligned}$$

3.2.2 Multiplication by a diagonal matrix D:

Multiplication by a diagonal matrix D is equivalent to element-wise multiplication by its diagonal d. The inverse operation is an element-wise division by d which exists if and only if all the entries of d are different from 0. To avoid a possible numerical instability during training, every entry d_i, and its inverse $1/d_i$, must be constrained to stay away from 0. Although many strategies are possible, one convenient way to do this is to define d_i such that $d_i = e^{\tanh(\tilde{d}_i)}$ with $\tilde{d}_i$ a trainable vector in $\mathbb{R}$.

$$\begin{aligned} \text{Forward pass:} \quad & x = Dz \\ \text{Inverse pass:} \quad & z = D^{-1}x \\ \text{Jacobian Determinant:} \quad & \log\left|\det J^{(l)}\right| = \mathrm{Tr}(\log|D|) \end{aligned}$$

3.2.3 Multiplication by an orthogonal matrix:

Orthogonal matrices have a determinant of 1 but are not easily invertible unless parameterized as a composition of Householder Reflections $R = \mathrm{Id} - 2\frac{vv^\intercal}{v^\intercal v}$ with v a non-null vector (chosen to be trainable within models). Householder Matrices are their own inverses and any orthogonal matrix of size $n \times n$ can be constructed as a product of at most n such reflections. If a single reflection is used, this gives:

$$\begin{aligned} \text{Forward pass:} \quad & x = \left(\mathrm{Id} - 2\frac{vv^\intercal}{v^\intercal v}\right)z \\ \text{Inverse pass:} \quad & z = \left(\mathrm{Id} - 2\frac{vv^\intercal}{v^\intercal v}\right)x \\ \text{Jacobian Determinant:} \quad & \log\left|\det J^{(l)}\right| = 0 \end{aligned}$$

3.2.4 Coupling Blocks based on products and sums:

A more expressive alternative is to use so-called "Coupling Blocks" [15] defined as an invertible element-wise transformation of half the features x_a, parameterized by a function M of the other half of the features x_b. In case the element-wise transformation is a product, it is important to make sure that the outputs of M stay away from 0. This is usually done by defining M as $M(x_b) = e^{\tanh(\tilde{M}(x_b))}$ with $\tilde{M}$ an arbitrary differentiable function.

$$\begin{aligned} \text{Forward pass:} \quad & \begin{cases} x_a = z_a \odot M(z_b) \\ x_b = z_b \end{cases} \\ \text{Inverse pass:} \quad & \begin{cases} z_a = x_a \oslash M(x_b) \\ z_b = x_b \end{cases} \\ \text{Jacobian Determinant:} \quad & \log\left|\det J^{(l)}(z)\right| = \mathrm{Tr}(\log|M(z_b)|) \end{aligned}$$

with $\odot$ denoting element-wise multiplication and $\oslash$ element-wise division. The element-wise transformation can also be a sum (sometimes denoted with the operators $\oplus$ for the forward pass and $\ominus$ for the inverse pass).

The high expressivity of Coupling Blocks [33] largely comes from the internal arbitrary function M, but also on the fact that J is a direct function of z (it does not act as a trained constant as in the previous subsections).

3.2.5 More general Normalizing Flow layers

The list above is only a very small sample of all the flow-based layers that have been developed in recent years:

- Kingma and Dhariwal [25] use an LU decomposition to parameterize a weight matrix $W = PL(U + \text{diag}(s))$ with P a random permutation matrix, L a lower-triangular matrix, U an upper-triangular matrix and s a simple vector. In this formulation only L, U and s are trained, while P is chosen at random once and remains fixed afterwards.
- Krämer et al. [26] discuss a method to optimize invertible matrices via low-rank updates by keeping track of the matrix inverse and its determinant during the optimisation.
- Wu et al. [58] introduce Stochastic Flows which consist in using stochastic MCMC sampling blocks as a stochastic layer.
- Behrmann et al. [5] and Chen et al. [7] propose Residual Flows based on residual layers whose inverse and Jacobian can be computated with iterative estimators.
- Huang et al. [21] use Coupling Blocks with additional Gaussian degrees of freedom to increase the dimensionality of the layers at the price of requiring an integration during density estimation.

3.3 Dealing with hydrogen permutations

One difficulty, unique to the task of generating molecular configurations, is the presence of many hydrogen atoms that are of little interest for most downstream applications. They often represent a large fraction of all the atoms to place in 3D space, and needlessly increase the dimensionality of the problem. They can also be permuted within CH_2 and CH_3 groups, multiplying the number of modes by 2 or 3 each time. Overcoming the complexity induced by hydrogen permutations is sometimes the difference between a training that converges and one that does not.[21]

The simplest way to address this problem is to avoid generating hydrogen atoms altogether. But since the energy of a molecule can only be computed (at the desired level of precision using current force fields) if the coordinates of all the atoms are known, hydrogen atoms must still be placed *after* the model has generated the heavy atoms. There are several possibilities here:

[21] A common theme of the following sections is that simplifying the task that the model has to solve can often go a long way toward better quality of generation.

1. The hydrogen atoms could be placed according to some custom-made algorithm, but this solution is often tedious and prone to errors.
2. They could be placed approximately and their position could then be optimized by gradient descent, but this solution, although more robust, is quite compute intensive since it requires iteratively minimizing the energy of the hydrogen atoms both during and after training.
3. The best method would probably consist in using a two-stage Normalizing Flow, with the second stage being conditional on the output of the first one. The first stage would output the coordinates of the heavy atoms and the second the coordinates of the hydrogen atoms. Crucially, it does not matter if the second stage collapses to only one mode, since they are all symmetrical. This is an approach that is explored in section 9.2.
4. A simpler solution which decreases the dimensionality of the problem is to make the second stage deterministic, by using a feed-forward model. This is the method chosen in the remainder of this work.

The full model is therefore composed of two stages:

- The first stage (denoted G^{C} in the representation below) is a function of $z_{\mathcal{N}}$ and its role is to generate the 3D coordinates of the heavy atoms. It is a flow-based model which is therefore bijective and preserves the dimensionality of the input. As a consequence, the dimensionality of $z_{\mathcal{N}}$ is chosen to be equal to the number of coordinates of the heavy atoms x_G.
- The second stage (denoted G^{H}) takes the output x_G^{C} of G^{C} and places the hydrogen atoms close to their minimum of energy. The reverse operation F^{H} removes hydrogen atoms from its input (typically x_B^{all}). G^{H} is *not* bijective since the dimensionality of its output x_G^{all} is equal to the *total* number of coordinates of the molecule (including the hydrogen atoms) which is strictly larger than that of x_G as soon as there is a single hydrogen atom in the molecule.

$$\begin{array}{rcccl} z_{\mathcal{N}} \sim q_{\mathcal{N}} & \xrightarrow{\boldsymbol{G^{\mathrm{C}}}} & x_G^{\mathrm{C}} = G^{\mathrm{C}}(z_{\mathcal{N}}) \sim p_G^{\mathrm{C}} & \xrightarrow{\boldsymbol{G^{\mathrm{H}}}} & x_G^{\mathrm{all}} = G^{\mathrm{H}}(x_G) \sim p_G^{\mathrm{all}} \\ z_F = F(x_B) \sim q_F & \xleftarrow[\boldsymbol{F^{\mathrm{C}}}]{} & x_B^{\mathrm{C}} = F^{\mathrm{H}}(x_B^{\mathrm{all}}) \sim p_B^{\mathrm{C}} & \xleftarrow[\boldsymbol{F^{\mathrm{H}}}]{} & x_B^{\mathrm{all}} \sim p_B^{\mathrm{all}} \end{array}$$

with:

- $x_G^{\mathrm{all}} \in \mathbb{R}^{3N}$ with N the total number of atoms
- $x_G^{\mathrm{C}} \in \mathbb{R}^{3N_{\mathrm{C}}}$ with N_{C} the total number of heavy atoms
- $x_G^{\mathrm{H}} \in \mathbb{R}^{3N_{\mathrm{H}}}$ with N_{H} the total number of hydrogen atoms

In summary, all-atom coordinates are generated as $x_G^{\mathrm{all}} = \{x_G^{\mathrm{C}}, x_G^{\mathrm{H}}\} = G^{\mathrm{H}}(G^{\mathrm{C}}(z_{\mathcal{N}}))$, and the reverse operation is $z_F = F^{\mathrm{C}}(F^{\mathrm{H}}(x^{\mathrm{C}}, x^{\mathrm{H}}))$. As a result, whereas the first stage G^{C} is bijective, the complete model $G^{\mathrm{H}} \circ G^{\mathrm{C}}$ is not. While $F^{\mathrm{C}} \circ F^{\mathrm{H}} \circ G^{\mathrm{H}} \circ G^{\mathrm{C}}$ is identity in the latent space, $G^{\mathrm{H}} \circ G^{\mathrm{C}} \circ F^{\mathrm{C}} \circ F^{\mathrm{H}} = G^{\mathrm{H}} \circ F^{\mathrm{H}}$ 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 remainder of this work, to simplify the notation on molecular tasks:
- G refers to the flow-based generator G^{C} (and $F = F^{\mathrm{C}}$ to its inverse).
- $p_G = p_G^{\mathrm{C}}$ refers to the probability of generation of the heavy atoms.
- $p_B = p_B^{\mathrm{C}}$ refers to the target distribution of G but not necessarily to the Boltzmann distribution of the full molecule (noted p_B^{all} when the distinction is necessary).
- The energy function U_B from section 2 is implicitly taken to mean $U_B^{\mathrm{all}} \circ G^{\mathrm{H}}$ to accommodate for the change.

See appendix A.2 for more details on how the ratio between modes changes when placing the hydrogen atoms deterministically near their energy minimum, and how the new ratio can be estimated in practice.

3.4 Building the complete flow-based model

In the remainder of this work, only two model architectures are used: one for `Double Well` datasets (presented in section 4.4), and one for molecular datasets like `Butane` and `Dialanine` (both presented in section 4.2).

The architecture used in `Double Well` experiments is a simple stack of 8×4 Coupling Blocks (as described in section 3.2.4) corresponding to 8 times the following substack:

- A multiplicative Coupling Block modifying the first half of the features (this would be denoted $x_a = z_a \odot M(z_b)$ in section 3.2.4),
- followed by an additive Coupling Block also modifying the first half of the features ($x_a = z_a \oplus M(z_b)$),
- followed by a multiplicative Coupling Block modifying the second half of the features ($x_b = z_b \odot M(z_a)$),
- followed by an additive Coupling Block again modifying the second half of the features ($x_b = z_b \oplus M(z_a)$).

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 [3].

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 (referred to as G^{H} in section 3.3). It has 3 fully-connected linear layers separated by CELUs and a hidden size of 512.

4 Using $KL(p_G||p_B)$ as a training objective

Formally, the training objective can be defined as finding the optimal parameter vector θ^* such that the divergence D between the two distributions p_G and p_B is minimized. As a reminder, p_G is the distribution that is *optimized* and is therefore a function of θ (i.e. $p_G = p_{G_\theta}$).

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, D(p_G||p_B) \tag{13}$$

Any divergence D measures how far apart two probability distributions are, and has two important properties[22]:

1. $D(p_G||p_B) \geqslant 0$
2. $D(p_G||p_B) = 0$ if and only if $p_G = p_B$

for all p_G, p_B with common support.

The fact that p_G and p_B are probability distributions implies that they are positive and sum to one, i.e. $p_G \geqslant 0$ and $\int p_G = 1$. In practice though, $\mathcal{Z}_B$ is typically not known. Only $\tilde{p}_B$ is known, which does *not* sum to 1. This can be a tricky problem in the design of a sensible training objective.

An important consequence of choosing D appropriately is that the optimization behavior may change drastically depending on which divergence D is chosen. One could assume that the choice of D is not too critical as long as θ^* is reached sufficiently quickly and reliably. Indeed, if the parametric function used to produce p_G is fully expressive (in the sense that it is able to model p_B *perfectly*), then p_G^* should be the same regardless of which divergence D is used.[23] But crucially, p_G^* may *also* depend on D if p_G cannot model p_B perfectly. Said differently, for a model that lacks expressivity (or faces optimization issues), the best solution depends on what is being measured.

4.1 Expression of the loss function

KL divergences, are typically not symmetrical (i.e. $KL(p_G||p_B) \neq KL(p_B||p_G)$). The optimization behavior resulting from choosing $D = KL(p_G||p_B)$ is analyzed in this section (the other KL divergence and other choices are analyzed in the following sections).

[22] Note that these two properties are necessary but not sufficient to define a divergence mathematically.

[23] With p_G^* being a shorthand for $p_{G_{\theta^*}}$. Note that several θ^* can produce the same p_G^*.

First, we rewrite the KL divergence in a numerically tractable form:

$$KL(p_G||p_B) = \int p_G(x) \log \frac{p_G(x)}{p_B(x)}\, dx \tag{14a}$$

$$= \int p_G(x) \log \mathcal{Z}_B\, dx + \int p_G(x) \log \frac{p_G(x)}{\tilde{p}_B(x)}\, dx \tag{14b}$$

$$= \log \mathcal{Z}_B + \int p_G(x) \log \frac{p_G(x)}{\tilde{p}_B(x)}\, dx \tag{14c}$$

$$= \log \mathcal{Z}_B + \int q_{\mathcal{N}}(z) \log \frac{q_{\mathcal{N}}(z) \cdot \left|\det\left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}{\tilde{p}_B(G(z))}\, dz \tag{14d}$$

$$= \log \mathcal{Z}_B - S_{\mathcal{N}} + \int q_{\mathcal{N}}(z) \log \frac{\left|\det\left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}{\tilde{p}_B(G(z))}\, dz \tag{14e}$$

$$= \log \mathcal{Z}_B - S_{\mathcal{N}} + \int q_{\mathcal{N}}(z) \log \frac{\left|\det\left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}{e^{-\beta U_B(G(z))}}\, dz \tag{14f}$$

$$= \log \mathcal{Z}_B - S_{\mathcal{N}} + \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\beta U_B(G(z_{\mathcal{N}})) + \log \left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|^{-1}\right] \tag{14g}$$

$$= \log \mathcal{Z}_B - S_{\mathcal{N}} + \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\beta U_B(G(z_{\mathcal{N}})) - \log \left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|\right] \tag{14h}$$

with:

- (14a) by definition of the KL divergence
- (14b) by using $p_B = \tilde{p}_B / \mathcal{Z}_B$
- (14c) by using $\int p_G(x)dx = 1$ (probabilities sum to one)
- (14d) by using the change of variable formula (from equation 11)
- (14e) by definition of the entropy: $S_{\mathcal{N}} = S(q_{\mathcal{N}}) = -\int q_{\mathcal{N}}(x) \log q_{\mathcal{N}}(x)dx$
- (14f) by using: $\tilde{p}_B(x) = e^{-\beta U_B(x)}$
- (14g) by definition of expectations: $\mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[f(z_{\mathcal{N}})\right] = \int q_{\mathcal{N}}(z) f(z) dz$

The gradient of equation 14h can be expressed as:

$$\nabla_\theta KL(p_G||p_B) = \nabla_\theta \left[\log \mathcal{Z}_B - S_{\mathcal{N}} + \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\beta U_B(G(z_{\mathcal{N}})) - \log \left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|\right]\right] \tag{15a}$$

$$= \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \nabla_\theta \left[\beta U_B(G(z_{\mathcal{N}})) - \log \left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|\right] \tag{15b}$$

There are a few things to notice here:

1. Since $\log \mathcal{Z}_B$ and $-S_{\mathcal{N}}$ are constants, those terms can be safely ignored during training. This is very convenient since $\mathcal{Z}_B$ is generally unknown.
2. The transition from equation 15a to equation 15b (which puts the gradient operator ∇_θ *within* the expectation) is the principle behind Stochastic Gradient Descent. It is only allowed if the probability that is sampled in the expectation ($q_{\mathcal{N}}$ in this case) is not itself differentiated with respect to θ.

3. Computing the expectation of equation 15b seems difficult since it would require sampling an infinite amount of points. Instead, training is performed by minimizing a loss function (inside the brackets) repeatedly over many points. In practice, mini-batches of finite size n are used and resampled at each iteration.

$KL(p_G||p_B)$ therefore leads to the definition of the following empirical loss function over mini-batches:

$$\mathcal{L}_{KLx}(\mathbf{z}_\mathcal{N}) = \sum_{i=1}^{n} \left[\frac{1}{n} \cdot \left[\beta U_B(G(z_{\mathcal{N},i})) - \log \left| \det \left(\frac{\partial G(z_{\mathcal{N},i})}{\partial z_{\mathcal{N},i}} \right) \right| \right] \right] \tag{16}$$

with $\mathbf{z}_\mathcal{N} \sim q_\mathcal{N}^n$ a mini-batch of size n, and $z_{\mathcal{N},i} \sim q_\mathcal{N}$ its i$^{\text{th}}$ element.[24]

This loss[25] (originally defined in Noé et al. [39]) is very intuitive since it amounts to a minimization of the potential energy of generated samples with $U_B(G(z_\mathcal{N}))$ and a maximization of the local expansion with $-\log \left| \det \left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}} \right) \right|$.[26] The balance between those two terms is controlled through β, which determines the temperature of the generated distribution. At high temperatures, $\beta = \frac{1}{k_B T}$ is small and the entropy term dominates, whereas at low temperatures, β is high and the energy term dominates.

This loss is also very similar to the one used in conventional Normalizing Flows [40] which minimizes $KL(q_F||q_\mathcal{N})$ (see appendix A.6). In that case, data samples from p_B are required to compute $F(x_B) = G^{-1}(x_B)$ by using the model in the other direction (the "inverse" pass) and by leveraging the fact that it is bijective[27]:

$$\mathcal{L}_{KLz}^{\text{dd}}(\mathbf{x}_B) = \sum_{i=1}^{n} \left[\frac{1}{n} \cdot \left[\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_{B,i})) - \log \left| \det \left(\frac{\partial F(x_{B,i})}{\partial x_{B,i}} \right) \right| \right] \right] \tag{17}$$

with

- $\mathbf{x}_B \sim p_B^n$ a mini-batch of size n, and $x_{B,i} \sim p_B$ its i$^{\text{th}}$ element
- σ the standard deviation of the multivariate Gaussian distribution used to generate $z_\mathcal{N}$ (denoted $\mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\sigma}^2))$, see appendix A.1).

Note that when using $\mathcal{L}_{KLx}(\mathbf{z}_\mathcal{N}) + \mathcal{L}_{KLz}^{\text{dd}}(\mathbf{x}_B)$ the second terms of these two losses (the log of the determinant of the Jacobians) cancel each other when the global

[24] It is preferable to have a good approximation of the gradient rather than the gradient of a good approximation. This is why the discretization step (which is the approximation that is made when using mini-batches) is done last.

[25] $\mathcal{L}_{KLx}(\mathbf{z}_\mathcal{N})$ is a function of a mini-batch $\mathbf{z}_\mathcal{N}$. The corresponding loss associated to a single data point $z_\mathcal{N}$ is denoted: $\mathcal{L}_{KLx}(z_\mathcal{N}) = \beta U_B(G(z_\mathcal{N})) - \log \left| \det \left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}} \right) \right|$. In a slight abuse of notation, the same operator $\mathcal{L}_{KLx}$ is used for both the loss function applied to a single data point and the loss function applied to a mini-batch.

[26] When averaged over x-space, this is, up to an additive constant, the Helmholtz free energy of the generated distribution (see section 3.1).

[27] If the pushforward measure q_F of p_B by F is sufficiently close to $q_\mathcal{N}$, then the pushforward measure p_G of $q_\mathcal{N}$ by G should be sufficiently close to p_B.

optimum is reached (i.e. when $p_B = p_G$). This is not true before convergence since the terms are not sampled over the same distribution.

To summarize, two losses are available at this point:

- $\mathcal{L}_{KLx}(z_{\mathcal{N}})$ has its energy term applied to the output x_G of a forward pass that itself takes $z_{\mathcal{N}}$ as input. Its local entropy term is a direct function of $z_{\mathcal{N}}$. The batch version of this loss is $\mathcal{L}_{KLx}(\mathbf{z}_{\mathcal{N}})$.
  It is sometimes referred to as just "KLx" in the legends of some figures.
  This loss does not rely on data.[a] See equations 14 and 16.
- $\mathcal{L}^{\text{dd}}_{KLz}(x_B)$ has its energy term applied to the output z_F of an inverse pass that itself takes x_B as input. Its local entropy term is a direct function of x_B. The batch version of this loss is $\mathcal{L}^{\text{dd}}_{KLz}(\mathbf{x}_B)$.
  It is sometimes referred to as just "KLz" in the legends of some figures.
  This loss relies on data from which to sample x_B.[b] See equations 92 and 17.

[a] A data-free version of $\mathcal{L}^{\text{dd}}_{KLz}$, denoted $\mathcal{L}^{\text{df}}_{KLz}$, is introduced in section 5.1.
[b] A data-dependent version of $\mathcal{L}_{KLx}$ could also be formulated.

4.2 Experiments on small molecules

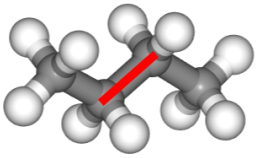
First, this section investigates what happens when these losses are used to try to capture the distribution of the configurations of two small molecules (butane and dialanine):

- The butane molecule is known to have three main modes (figure 4b), corresponding to different values of the dihedral angle ϕ of its carbon chain (around the red axis in figure 4a).
- The dialanine molecule also has several modes (shown in the Ramachandran plot of figure 5b), with the main ones (C7eq and C7ax) corresponding to movements around the backbone dihedral angles ϕ and ψ (represented by the red axes in figure 5a).

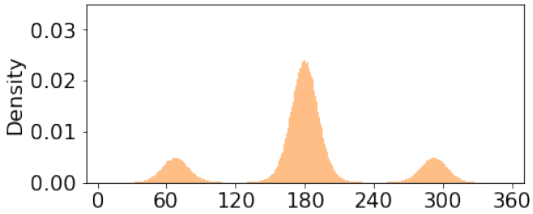
For both molecules, the model (whose architecture is described in section 3.4) is trained to generate all the atomic Cartesian coordinates from a Gaussian vector of the same dimensionality c.[28] For butane this amounts to $14 \times 3 = 42$ coordinates, and for dialanine $22 \times 3 = 66$ coordinates. Other types of molecular encodings (sometimes involving internal coordinates) can be considered, mainly to simplify the task, but also to make transfer learning between molecules possible (as described in section 9).

In both figures (4 and 5), $\mathcal{L}^{\text{dd}}_{KLz}$ is used during pre-trainings and $\mathcal{L}_{KLx}$ is used during fine-tunings (see section 4.4 for more information on the experimental setup).

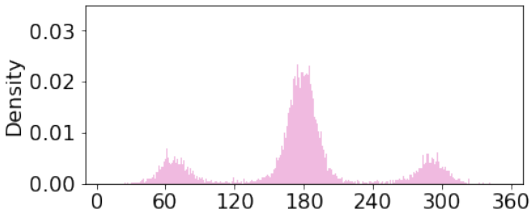
[28] Both `Butane` and `Dialanine` datasets are the result of long Metropolis-Hastings simulations with Parallel tempering.



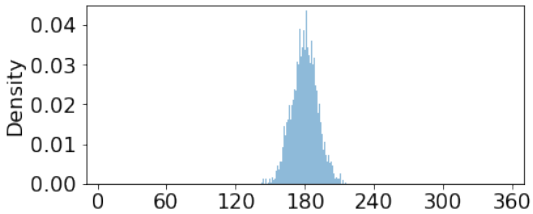
(a) Butane molecule



(b) Reference distribution of ϕ

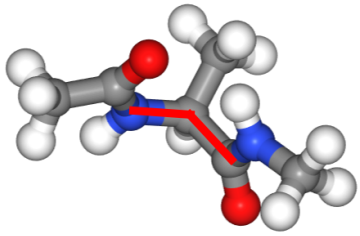


(c) Generated distribution of the dihedral angle ϕ after pre-training with: $\mathcal{L}_{KLz}^{\text{dd}}$

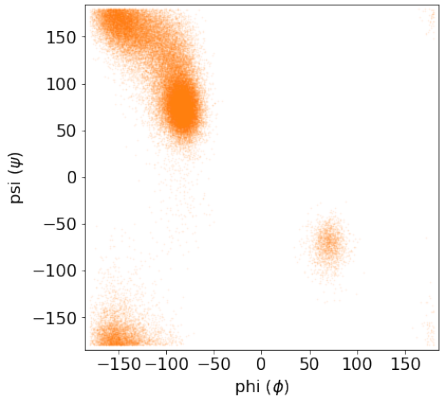


(d) Generated distribution of the dihedral angle ϕ after fine-tuning with: $\mathcal{L}_{KLx}$

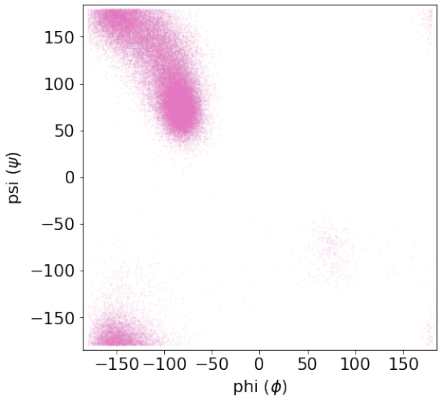
Figure 4: Pre-training with $\mathcal{L}_{KLz}^{\text{dd}}$ on `Butane` (in pink), followed by a fine-tuning with $\mathcal{L}_{KLx}$ (in blue) *without alignment penalty*. Reference data is in orange according to the color theme described in section 0.



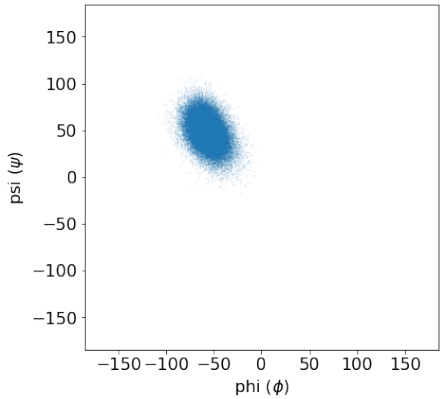
(a) Dialanine molecule



(b) Reference joint distribution of ϕ and ψ



(c) Generated dihedral angles ϕ and ψ after pre-training with: $\mathcal{L}_{KLz}^{\text{dd}}$



(d) Generated dihedral angles ϕ and ψ after fine-tuning with: $\mathcal{L}_{KLx}$

Figure 5: Pre-training with $\mathcal{L}_{KLz}^{\text{dd}}$ on `Dialanine` (in pink), followed by a fine-tuning with $\mathcal{L}_{KLx}$ (in blue) *without alignment penalty*. Reference data is in orange according to the color theme described in section 0.

Importantly, the alignment penalty (described in section 2.4) is removed during the fine-tuning (more on this in section 5.2).[29]

$\mathcal{L}_{KLz}^{\text{dd}}$ works more or less as intended (figures 4c and 5c) although it leads to an overestimation of the probability density in some regions of the space (like the inter-modes of butane for example). This problem could be referred to as the "mode-add" problem, which is assumed to be caused by the fact that the energy of generated samples is not penalized directly. This loss is well defined because it takes *real* molecular configurations as inputs and therefore forces the generated configurations to span the entire space covered by the dataset.

The most concerning problem is that when the generator is pre-trained with $\mathcal{L}_{KLz}^{\text{dd}}$ and then fine-tuned with $\mathcal{L}_{KLx}$ without alignment penalty, the multimodality is *lost*. This problem is often referred to as "mode-drop" or "mode collapse" or "mode-seeking" in the literature. Once a mode is lost it is never recovered, since the exploration strategy implicitly used here (which consists in maximizing the expansion with the term $-\log\left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|$ is only local, not global, and generally not sufficient to break through high free-energy barriers. The model has no incentive to match the *full* Boltzmann distribution, and is not penalized for ignoring important parts of the target distribution since $\mathcal{L}_{KLx}$ is only defined on x_G points from p_G.

Note that it is highly probable that q_F also experiences some mode collapse in z-space when using $\mathcal{L}_{KLz}^{\text{dd}}$. But since the target distribution $q_{\mathcal{N}}$ has only one mode, that behavior is not noticeable.

4.3 Gradient of $KL(p_G||p_B)$

When minimizing $KL(p_G||p_B)$, p_G follows the following negative gradient (see appendix A.3 for a proof):

$$-\nabla_{p_G} KL(p_G||p_B) = \log(p_B) - \log(p_G) - 1 \tag{18}$$

A stable point is reached when $\nabla_{p_G} KL(p_G||p_B) = 0$. At first glance, this would imply that the optimization ends when: $\log(p_B) - \log(p_G) - 1 = 0 \implies p_G = \frac{p_B}{e}$.

But since p_G is assumed to be a *probability* distribution, the optimization being performed is actually under the constraint that $\int p_G = 1$. This implies that p_G cannot be decreased (nor increased) everywhere, and constants (like the -1 term of equation 18) can be ignored. Therefore, the steady point is indeed reached when $p_G = p_B$.

[29] Discarding the alignment penalty during fine-tuning makes the performance *clearly worse* but is done on purpose to showcase the phenomenon of mode collapse. As studied in section 5.2, keeping the alignment penalty alleviate the problem *but does not fix it.*

However, crucially, the same conclusion cannot be drawn when optimizing on batches of finite size. In this case, the loss is not rebalancing the distribution over the whole support but only *between the points that have been sampled in the batch.* Indeed, equation 14h uses an *exact* expectation value which cannot be computed on batches of finite size. But does the sum of the probabilities of the batch stays the same after one gradient update? In other words: Are the points that are sampled in each batch becoming more or less probable *globally* (when compared to all the points, including those that have not been sampled)?

No approximation is made between equation 14a and equation 15b, and $\mathcal{Z}_B$ should be unnecessary when using this particular training objective. But many other approximations used elsewhere could be causing the observed mode collapse regardless. For example:

- The expectation from equation 14h cannot be computed exactly over the *whole* space, and is estimated with batches have a finite size instead.
- The model itself has limited expressivity, which means that it cannot capture every possible probability distribution and may suffer from an unfavorable inductive bias.
- Even the training procedure is performed with optimizers (like Adam [24] or RMSProp [52]) that change the dynamics of the optimization and that are dependent on important hyperparameters like the learning rate.

Since the mode collapse could come from any of those (known or unknown) reasons or any combination thereof, section 4.4 focuses on describing its behavior under different experimental settings.

4.4 Observed behavior of the mode collapse

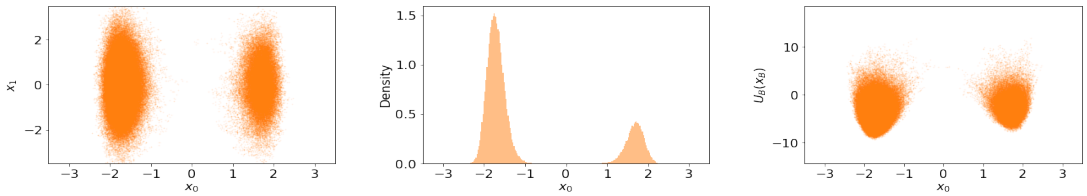
As a quick reminder, the objective of this work is to successfully train a model *without data* (see end of section 1.2), by only relying on the information coming from the energy U_B. In practice, since training from scratch on large molecules is probably extremely challenging, some form of curriculum learning should probably be used (see section 9.4). To imitate this setup, most experiments are performed in two steps:

- A pre-training step, which *does* use data but only for the purpose of proper initialization of the model during the next step.
- And a fine-tuning step which *does not* use data (except for illustrative purposes).

Three `Double Well` datasets are used in this section: `Double Well Medium 12D`, `Double Well Wide 12D` and `Double Well Narrow 12D`.[30] They are all composed of one bimodal dimension (denoted x_0) and 11 centered unimodal Gaussian dimensions.

[30]Much like with `Butane` and `Dialanine`, those datasets are the result of long Metropolis-Hastings simulations with Parallel tempering.

The only difference between the datasets is the standard deviation of the 11 Gaussians: in the `Medium` the standard deviation is 1 (see figure 6), in the `Wide` case the standard deviation is 10 and in the `Narrow` case the standard deviation is 0.1.



(a) 2D Projection of the dataset

(b) Projection of the dataset on its multimodal dimension

(c) U_B energies of real samples $x_B \sim p_B$ as a function of x_0

Figure 6: Various representations of the `Double Well Medium 12D` Dataset.

The conventional loss $\mathcal{L}^{\text{dd}}_{KLz}$ is used here during pre-training on data from `Double Well Medium 12D`.[31] The model is composed of a stack of 8×4 Coupling Blocks (as described in sections 3.2.4 and 3.4) and is able to perfectly capture the target distribution (see figure 7), with the exception of a slight residual connection between the two modes. See figures 43 and 44 in appendix A.4 for a visualization of the transformation performed by a similar model on `Double Well Medium 2D`.

The model obtained at the end of the pre-training with $\mathcal{L}^{\text{dd}}_{KLz}$ (from figure 7) can then be fine-tuned with $\mathcal{L}_{KLx}$ (see figure 8). Unlike the results from figures 4d and 5d, no mode collapse is observed.

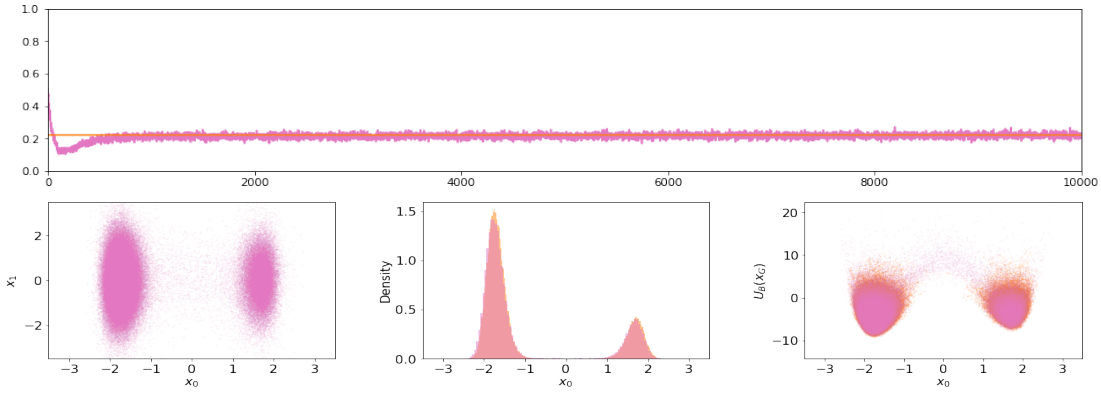


Figure 7: Pre-training with $\mathcal{L}^{\text{dd}}_{KLz}$ on `Double Well Medium 12D`. Top: Percentage of points generated by p_G in the minor mode during training (in pink) compared with the real ratio of ≈ 0.22 from the dataset (in orange). Bottom Left: Scatter plot of p_G projected in 2D. Bottom Center: Histogram of p_G projected onto its multimodal dimension x_0. Bottom Right: Scatter plot of U_B energies of generated samples $x_G \sim p_G$ (in pink) and U_B energies of real samples from the dataset $x_B \sim p_B$ (in orange).

The results obtained on `Double Well Wide 12D` give a very different picture. The dataset itself seems virtually identical to `Double Well Medium 12D` except for the standard deviation of the 11 Gaussian dimensions. The pre-training however is a bit less successful since after 2000 iterations the ratio between the 2 modes is still

[31] Unless specified otherwise, every pre-training and fine-tuning is done with Adam and a learning rate of 10^{-4} on `Double Well` datasets and 10^{-5} on molecular datasets, without regularization.

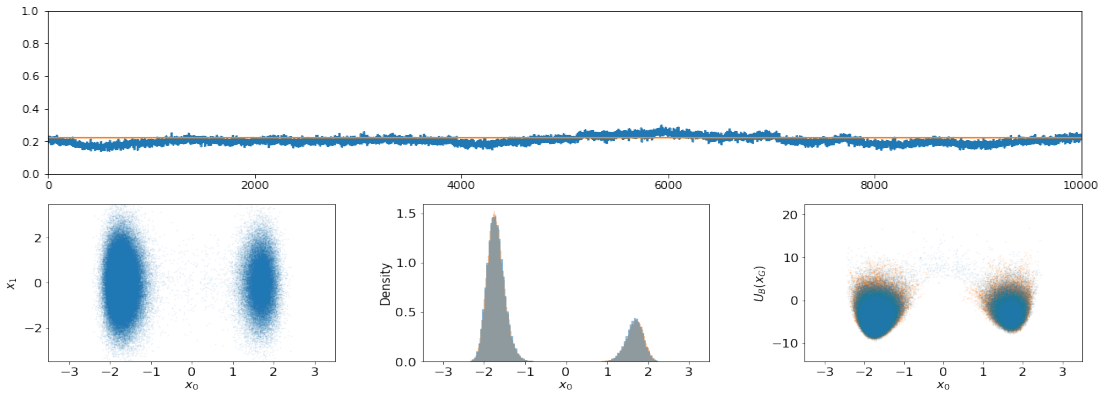


Figure 8: Fine-tuning with $\mathcal{L}_{KLx}$ on `Double Well Medium 12D` (see caption of figure 7 for more details).

incorrect and the energies remain too high (see figure 9).[32] When fine-tuning with $\mathcal{L}_{KLx}$ alone (in figure 10), p_G completely collapses after less than 200 training steps. This collapse is reminiscent of the ones observed in figure 4d and 5d.[33]

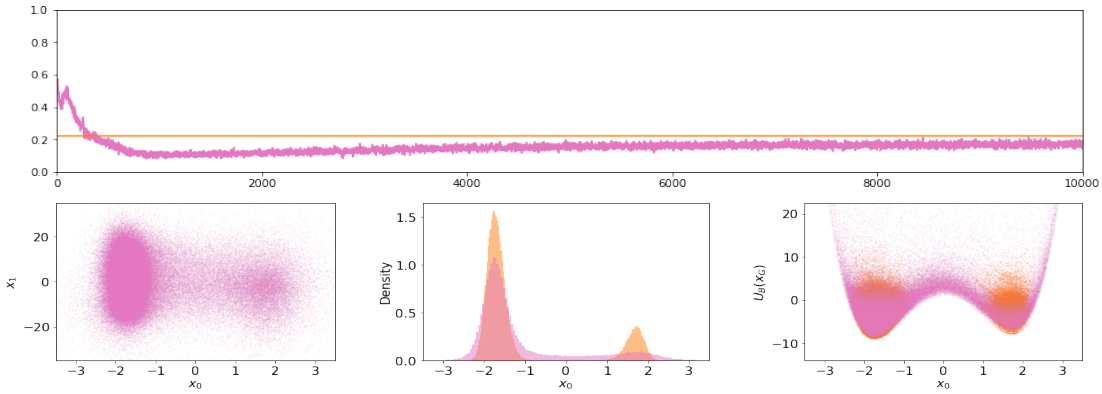


Figure 9: Pre-training with $\mathcal{L}^{\text{dd}}_{KLz}$ on `Double Well Wide 12D` (see caption of figure 7 for more details).

The difference between the perfect generation of figure 8 and the collapse of figure 10 suggests that the relative standard deviation of the 11 unimodal dimensions plays a role in the collapse. Locally, this corresponds to flater or steeper directions of space. The flater the other directions, the more the model is prone to collapse in the multimodal direction. When working on `Butane`, the dimensions of p_B in Cartesian coordinates can roughly be separated in 3 groups:

- The multimodal direction of interest corresponding to different values of its central dihedral angle.
- Other directions that also correspond to changes in the shape of the molecule (like bond lengths and angles). These directions are typically unimodal and quite steep in the sense that small variations in the shape of the molecule

[32] A longer pre-training does lead to a correct ratio between the modes and also improves the collapse situation during fine-tuning as discussed in section 4.4

[33] To observe the collapse "as it happens", see figure 45 in appendix A.5.

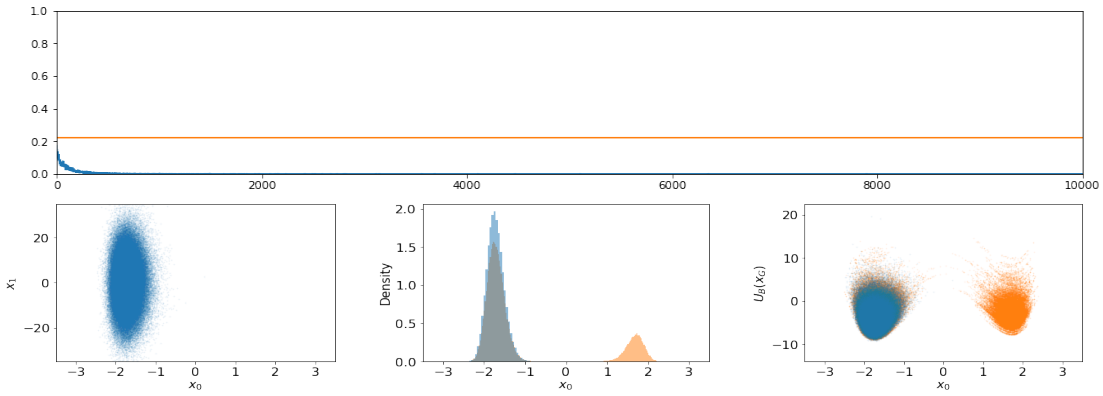


Figure 10: Fine-tuning with $\mathcal{L}_{KLx}$ on `Double Well Wide 12D` (see caption of figure 7 for more details).

lead to large changes in potential energy U_B. As long as there is no numerical instability (see section 2.3), they don't seem to be an issue from a optimization perspective.

- And then remains the 6 degrees of freedom related to movements of global translation and rotation. These directions, if not properly penalized, make the collapse problem much worse since they are absolutely flat (see section 5.2). As a consequence, it is important to not forget the $U_{\text{align}}(x)$ term within the energy function $U_B(x)$ (which penalize both translations and rotations as described in section 2.4) both during simulation (when generating the pre-training data) and during generation (when training the model).

When fine-tuning with both $\mathcal{L}^{\text{dd}}_{KLz}$ and $\mathcal{L}_{KLx}$ (in figure 11), there is no collapse but the ratio between the two wells still gets worse after the poor pre-training of figure 9, hinting to the fact that $\mathcal{L}_{KLx}$ may lead to collapse in an *active* manner and not as a consequence of the variance inherent to the discretization of the dataset in batches.[34]

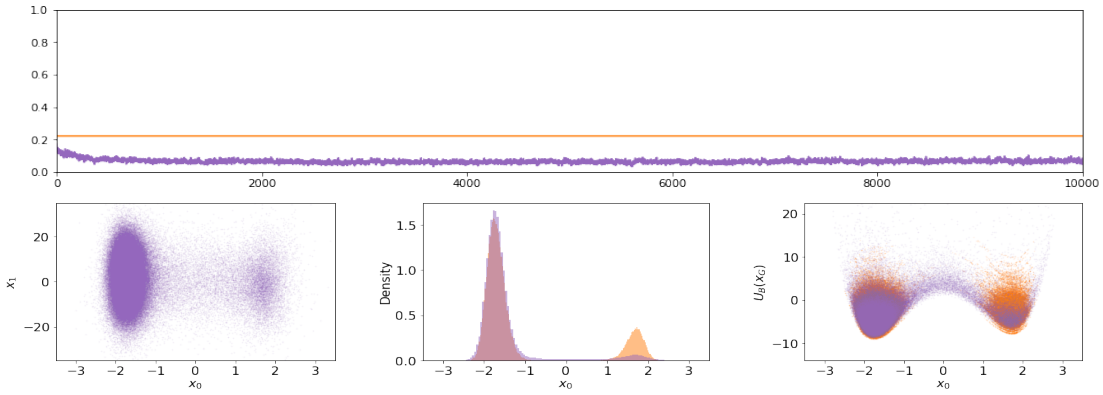
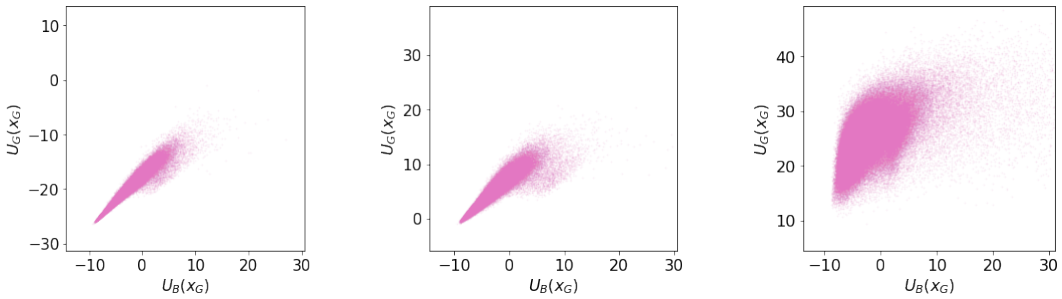


Figure 11: Fine-tuning with *both* $\mathcal{L}^{\text{dd}}_{KLz}$ and $\mathcal{L}_{KLx}$ on `Double Well Wide 12D` (see caption of figure 7 for more details).

[34] $\mathcal{L}_{KLx}$ is indeed the culprit here since using $\mathcal{L}^{\text{dd}}_{KLz} + 0.01 \cdot \mathcal{L}_{KLx}$ leads to ratios that are indistinguishable from the correct ones, but increasing the factor from 0.01 to 1 yields worse ratios.

4.5 Relationship between pre-training quality and collapse

At the end of the pre-training, the performance of the *same* model architecture is already much worse on `Double Well Wide 12D` (figure 9) than it is on `Double Well Medium 12D` (figure 7) which is an almost identical dataset. This is further illustrated in figure 12 when comparing the correlations between U_B and U_G. Note that the *only* change between figures 12a, 12b, and 12c is the standard deviation of the 11 unimodal dimensions of the datasets.



(a) `Double Well Narrow 12D` (with unimodal standard deviations of **0.1**)

(b) `Double Well Medium 12D` (with unimodal standard deviations of **1**)

(c) `Double Well Wide 12D` (with unimodal standard deviations of **10**)

Figure 12: Correlations between U_B and U_G on generated samples $x_G \sim p_G$ after pre-training on several `Double Well 12D` datasets with $\mathcal{L}^{\text{dd}}_{KLz}$. The more "diagonal" the point cloud is, the better the quality of generation.

This is surprising, since the model is perfectly able to simply multiply by a learned factor each p_G dimension that needs it, and yet, has difficulty doing it. This is an important fact to notice already in 12 dimensions, since it shows how critical data normalization is.[35] It also implies that simplifying the data (as with the removal of hydrogen atoms in section 3.3) or using more expressive models adapted to the structure of molecular data (see sections 2.4 and 9) could go a long way toward improving the performance of generative models on molecular data.

In a more general sense, it may be that the pre-training performance (measured by the ability of the model to correlate U_B and U_G) may predict the risk of collapse. A much longer pre-training on `Double Well Wide 12D` (similar to the one from figure 9 but 50 times longer) eventually leads to the correct ratios (data not shown). Fine-tuning with $\mathcal{L}_{KLx}$ from that point onward does not lead to collapse (see figure 13) contrary to the fine-tuning of figure 10 which suffered from an inferior pre-training quality.

Given this result, it may be tempting to think that the solution is simply to pre-train the model longer. However, in order to scale to larger molecules with curriculum learning (see section 9.4), one must assume that the pre-training quality will often not be sufficient. Besides, already in figure 9, the bottom of both wells

[35] Ensuring that models can be trained just as efficiently regardless of the relative scaling of the dimensions of the data seems a promising avenue of research. Noé et al. [39] uses a data-dependent whitening layer.

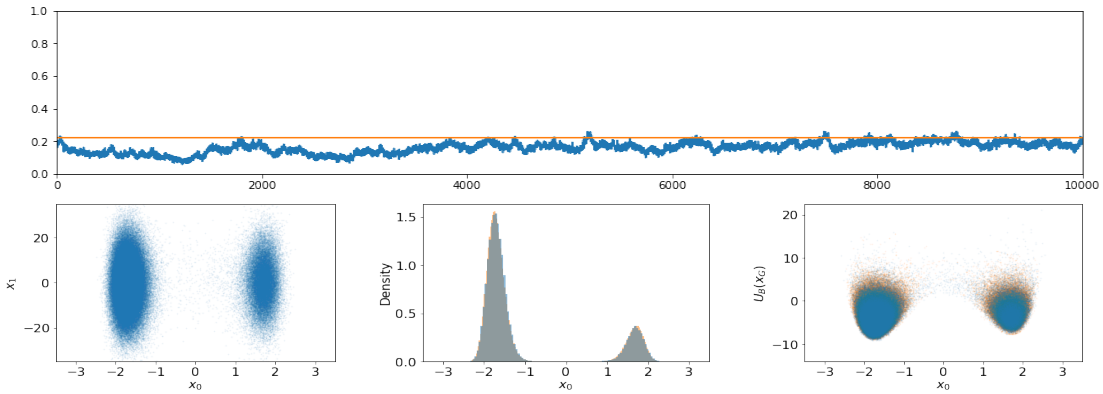


Figure 13: Fine-tuning with $\mathcal{L}_{KLx}$ on `Double Well Wide 12D` after a pre-training 10 times longer than the one from figures 9 and 12c used in the fine-tunings of figures 10 and 11 (see caption of figure 7 for more details).

are properly sampled, and whatever fine-tuning is applied after that should not lead to the loss of a mode.

In summary, although poor pre-trainings can exacerbate the collapse problem, they do not cause it. Simplifying the data or increasing the expressivity of the model may improve the situation but does not fix the root issue. One must keep in mind that in order to scale to larger molecules, fine-tunings should be stable (i.e. they should not collapse) *regardless* of the pre-training quality, as long as there are at least a few points in each mode.

Another important result is that the *temperature* of the data also has an impact on the risk of collapse. In figure 14, the temperature of the target distribution p_B is increased by a factor 2, and as a result, the stability of the fine-tuning noticeably improves.[36] This is made clear by comparing the curve of figure 14 (which does not display any noise beyond the variance induced by the sampling of mini-batches) with the one of figure 13 (which is noisy in a *time-correlated fashion*).

It could be suggested that the pre-training quality has a favorable impact or that more balanced wells, with a lower free-energy barrier, are easier for the model to generate.[37]

Another supposition could be made that the better performance at the higher temperature comes from the fact that it decreases the influence of U_B within the loss (by reducing $\beta = C^{te}/T$ in equation 16). This would make sense since a relative lower influence of U_B implies a relative higher influence of the local expansion term $-\log|\det(J_G)|$ which directly counteracts the collapse. A more comprehensive hypothesis, justifying all the observations of this section (why mode collapses is made

[36] Since the data changes, new simulations and pre-trainings have to be prepared accordingly.

[37] This is difficult to assess with certainty one way or the other since comparing the quality of generative models is notoriously hard. Even with a good metric (i.e. divergence) between the two distributions to compare, it is virtually impossible to sample enough points to estimate it properly in high dimension. The strategy used in the remainder of this work (starting in figures 17b and 17c) is to measure the potential energy of generated samples (i.e. $U_B(x_G)$) and the energy of generation of samples from the dataset (i.e. $U_G(x_B)$). More on this at the end of section 5.1.

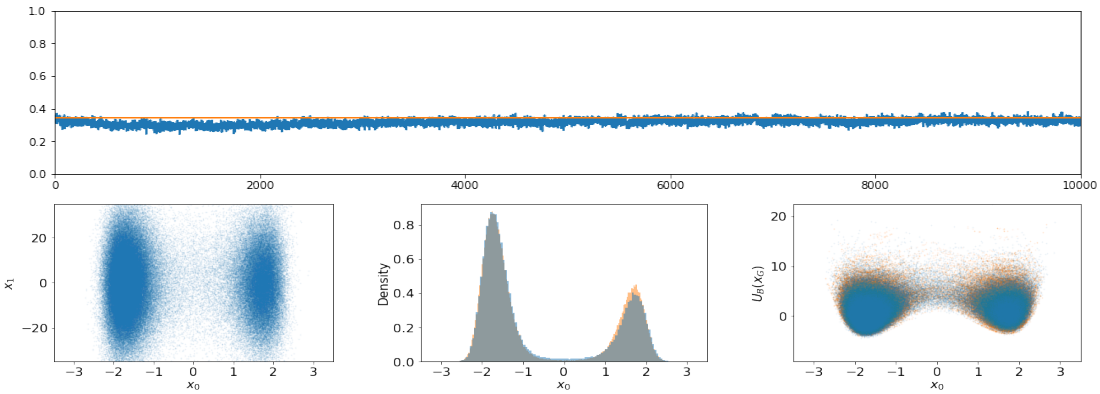
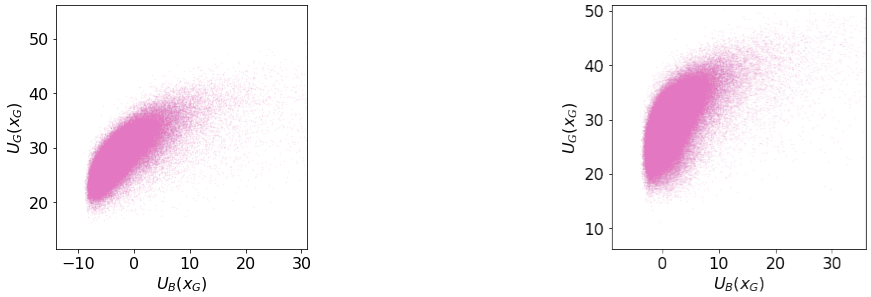


Figure 14: Fine-tuning with $\mathcal{L}_{KLx}$ on `Double Well Wide 12D` at a temperature of $\times 2$ (see caption of figure 7 for more details). Note that the reference data used here in orange and the pre-training (not shown) are also at a temperature of $\times 2$ and that the target ratio between the modes increases from ≈ 0.22 (at a temperature of $\times 1$) to ≈ 0.37 (at a temperature of $\times 2$).



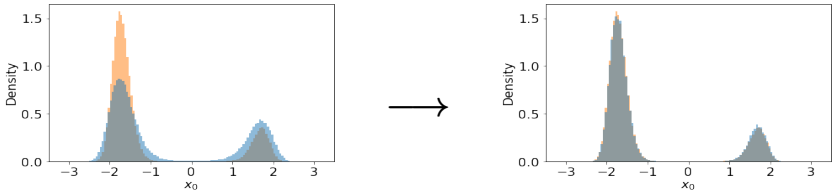
(a) Pre-training at a temperature of $\times 1$ as used in figure 13.

(b) Pre-training at a temperature of $\times 2$ as used in figure 14.

Figure 15: Correlations between U_B and U_G on generated samples $x_G \sim p_G$ after two pre-trainings (at two different temperatures) on `Double Well Wide 12D` with $\mathcal{L}^{\text{dd}}_{KLz}$.

worse by poor data normalization, by inferior pre-trainings, and low temperatures) and the next (why the alignment penalty is so critical), is formulated in section 5.4.

Finally, note that training at higher temperatures is not prohibitive for density estimation since the generated distribution p_G can be re-weighted by a factor $\frac{\tilde{p}_B}{p_G}$ after the training is complete (equation 9, as demonstrated in figure 16).



(a) Before re-weighting.

(b) After re-weighting.

Figure 16: Re-weighting of the p_G distribution after fine-tuning with $\mathcal{L}_{KLx}$ on `Double Well Wide 12D` at a temperature of $\times 2$ (as in figure 14). Here, the reference data in orange is at a temperature of $\times 1$.

5 Using batch weights for data-free trainings

5.1 Data-free $KL(q_F||q_\mathcal{N})$

$\mathcal{L}^{\text{dd}}_{KLz}$ probably leads to collapse in z-space, but since the target multivariate Gaussian distribution $q_\mathcal{N}$ has only one mode, the problem is less critical and does not lead to collapse in x-space. One limitation of using this loss is that it requires real x_B samples from the target distribution p_B (see equation 17), but properly detached $x^\ddagger_G$ samples can be used instead by using importance sampling:

$$\nabla_\theta KL(q_F||q_\mathcal{N}) = \nabla_\theta \left[\mathop{\mathbb{E}}_{x_B\sim p_B} \left[\frac{1}{2\sigma^2} U_\mathcal{N}(F(x_B)) - \log \left| \det \left(\frac{\partial F(x_B)}{\partial x_B} \right) \right| \right] \right] \tag{19a}$$

$$= \nabla_\theta \left[\int p^\ddagger_B(x) \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x)) - \log \left| \det \left(\frac{\partial F(x)}{\partial x} \right) \right| \right) dx \right] \tag{19b}$$

$$= \nabla_\theta \left[\int p^\ddagger_G(x) \left(\frac{p_B(x)}{p_G(x)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x)) - \log \left| \det \left(\frac{\partial F(x)}{\partial x} \right) \right| \right) dx \right] \tag{19c}$$

$$= \nabla_\theta \left[\mathop{\mathbb{E}}_{x^\ddagger_G\sim p^\ddagger_G} \left[\left(\frac{p_B(x^\ddagger_G)}{p_G(x^\ddagger_G)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x^\ddagger_G)) - \log \left| \det \left(\frac{\partial F(x^\ddagger_G)}{\partial x^\ddagger_G} \right) \right| \right) \right] \right] \tag{19d}$$

$$= \frac{1}{\mathcal{Z}_B} \cdot \nabla_\theta \left[\mathop{\mathbb{E}}_{x^\ddagger_G\sim p^\ddagger_G} \left[\left(\frac{\tilde{p}_B(x^\ddagger_G)}{p_G(x^\ddagger_G)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x^\ddagger_G)) - \log \left| \det \left(\frac{\partial F(x^\ddagger_G)}{\partial x^\ddagger_G} \right) \right| \right) \right] \right] \tag{19e}$$

$$= \frac{1}{\mathcal{Z}_B} \cdot \mathop{\mathbb{E}}_{x^\ddagger_G\sim p^\ddagger_G} \nabla_\theta \left[\left(\frac{\tilde{p}_B(x^\ddagger_G)}{p_G(x^\ddagger_G)} \right)^\ddagger \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x^\ddagger_G)) - \log \left| \det \left(\frac{\partial F(x^\ddagger_G)}{\partial x^\ddagger_G} \right) \right| \right) \right] \tag{19f}$$

with:

- (19a) by taking the gradient of equation 92h.
- (19b) by definition of expectations. Note the subtle replacement of p_B with $p^\ddagger_B$ which is allowed inside the gradient operator ∇_θ since p_B is not a function of θ.
- (19c) by importance sampling.
- (19d) by definition of expectations.
- (19e) by definition of $\tilde{p}_B = \mathcal{Z}_B p_B$.
- (19f) by noticing that, although $p^\ddagger_G$ is a function of θ, it is not differentiated with respect to 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.

This leads to the definition of the following loss function:

$$\mathcal{L}^{\text{df}}_{KLz}(\mathbf{x}^\ddagger_G) = \sum_{i=1}^{n} \frac{1}{n} \cdot \left[w_i \cdot \left(\frac{1}{2\sigma^2} U_\mathcal{N}(F(x^\ddagger_{G,i})) - \log \left| \det \left(\frac{\partial F(x^\ddagger_{G,i})}{\partial x^\ddagger_{G,i}} \right) \right| \right) \right] \tag{20}$$

with:

- $\mathbf{x}_G^\ddagger$ a detached mini-batch of size n, and $x_{G,i}^\ddagger \sim p_G^\ddagger$ its i$^{\text{th}}$ element.
- w_i a quantity proportional to $\left(\tilde{p}_B(x_G^\ddagger)/p_G(x_G^\ddagger)\right)^\ddagger$ as in equation 19f. It is this term that focuses the loss onto regions of the space that are under-sampled by the model. The set $\{w_i | i \in [1, ..., n]\}$ is referred to as the "batch weights" of the loss function.[38]

Equation 19d cannot be turned into a loss function directly since $\mathcal{Z}_B$, and therefore p_B, are unknown. As a consequence, the fraction $\left(p_B(x_G^\ddagger)/p_G(x_G^\ddagger)\right)^\ddagger$ can only be computed up to a factor. This is not an obstacle because batch weights can be scaled for convenience by any quantity that can be factored out of both the expectation $\mathbb{E}_{x_G^\ddagger \sim p_G^\ddagger}$ and the gradient operator ∇_θ. In equation 19f, the factor $\frac{1}{\mathcal{Z}_B}$ is factored out of both, but any other multiplicative constant could be treated similarly. So defining $w_i = \frac{e^{-U_{B,i}}}{e^{-U_{G,i}}}$ should suffice in theory, but in practice, to avoid computing numerically unstable exponentials, softmaxes are used instead[39]:

$$\tilde{w}_i = \frac{\text{softmax}\left(-U_B(x_{G,i})\right)}{\text{softmax}\left(-U_G(x_{G,i})\right)} \tag{21a}$$
$$\text{or} \quad \text{softmax}\left(-U_B(x_{G,i})\right) \cdot \text{softmax}\left(U_G(x_{G,i})\right)$$
$$\text{or} \quad \text{softmax}\left(-U_B(x_{G,i}) + U_G(x_{G,i})\right)$$

$$w_i = \left[\tilde{w}_i \cdot \frac{n}{\sum_j^n \tilde{w}_j}\right]^\ddagger \tag{21b}$$

with the softmaxes being computed between the n elements of a mini-batch according to the definition: $\text{softmax}(v_i) = \frac{e^{v_i}}{\sum_j^n e^{v_j}}$. The three alternatives of equation 21a are proportional to one another[40] and lead to the same w_i after normalization in equation 21b. Note that the softmax formula is reminiscent of the one used to compute the probabilities of a Boltzmann distribution (see equation 1).[41]

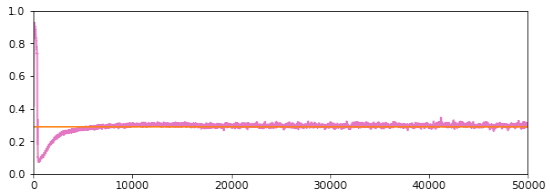
[38] Since quantities like these are importance sampling weights but only up to a factor, they are simply referred to as "batch weights" (or even just "bws" in the legends of some figures).

[39] The normalization factor of the softmaxes changes between mini-batches but this can be safely ignored.

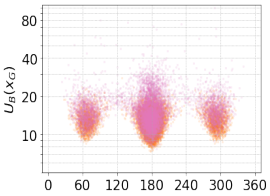
[40] The $x_{G,i}$ points used in equation 21a do not have to be detached because of the detach operator that follows in equation 21b. Mathematically: $\nabla_\theta \left[f(x^\ddagger)\right]^\ddagger = \nabla_\theta \left[f(x)\right]^\ddagger$

[41] A softmax is almost a discretized version of the Boltzmann distribution formula. However, it uses a incorrect partition function computed over a finite set of discrete samples instead of the correct partition function $\mathcal{Z}_B$, which is computed on the whole support. Another difference in our case is that the softmaxes from equation 21a use x_G points that are sampled according to p_G whereas the probabilities of a Boltzmann distribution are defined according to a *uniform* integral over the support. But all those scaling factors are made irrelevant by the normalization of equation 21b.

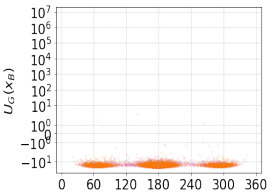
At this point, two losses are available *in a data-free setting*:
- $\mathcal{L}_{KLx}(\mathbf{z}_\mathcal{N})$ stems from a divergence defined in x-space and is referred to as just "KLx" in the legends of some figures. See equations 14 to 16.
- $\mathcal{L}^{\text{df}}_{KLz}(\mathbf{x}^\ddagger_G)$ stems from a divergence defined in z-space, leverages importance sampling, and is referred to as just "bws * KLz" in the legends of some figures. See equations 92, 19 and 20.



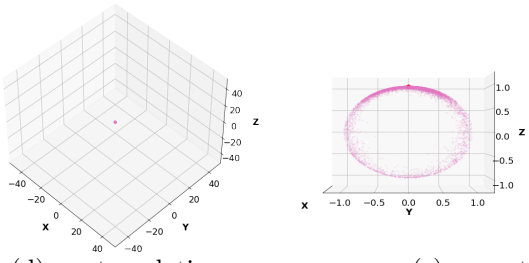
(a) Ratio of generated x_G samples in the minor modes during pre-training.



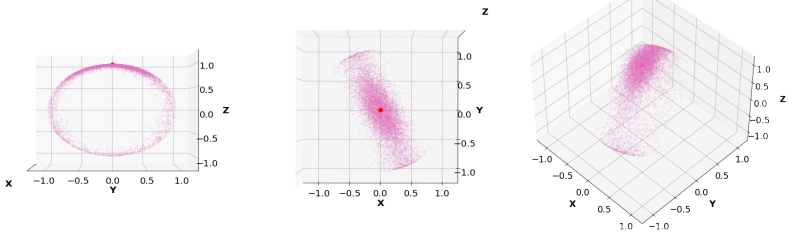
(b) U_B energies of samples $x_G \sim p_G$.



(c) U_G energies of samples $x_B \sim p_B$.



(d) x_G translations.



(e) x_G rotations under 3 different view points.

Figure 17: Pre-training with $\mathcal{L}^{\text{dd}}_{KLz} + 0.02 \cdot \mathcal{L}_{KLx}$ on `Butane`. The reference data is represented in orange and the pre-training data in pink (according to the color scheme described in section 0).
- Figure 17b is a scatter plot that compares the potential energies $U_B(x_G)$ (in pink and in the label of the ordinates) with $U_B(x_B^{\text{MinH}})$ (in orange). Here, x_B^{MinH} represents x_B points that had the energy of their hydrogen atoms minimized. This is to take into account the effect of using G^{H} which is trained to minimize the energy of the generated hydrogen atoms (as described in section 3.3).
- Figure 17c is a scatter plot that compares the energies of generation $U_G(x_B)$ (in orange) with $U_G(x_G)$ (in pink and in the label of the ordinates). Note that even though $G(F(x_B)) \neq x_B$, it is still true that $U_G(G(F(x_B))) = U_G(x_B)$ since U_G is computed on the heavy atoms only (see section 3.3). The log scale of the y-axis is kept on purpose for comparison with other figures (figures 18d and 19c in particular).
- Figure 17d is a 3D scatter plot where each point represents the barycenter of one generated $x_G \sim p_G$ sample. The large scale of the axes is chosen for better comparison with figure 19d.
- Figure 17e shows three different viewpoints of the same 3D scatter plot where each point is the rotation of the red reference point by the uncapped alignment rotation matrix of one $x_G \sim p_G$ sample (see section 2.4 for more details). In essence, it represents the rotations that align x_G samples with the configuration of reference (as used in $U_{\text{align}}(x)$).

The pre-training on butane is performed with $\mathcal{L}^{\text{dd}}_{KLz} + 0.02 \cdot \mathcal{L}_{KLx}$ and is illustrated in figure 17. Only $\mathcal{L}^{\text{df}}_{KLz}$ is required but the regularization with $0.02 \cdot \mathcal{L}_{KLx}$ speeds up the training since it adds a direct penalization of the energy U_B of the generated configurations (see equation 16).

After pre-training, the generated data is virtually identical to the real data (not

shown for this reason) and several observations can be made:

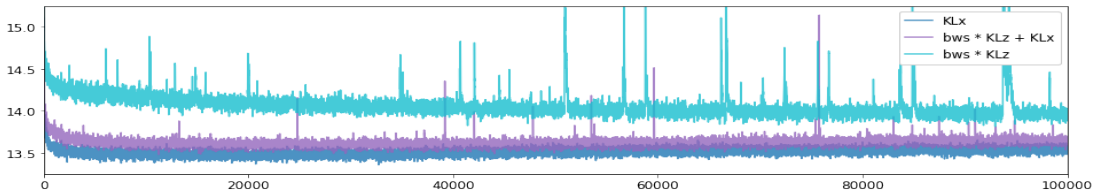
- In figure 17a the ratio of the minor modes is perfectly stable already at iteration 10^5.[42]
- In figure 17b the U_B energies of generated samples match those of the reference data: *roughly, one can say that* $p_G \subset p_B$ (in the sense that $p_G \propto p_B$ within the support of p_G). And in figure 17c the U_G energies of samples from the simulated dataset are very low and match those of generated samples: *roughly, one can say that* $p_B \subset p_G$ (in the sense that $p_B \propto p_G$ within the support of p_B).
- Figure 17d shows how the generated data is well centered. Although this cannot be seen in the figure, there is a little bit of Gaussian noise in the barycenters of the generated configurations since the centering of the distribution is not enforced exactly but through the alignment penalty.
- Figure 17e shows how one axis of rotation remains dominant during simulation (and therefore in the pre-training data) despite the alignment penalty. This is explained by the fact that rotations around the main axis of butane (the one parallel to its carbon chain) result in a lower penalty than rotations in the other directions. Since the rotation penalty is capped, the lower part of the circle has the same density everywhere, and the density is higher at the top of the circle, where the rotation penalty gets close to zero. A higher λ_{align} could be chosen to concentrate all the points near the top of the sphere (as in figure 21e).

Fine-tunings with $\mathcal{L}_{KLx}$, $\mathcal{L}^{\text{df}}_{KLz}$ and a combination thereof are represented in figure 18:

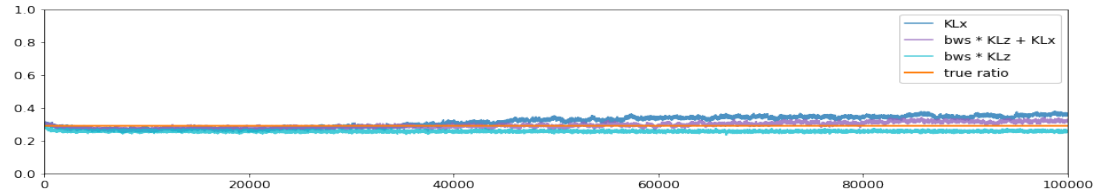
- Figure 18a shows how using $\mathcal{L}_{KLx}$ (which penalizes $U_B(x_G)$ directly in one of its terms) lead to lower energies very early during training when compared with using $\mathcal{L}^{\text{df}}_{KLz}$ alone which optimizes U_B only indirectly.
- Figure 18b shows ratios between the modes that are almost correct with all losses, although one can notice that $\mathcal{L}_{KLx}$ seems to be slightly unstable when compared with $\mathcal{L}^{\text{df}}_{KLz}$. This is a phenomenon that can be exacerbated and that is investigated in section 5.2.
- Figure 18c shows an excellent distribution of $U_B(x_G)$ for both losses but figure 18d suggests that, when using $\mathcal{L}_{KLx}$, some collapse is occurring *within* the central mode since some data samples have an energy of generation U_G that is orders of magnitude too high. A similar phenomenon occurs in figure 19c which is explained by the fact that some rotations are lost during training (as illustrated in figure 19e).

$\mathcal{L}^{\text{df}}_{KLz}$ is the first loss that is stable in a data-free regime.
$\mathcal{L}_{KLx}$, as investigated in section 5.2, leads to more ambiguous results.

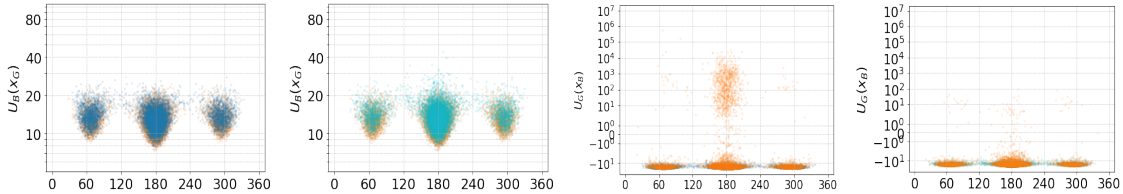
[42] The pre-training is carried out for $4 \cdot 10^5$ more iterations to ensure that its quality is not redhibitory in the subsequent fine-tunings (such as those of figures 18 and 19).



(a) Potential energies U_B of generated x_G samples during fine-tuning.



(b) Ratio of generated x_G samples in the minor modes during fine-tuning.



(c) U_B energies of samples $x_G \sim p_G$.

(d) U_G energies of samples $x_B \sim p_B$.

Figure 18: Comparison of three data-free fine-tunings on `Butane` with $\lambda_{\text{align}} = 10$.
- In blue: a fine-tuning with $\mathcal{L}_{KLx}$ alone.
- In purple: a fine-tuning with $\mathcal{L}^{\text{df}}_{KLz} + \mathcal{L}_{KLx}$.
- In cyan: a fine-tuning with $\mathcal{L}^{\text{df}}_{KLz}$ alone.
See caption of figure 17 for more details.

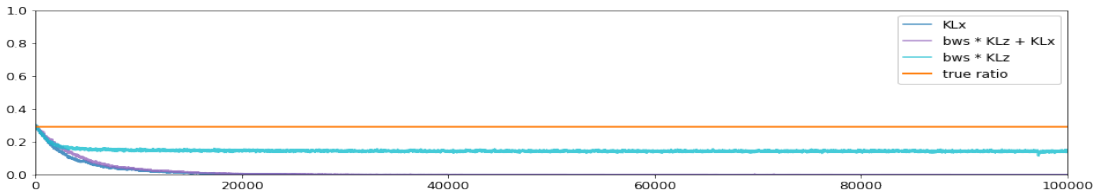
5.2 Exploring some directions leads to collapse in others

As a reminder, the objective of this work is to train bi-directional generative models in a *data-free* manner. Given that randomly initialized models typically produce absurd configurations, *data-dependent* pre-trainings are used to lower the U_B energies of generated samples and to ensure that the model does not completely miss large portions of the target distribution. But one has to assume that the quality of the pre-training is poor since in general the available data may be insufficient or even non-existent (as in a curriculum learning setup for example, see section 9.4).

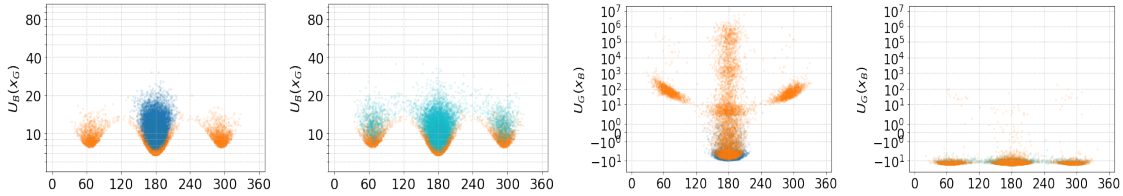
Fine-tuning behaviors, starting from poor pre-trainings, can be examined by changing the target distribution p_B between the two training phases, thereby pre-training the model to generate a distribution that is not exactly correct. This can be achieved in a controlled manner by tweaking the terms of the potential energy U_B.

Two such experiments, starting from the same pre-training from figure 17, are analyzed in this section, and a comprehensive hypothesis explaining all the observed results is presented in section 5.4.

Before the fine-tunings of figure 19 the factor λ_{align} (described in section 2.4) is decreased from 10 (at the end of the pre-training from figure 17) to 0. As a consequence, the total volume of p_B is effectively "widened" and many translations

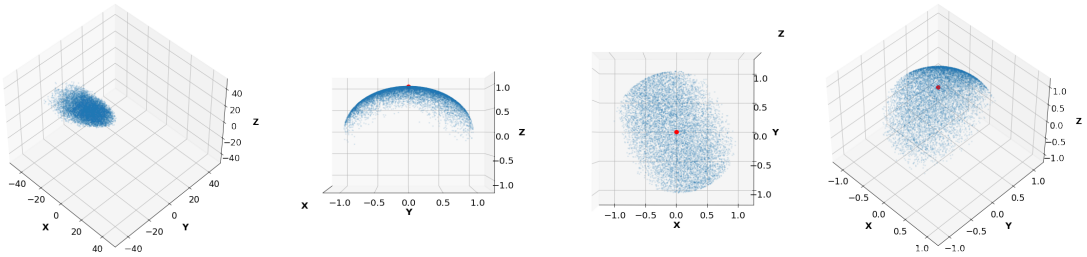


(a) Ratio of generated x_G samples in the minor modes during fine-tuning.



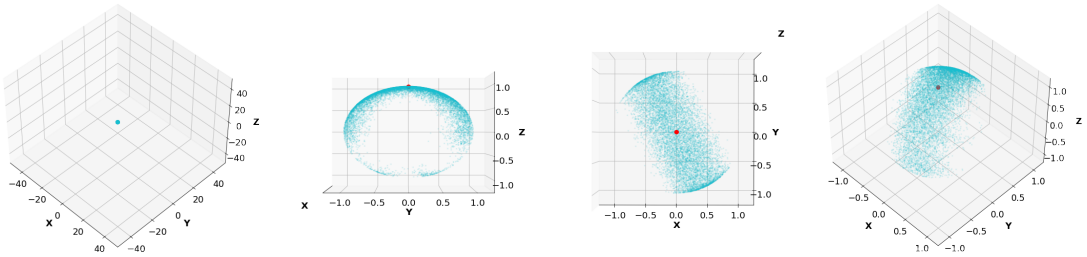
(b) U_B energies of samples $x_G \sim p_G$.

(c) U_G energies of samples $x_B \sim p_B$.



(d) x_G translations.

(e) x_G rotations under 3 different view points.



(f) x_G translations.

(g) x_G rotations under 3 different view points.

Figure 19: Comparison of three data-free fine-tunings on `Butane` with $\lambda_{\text{align}} = 0$.
- In blue: a fine-tuning with $\mathcal{L}_{KLx}$ alone.
- In purple: a fine-tuning with $\mathcal{L}^{\text{df}}_{KLz} + \mathcal{L}_{KLx}$.
- In cyan: a fine-tuning with $\mathcal{L}^{\text{df}}_{KLz}$ alone.
See caption of figure 17 for more details.

and rotations penalized during pre-training are now allowed:

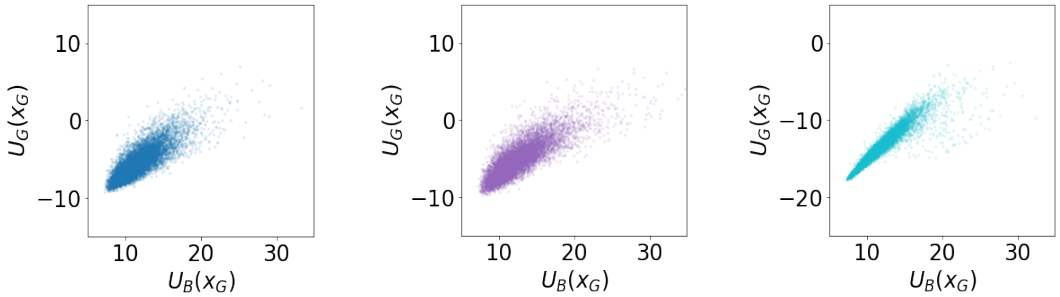
- Figure 19a shows how using $\mathcal{L}_{KLx}$ leads to immediate collapse whether or not it is used in conjunction with $\mathcal{L}^{\text{df}}_{KLz}$. This result is reminiscent of the fast collapse observed in figure 10 on `Double Well Wide 12D`. If $\mathcal{L}^{\text{df}}_{KLz}$ is used alone on the other hand, although the percentage of points in the minor modes decreases rapidly at first, it eventually stabilizes and no collapse occurs.
- Figure 19b shows that the collapse does not lead to an increase in the energy of generated samples $U_B(x_G)$ which means that $p_G \subset p_B$ remains true.[43]
- Conversely figure 19c demonstrates how the collapse induced by $\mathcal{L}_{KLx}$ im-

[43] One can also notice that in figure 19b that the energies of $U_B(x_B^{\text{MinH}})$ (in orange) are slightly lower than those of $U_B(x_G)$ (in blue and cyan). This can be explained by the fact that G^{H} is not perfect and may not generate hydrogen positions that strictly minimize U_B.

pacts all three modes (the minor modes are completely lost and the central mode also loses some of its mass). Surprisingly $p_B \subset p_G$ remains true when using $\mathcal{L}^{\text{df}}_{KLz}$ instead.

- Figure 19d shows how $\mathcal{L}_{KLx}$ leads to the exploration of the newly available translations through the explicit optimization of the local entropy term $-\log\left|\det\left(\frac{\partial G(z_{\mathcal{N},i})}{\partial z_{\mathcal{N},i}}\right)\right|$ and figure 19e indicates that the mass that is lost *within* the central mode corresponds to the rotations in the lower part of the sphere.
- In figure 19f, although $\mathcal{L}^{\text{df}}_{KLz}$ leads to a somewhat stable data-free fine-tuning (since neither the modes nor the rotations are lost as shown in figures 19c and 19g) it does not seem to promote exploration whatsoever. This lack of exploration can be explained by the fact that even though this loss increases the probability of under-sampled points, it does so only among those that are *already generated* by p_G, and does not encourage expansion explicitly.

It would be tempting to believe that the collapse mostly occurs on regions of the space that are the hardest to generate, and that by removing those regions, better correlations are obtained between U_B and U_G on generated samples. But this assumption is refuted by the results of figure 20, where clearly, the fine-tuning that did *not* collapse (with $\mathcal{L}^{\text{df}}_{KLz}$ in cyan) is the one with the best correlations. Said differently: the fine-tuning that gave the best correlations (with $\mathcal{L}^{\text{df}}_{KLz}$) did not need to collapse for that.



(a) $\mathcal{L}_{KLx}$ alone (b) $\mathcal{L}^{\text{df}}_{KLz} + \mathcal{L}_{KLx}$ (c) $\mathcal{L}^{\text{df}}_{KLz}$ alone

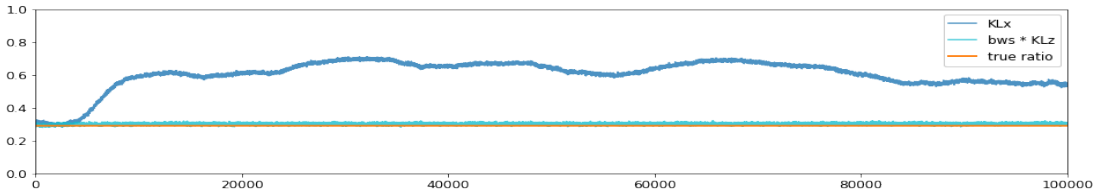
Figure 20: Correlations between U_B and U_G on generated samples $x_G \sim p_G$ after fine-tunings on `Butane` with $\lambda_{\text{align}} = 0$.

In figure 21, contrary to figure 19, λ_{align} is *increased* from 10 to 100, effectively "narrowing" the total volume of p_B. This means that at the beginning of the fine-tuning, the generated distribution p_G already covers the whole space, but also generates some additional, unwanted, configurations (namely the lower part of the sphere from figure 17e).

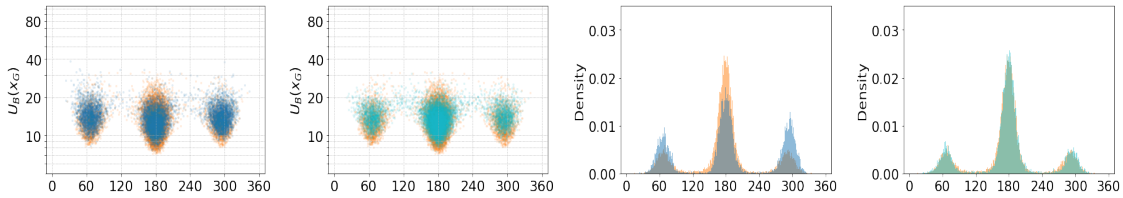
- Figures 21a and 21c show how $\mathcal{L}_{KLx}$ leads to unstable ratios between the modes, but no collapse. In fact, quite the opposite is observed, since the ratio between the modes shows *more* mass being assigned to the minor modes. $\mathcal{L}^{\text{df}}_{KLz}$, as usual, is remarkably stable.
- In figure 21b, the energies of generated configurations $U_B(x_G)$ are very low, thereby demonstrating that both losses manage to ensure that $p_G \subset p_B$ at

the end of the fine-tuning (by making the model lose the portions of the distribution it generated in excess at the start of the fine-tuning).

- Finally, figure 21e displays the impact of the higher $\lambda_{\text{align}} = 100$ on the shape of the distribution, and shows very constrained rotations as expected.

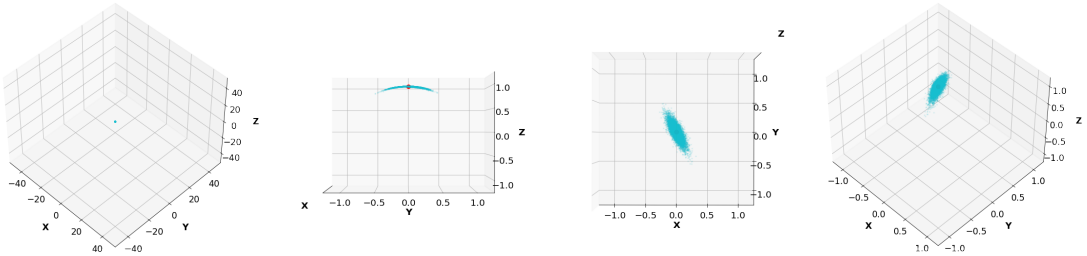


(a) Ratio of generated x_G samples in the minor modes during fine-tuning.



(b) U_B energies of samples $x_G \sim p_G$.

(c) p_G distribution of the dihedral angle ϕ.



(d) x_G translations.

(e) x_G rotations under 3 different view points.

Figure 21: Comparison of two data-free fine-tunings on `Butane` with $\lambda_{\text{align}} = 100$.
- In blue: a fine-tuning with $\mathcal{L}_{KLx}$ alone.
- In cyan: a fine-tuning with $\mathcal{L}^{\text{df}}_{KLz}$ alone.
See caption of figure 17 for more details.

A complete depiction of the collapse phenomenon and its hypothesized inner-workings are presented in section 5.4.

5.3 Managing the instability of batch weights

Although softmaxes are *numerically* stable, they are inherently "peaky" in the sense that they tend to put most of their mass on just a few points. As a consequence, most of the batch weights are very close to zero, which has an effect similar to reducing the size of the mini-batch. The only rare points on which the loss stays significant are where $\frac{p_B}{p_G}$ is high (i.e. where the model is under-sampling the most). This can be illustrated by noting that the maximum of a softmax computed over 1024 values distributed normally and with a standard deviation of 10, is above 76% on average:

$$\mathop{\mathbb{E}}_{\mathbf{w}\sim\mathcal{N}^{1024}(0,10)} \max\left(\{\text{softmax}(w_i)|i \in [1, 1024]\}\right) > 0.76 \tag{22}$$

If $-U_B + U_G$ were to have a standard deviation higher than 10 in equation 21a (as is often the case in practice), then more than 76% of the loss would be focused on a single point out of 1024.

One approach to address that problem is to adjust the batch weights from equation 21 by dividing the energy differences by their standard deviation (which essentially puts the "temperature" of the softmaxes in a more acceptable range):

$$\tilde{\tilde{w}}_i = \frac{-U_B(x_{G,i}) + U_G(x_{G,i})}{\max\left(1, \sigma\left(-U_B(\mathbf{x}_G) + U_G(\mathbf{x}_G)\right)\right)} \tag{23a}$$

$$\tilde{w}_i = \text{softmax}\left(\tfrac{1}{s_{\text{temp}}} \cdot \tilde{\tilde{w}}_i\right) \tag{23b}$$

$$w_i = [\tilde{w}_i \cdot n]^{\ddagger} \tag{23c}$$

with:

- $\sigma(v) = \sqrt{\frac{1}{n}\sum_i^n \left(v_i - \mu(v)\right)^2}$ the standard deviation of v over the mini-batch.
- $\mu(v) = \frac{1}{n}\sum_i^n v_i$ the expected value (the average) of v over the mini-batch.
- s_{temp} the enforced "temperature" of the softmax.

Such batch weights are much less peaky, but incorrect everywhere since they are not proportional to the importance sampling weights from equation 19d. A second approach is to *clamp* the batch weights in the following fashion:

$$\tilde{\tilde{w}}_i = \text{softmax}\left(-U_B(x_{G,i}) + U_G(x_{G,i})\right) \tag{24a}$$

$$\tilde{w}_i = \text{clamp}\left(\tilde{\tilde{w}}_i, 0.1, 10\right) \tag{24b}$$

$$w_i = \left[\tilde{w}_i \cdot \frac{n}{\sum_j^n \tilde{w}_j}\right]^{\ddagger} \tag{24c}$$

$$\text{with } \text{clamp}(x, c_{\min}, c_{\max}) = \begin{cases} c_{\min} & \text{if } x < c_{\min} \\ x & \text{if } c_{\min} \leqslant x < c_{\max} \\ c_{\max} & \text{if } x \leqslant c_{\max} \end{cases}$$

This is better, since only clamped points have incorrect weights but such clamping still impacts negatively the quality of experimental results.[44]

A third (and better) option is to use very low learning rates.[45][46] Peaky batch weights are similar to applying the loss to just a few points of the mini-batch, which is similar to sampling much smaller mini-batches on whatever region the batch weights favor. And just like small mini-batches need smaller learning rates, losses with peaky batch weights have the same requirement.

[44] The choice of $c_{\min} = 0.1$ and $c_{\max} = 10$ means that each point weighs between 0.1 and 10 times the uniform weights.

[45] Using a learning rate 10 times smaller implies having to train for roughly 10 times longer.

[46] Small learning rates also address other instabilities (not just the one induced by the use of batch weights).

In the context of `Butane`, the learning rate is lowered from 10^{-4} to 10^{-5} during the fine-tuning. This ensures that, if the softmax is too peaky and only a small region of the space receives a non-negligible loss, then the effect of one weight update is not too damaging to the rest of p_G before the softmax of the next mini-batch focuses on another point. The effect being leveraged here is a form of time-averaging.

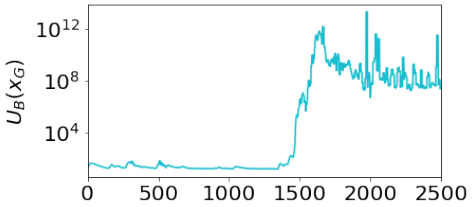


Figure 22: U_B energies of a fine-tuning on `Butane` with $\mathcal{L}_{KLz}^{\text{df}}$ and a learning rate of 10^{-4}. This is to be compared with figure 18a, which uses a learning rate of 10^{-5}.

To appreciate how important the choice of the learning rate is, one only needs to compare the fine-tunings of figure 18a (which is completely stable at a learning rate of 10^-5) and figure 22 (which produces absurd configurations in less than 2000 iterations at a learning rate of 10^{-4}). Both use $\mathcal{L}_{KLz}^{\text{df}}$ and are perfectly identical except for learning rate.

5.4 How to minimize the risk of collapse

$\mathcal{L}_{KLx}$ is a combination of two terms: the energy term $\beta U_B(G(z_{\mathcal{N}}))$ and the entropy term $-\log\left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|$. Combined, they interact to ensure that the distribution p_G generated by the model converges to the target distribution p_B. But this is true only under several assumptions that cannot be verified. In particular, the true expectation from equation 15 cannot be computed, only estimated, and the model is limited in expressivity. Given the results of section 5.2, it is clear that at least one assumption made along the way breaks the promise of perfect convergence.

In figure 19, setting λ_{align} to zero is equivalent to removing the energy term in *some* directions of the space (the directions of translations and rotations). As a consequence, the entropy term is now free to expand p_G there, but it does so at the expense of letting the distribution contract in other directions, leading to collapse. It is tempting to believe that it is the energy term that causes the contraction in the multimodal direction by overpowering the entropy term (which is suddenly busy expanding the distribution somewhere else). But, although it is possible that the energy term accelerates the collapse, it is not necessary for it to occur. This is illustrated in figure 19e, where the rotations represented at the bottom of the sphere are completely lost, regardless of the fact that no energy term encourages this (since $\lambda_{\text{align}} = 0$).

In figure 21, still with $\mathcal{L}_{KLx}$, the phenomenon is reversed. The energy term is suddenly increased in some directions, by setting λ_{align} to 100, and the model quickly contracts in those directions, but at the expense of spreading more uniformly in other directions, resulting in the loss of the proper balance between the modes (see figures 21a and 21c).

Since the overall mass of p_G remains constant (it always sums to one), what seems to be happening when using $\mathcal{L}_{KLx}$ is the following:

- When having to *expand* in some directions, p_G must find the mass somewhere, and it tends to contract in other directions rather that decrease its mass uniformly (see diagram of figure 23a, and experiments of figure 19).
- When having to *contract* in some directions, p_G must redistribute that excess mass uniformly, but instead tends to increase unevenly in other directions (see diagram of figure 23b, and experiments of figure 21).

(a) If expansion comes first (in black), it may be simpler for p_G to contract in other directions (in gray), rather than to lose mass uniformly everywhere (as in figure 19).

(b) If contraction comes first (in black), it may be simpler for p_G to expand in other directions (in gray), rather than to gain mass uniformly everywhere (as in figure 21).

Figure 23: Diagram summarizing the *observed* behavior of $\mathcal{L}_{KLx}$. Note that this merely an interpretation of what is observed experimentally and not proven mathematically.

In retrospect, this phenomenon is not too surprising since $\mathcal{L}_{KLx}$ does not explicitly move generated x_G points from the over-sampled regions toward the under-sampled regions. In other words, the loss is not applied on the generated points themselves, but on their probability distributions instead, which is considerably less informative, and becomes a significant problem when only a finite number of points is sampled within each mini-batch.

According to the change of variable formula from equation 11, p_G can be expressed as follows:

$$p_G(x_G) = p_G(G(z_\mathcal{N})) = q_\mathcal{N}(G(z_\mathcal{N})) \cdot \left|\det\left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}}\right)\right|^{-1} \tag{25}$$

$\mathcal{L}_{KLx}$ pushes p_G to increase where it is under-sampled by maximizing the second term of equation 25: $\left|\det\left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}}\right)\right|^{-1}$.[47] To increase that term near a given point $x_G = G(z_\mathcal{N})$, the model G brings the image of points that are near $z_\mathcal{N}$ closer to x_G, thereby "stealing" them from other x_G points. This effect is local and does not guarantee that the other x_G points are remapped correctly if they are not sampled in the next batch.

[47] $q_\mathcal{N}(G(z_\mathcal{N}))$ is optimized only indirectly.

Alternative losses are presented in sections 6 to 8 that aim to address that issue, but for now, here is a summary of the list of practical training considerations that alleviate the problem (including the risk of collapse):

- Trainings with $\mathcal{L}^{\text{df}}_{KLz}$ are more stable than trainings with $\mathcal{L}_{KLx}$. However, $\mathcal{L}^{\text{df}}_{KLz}$ can only be used if exploration is not a significant requirement. This is because it merely redistributes p_G among points that are already probable according to it. It does not expand in new directions (as illustrated in figure 19).
- Removing (or constraining) degrees of freedom that are unnecessary for downstream tasks in paramount. For example in this work, hydrogen permutations are removed altogether (see section 3.3) and translations and rotations are constrained with the alignment penalty $U_{\text{align}}(x)$. For the later point, many other possibilities exist, including using model architectures that are invariant to such translations and rotations (see in section 2.4).
- If data is available, a better pre-training quality always helps (see section 4.5). This point will surely become critical in the context of curriculum learning (see section 9.4).
- The data should be standardized properly. Although this is a very general statement, it is especially true in the context of generative models. This means that not only $q_{\mathcal{N}}$ should have a variance close to 1, but p_B too.[48] The encoding of x_B points should have a reasonable scale and the energy function U_B should be adapted accordingly.[49]
- Increasing the temperature simplifies the task that the model has to solve (see section 4.5) since the target distribution p_B is slightly more uniform. However, the training temperature should not be too high[50], so as to ensure that re-balancing the distribution remains numerically stable (see figure 16).
- Low learning rates are usually preferable, particularily when using batch weights with $\mathcal{L}^{\text{df}}_{KLz}$, even at the cost of longer trainings.

Although these important precautions significantly improve the quality of trainings, they do not fix the underlying issue. It is unavoidable that better losses are required (see the following sections).

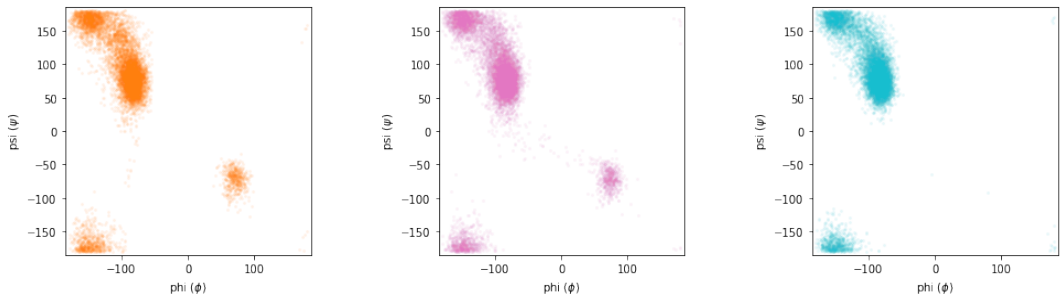
Further experiments on the `Dialanine` dataset confirm that $\mathcal{L}^{\text{df}}_{KLz}$ remains stable on somewhat larger systems (see figures 24 and 25), although it seems that very small modes are still at risk of being lost if not trained at a high temperature. Between the pre-training (figure 24b) and the fine-tuning (figure 24c), the mode in the lower left quadrant of the Ramachandran plot is completely lost, and as a result the energy of generation of those samples explodes (figure 25c).

$\mathcal{L}^{\text{df}}_{KLz}$ also manages to adapt the distribution to a change of thickness when λ_{align} is either increased or decreased by a factor 10 as illustrated in figure 26.

[48] If possible in every direction of space, although this is hardly feasible in the context of molecular datasets.

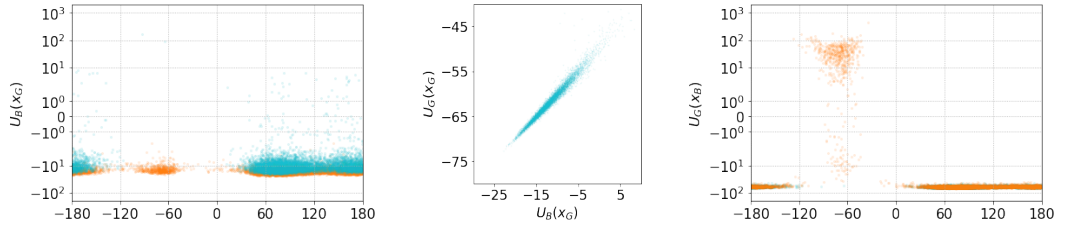
[49] It would be interesting to know whether invariance to homotheties is a desirable feature.

[50] Roughly, no more than 4 times the baseline temperature.



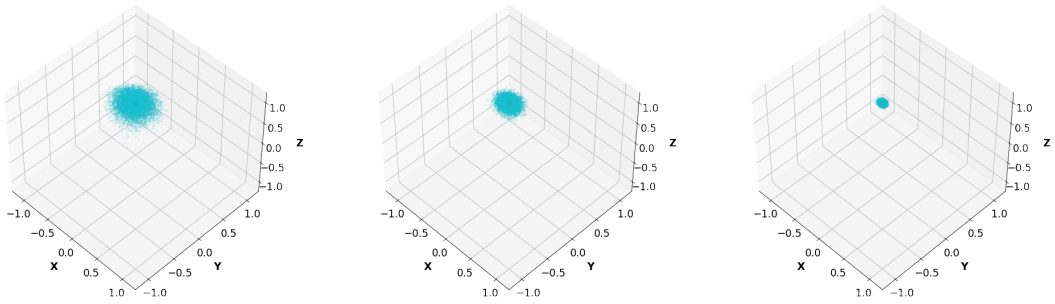
(a) ϕ and ψ from the `Dialanine` dataset. (b) ϕ and ψ after pre-training with $\mathcal{L}_{KLz}^{\text{dd}}$. (c) ϕ and ψ after fine-tuning with $\mathcal{L}_{KLz}^{\text{df}}$

Figure 24: Dihedral angles ϕ and ψ of the dialanine molecule.



(a) U_B energies of samples $x_G \sim p_G$. (b) Correlations. (c) U_G energies of samples $x_B \sim p_B$.

Figure 25: Quality of generation after fine-tuning on `Dialanine` with $\mathcal{L}_{KLz}^{\text{df}}$.



(a) Fine-tunning with $\lambda_{\text{align}} = 1$. (b) Fine-tunning with $\lambda_{\text{align}} = 10$. (c) Fine-tunning with $\lambda_{\text{align}} = 100$.

Figure 26: Results of a fine-tuning on `Dialanine` with $\mathcal{L}_{KLz}^{\text{df}}$ after a pre-training with $\lambda_{\text{align}} = 10$.

$\mathcal{L}_{KLx}$, on the other hand, completely fails to maintain the p_G distribution stable[51], even after lowering the learning rate a second time to 10^{-6}. The potential energy of generated samples gets progressively worse, until it becomes off by several orders of magnitude (data not shown).

5.5 Re-weighting $KL(p_G||p_B)$

When training with $\mathcal{L}_{KLx}$ on a dataset like `Double Well Wide 12D`, the model always collapses to the *same* well (the larger one as in figure 10). Since the conclusions from section 4.3 suggest that the collapse may (at least in part) be caused by the sampling procedure itself, reweighting the gradients between the elements

[51] As foreshadowed in section 5.2

of the mini-batch seems to be a promising approach. The idea here is to focus the loss differently between the sampled points so as to imitate a different sampling procedure. It is very similar to the classic method of scaling weight updates used to address class imbalance in classification problems [32]. The loss becomes:

$$\mathcal{L}^{\text{bw}}_{KLx}(\mathbf{z}_{\mathcal{N}}) = \sum_{i=1}^{n} \frac{1}{n} \cdot \left[w_i \cdot \left(\beta U_B(G(z_{\mathcal{N},i})) - \log \left| \det \left(\frac{\partial G(z_{\mathcal{N},i})}{\partial z_{\mathcal{N},i}} \right) \right| \right) \right] \tag{26}$$

with $\mathbf{z}_{\mathcal{N}} \sim q_{\mathcal{N}}^n$ a mini-batch of size n, and $z_{\mathcal{N},i} \sim q_{\mathcal{N}}$ its i$^{\text{th}}$ element.

This loss is not an unbiased estimator of the KL divergence, so optimizing it is expected to lead to a biased p_G, however a moderate bias in p_G is acceptable in practice, thanks to the reweighting approach used in applications (see section 3.1).

The efficiency of this approach can be demonstrated by trying to collapse on the smaller well. This can be done very simply by using weights w_i of the form:

$$\tilde{w}_i = \begin{cases} 1 & \text{if } x_{G,i} \text{ is in the minor mode} \\ 0.1 & \text{if } x_{G,i} \text{ is in the major mode} \end{cases} \tag{27a}$$

$$w_i = \tilde{w}_i \cdot \frac{n}{\sum_j^n \tilde{w}_j} \tag{27b}$$

The results of figure 27 show that even if some weight is still attributed to points from the major mode, the model ends up collapsing completely.

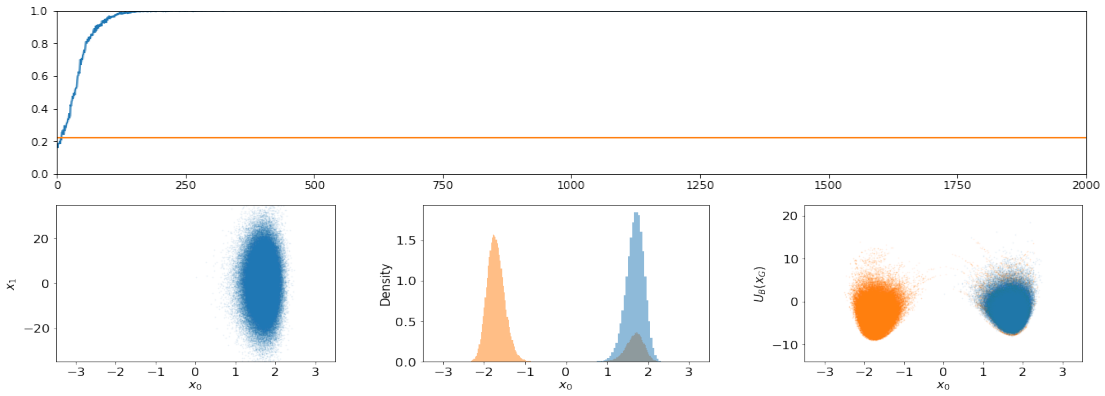


Figure 27: Fine-tuning with $\mathcal{L}^{\text{bw}}_{KLx}$ on `Double Well Wide 12D` with batch weights focusing the loss on the minor mode (see equation 27) (see caption of figure 7 for more details).

Now that it is clear that batch weights can control which well is favored during collapse, it can be shown that there exist batch weights that prevent mode collapse altogether. This can be achieved by leveraging information coming from our knowledge of p_B: the target ratio between the wells.

Roughly, in `Butane`'s dataset, 71.0% of the points are in the major mode, and the remaining 29.0% are split between the two minor modes in equal proportions

(14.5% each). These target ratios are denoted as follows, with the arrows $\leftarrow$, $\uparrow$ and $\rightarrow$ corresponding the "left", "middle" and "right" modes respectively:

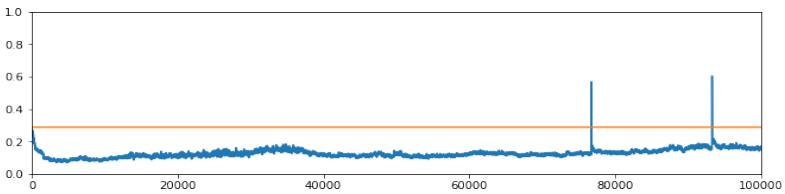
$$r^{\leftarrow}_{\text{trg}} \approx .145 \qquad r^{\uparrow}_{\text{trg}} \approx .710 \qquad r^{\rightarrow}_{\text{trg}} \approx .145 \tag{28}$$

During training, proper ratio between the wells can be enforced by using batch weight of the form:

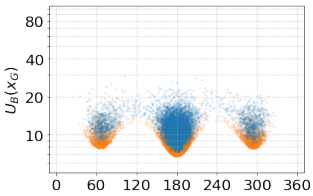
$$\tilde{w}_i = \begin{cases} r^{\leftarrow}_{\text{trg}} / r^{\leftarrow}_{\text{out}} & \text{if dihedral}(x_i) < 120° \\ r^{\uparrow}_{\text{trg}} / r^{\uparrow}_{\text{out}} & \text{if } 120° \leqslant \text{dihedral}(x_i) \leqslant 240° \\ r^{\rightarrow}_{\text{trg}} / r^{\rightarrow}_{\text{out}} & \text{if } 240° < \text{dihedral}(x_i) \end{cases} \tag{29a}$$

$$w_i = \left[\tilde{w}_i \cdot \frac{n}{\sum_j^n \tilde{w}_j} \right]^{\ddagger} \tag{29b}$$

with $r^{\leftarrow}_{\text{out}}$, $r^{\uparrow}_{\text{out}}$, and $r^{\rightarrow}_{\text{out}}$ being the ratios of output points x_G, in each mini-batch, that are in the "left", "middle" and "right" modes respectively. This ensures that the less populated a well is, the larger the batch weights become. As a result, the training is somewhat stabilized (see figure 28).



(a) Ratio of generated $x_G \sim p_G$ points that are in the minor modes during training (in blue) compared with the real ratio of ≈ 0.29 from the dataset (in orange).



(b) U_B energies computed on generated samples $x_G \sim p_G$.

Figure 28: Fine-tuning with $\mathcal{L}^{\text{bw}}_{KLx}$ on `Butane` with batch weights from equation 29.

Despite this relative improvement the correct ratios are not maintained during fine-tuning. This problem could be somewhat alleviated by using more adaptive batch weights. Since $\tilde{w}_i$ (from equation 29a) can only have three possible value at each time step, simply increasing $\tilde{w}_i$ gradually each time a well is undersampled and decreasing it each time it is oversampled should be sufficient:

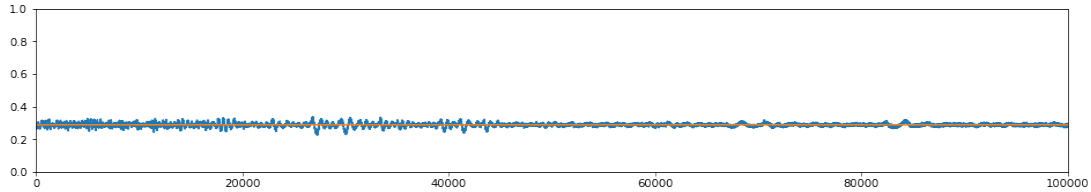
$$\tilde{\tilde{w}}^{\leftarrow\uparrow\rightarrow}_{i,t} = \begin{cases} \tilde{\tilde{w}}^{\leftarrow\uparrow\rightarrow}_{i,t-1} \times 1.01 & \text{if } r^{\leftarrow\uparrow\rightarrow}_{\text{out}} < r^{\leftarrow\uparrow\rightarrow}_{\text{trg}} \\ \tilde{\tilde{w}}^{\leftarrow\uparrow\rightarrow}_{i,t-1} \div 1.01 & \text{if } r^{\leftarrow\uparrow\rightarrow}_{\text{out}} \geqslant r^{\leftarrow\uparrow\rightarrow}_{\text{trg}} \end{cases} \tag{30a}$$

$$\tilde{w}_i = \begin{cases} \tilde{\tilde{w}}^{\leftarrow}_{i,t} & \text{if dihedral}(x_i) < 120° \\ \tilde{\tilde{w}}^{\uparrow}_{i,t} & \text{if } 120° \leqslant \text{dihedral}(x_i) \leqslant 240° \\ \tilde{\tilde{w}}^{\rightarrow}_{i,t} & \text{if } 240° < \text{dihedral}(x_i) \end{cases} \tag{30b}$$

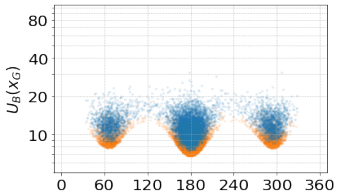
$$w_i = \left[\tilde{w}_i \cdot \frac{n}{\sum_j^n \tilde{w}_j} \right]^{\ddagger} \tag{30c}$$

with:

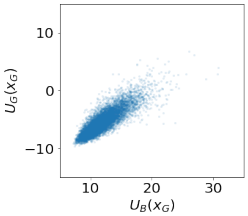
- t: the fine-tuning time step.
- $\tilde{w}_{i,t}^{\leftarrow\uparrow\rightarrow}$: three values corresponding to each mode and initialized at 1.[52]



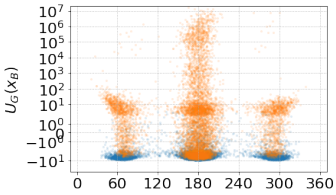
(a) Ratio of generated $x_G \sim p_G$ points that are in the minor modes during training (in blue) compared with the real ratio of ≈ 0.29 from the dataset (in orange).



(b) U_B energies computed on generated samples $x_G \sim p_G$.



(c) Energy correlations.



(d) U_G energies computed on simulated samples $x_B \sim p_B$.

Figure 29: Fine-tuning with $\mathcal{L}_{KLx}^{\text{bw}}$ on `Butane` with batch weights from equation 30.

This time, the ratios are perfectly maintained (see figure 29a) but this does not stop the distribution from collapsing *within* each mode (see figure 29d).

A similar method, based on this approach, would consist in replacing the estimation of how much a mode is over- or under-sampled in equation 29a, with some other estimate, this time computed in a *neighborhood* and without expert information about what the correct ratios should be. One of the most rudimentary example of this is:

$$x_{G,i}^{\text{low}} = \texttt{minimize\_energies}\left(x_{G,i}\right) \tag{31a}$$

$$p_{B,i}^{\text{agg}} = \texttt{aggregate}\left(\text{softmax}\left(-U_B(x_{G,i}^{\text{low}})\right), x_{G,i}^{\text{low}}\right) \tag{31b}$$

$$p_{G,i}^{\text{agg}} = \texttt{aggregate}\left(\text{softmax}\left(-U_G(x_{G,i}^{\text{low}})\right), x_{G,i}^{\text{low}}\right) \tag{31c}$$

$$\tilde{w}_i = \frac{p_{B,i}^{\text{agg}}}{p_{G,i}^{\text{agg}}} \tag{31d}$$

$$w_i = \left[\tilde{w}_i \cdot \frac{n}{\sum_j^n \tilde{w}_j}\right]^{\ddagger} \tag{31e}$$

with:

- `minimize_energies`(x): a function that outputs the last x obtained from a few steps of minimization of the potential energy U_B.
- $U_G(x_{G,i}^{\text{low}})$: the energy of generation computed with an inverse pass through the model.

[52] Note that when using the batch weights of equation 29b the mini-batch size must be large enough (say 1024) to ensure that there is always at least one point sampled in each mode to avoid a division by zero. This requirement is dropped here in equation 30c.

- `aggregate`(v_i, x_i): a function that averages the value v_i in the neighborhood of x_i (which requires some measure of similarity between $x^{\text{low}}_{G,i}$ points).

Experimental results based on variation of $\mathcal{L}^{\text{bw}}_{KLx}$ are generally underwhelming, with or without an `aggregate` function, based or not on energy minimization and using different distance metrics (data not shown), and do not successfully counteract the collapse without expert information. That is not to say that designing better batch weights for $\mathcal{L}^{\text{bw}}_{KLx}$ may not be a promising avenue of research. But significant difficulties have to be overcome to pursue this strategy:

- The most fundamental problem is that batch weights would be used to counteract a phenomenon that is not well measured. Although it seems possible in practice (as the result from figure 29 suggests), without a good quantification of the collapse phenomenon, it seems more likely that batch weights can only *approximately* counteract it. In other words, it is unclear what batch weights should actually do and what quantity they should help minimize. Furthermore, it is unclear whether averaging quantities of the form $\tilde{p}_B/p_G$ (as in equations 29a and 31d) is sufficient or not since some completely different type of information may be required.
- Even if one accepts the imprecise nature of $\mathcal{L}^{\text{bw}}_{KLx}$ and decides to use batch weights similar to those of equation 31, the clustering method alone remains a substantial complication. It heavily relies on a good distance metric in x-space that represents the difference between molecular configurations, but given such a metric, better methods may be available (see section 8).

In conclusion, using $\mathcal{L}^{\text{bw}}_{KLx}$ seems tricky. $\mathcal{L}^{\text{df}}_{KLz}$ on the other hand, is both theoretically justified and leads to convincing experimental results as shown in sections 5.1 and 5.2.

6 Using Renyi divergences and L^2 losses

6.1 Using $KL(p_B||p_G)$

In section 4.1, the choice has been made to use $KL(p_G||p_B)$ as a training objective, but it is a well known result from the literature[53] that when the KL is not weighted by the target distribution (i.e. p_B), it can under-estimate its support.[54] $KL(p_B||p_G)$ on the other hand tends to over-estimate the support of the target distribution p_B[55] which could be a valuable property to avoid mode collapse on problems more complex than `Butane`.

There are two main ways to derive a loss function from the $KL(p_B||p_G)$ objective. The first one consists in noticing that $KL(p_B||p_G)$ is actually equivalent to $KL(q_F||q_{\mathcal{N}})$[56] (see Papamakarios et al. [40]):

$$
\begin{aligned}
KL(p_B||p_G) &= \int p_B(x)\Big[\log p_B(x) - \log p_G(x)\Big]\ dx && \text{(32a)}\\
&= \int p_B(x)\left[\log p_B(x) - \log\left|\det\left(\frac{\partial G^{-1}(x)}{\partial x}\right)\right| - \log q_{\mathcal{N}}(G^{-1}(x))\right]\ dx && \text{(32b)}\\
&= \int q_F(z)\left[\log q_F(G(z)) + \log\left|\det\left(\frac{\partial F^{-1}(z)}{\partial z}\right)\right| - \log q_{\mathcal{N}}(z)\right]\ dz && \text{(32c)}\\
&= \int q_F(z)\left[\log q_F(z) - \log q_{\mathcal{N}}(z)\right]\ dz && \text{(32d)}\\
&= KL(q_F||q_{\mathcal{N}}) && \text{(32e)}
\end{aligned}
$$

with:

- (32a) by definition of the KL divergence
- (32b, 32c and 32d) by using the change of variable formula
- (32e) by definition of the KL divergence

The $\mathcal{L}^{\text{dd}}_{KLz}$ loss and its data-free variant $\mathcal{L}^{\text{df}}_{KLz}$ are therefore already minimizing this reverse KL which may explain (at least in part) their good performance. One can also notice that during training $\mathcal{L}^{\text{df}}_{KLz}$ only differentiates the *inverse pass* $F = G^{-1}$ of the generator (i.e. it moves the points $\mathbf{z}_F$ for a given mini-batch $\mathbf{x}_B$), whereas $\mathcal{L}_{KLx}$ only differentiates the *forward pass* $G = F^{-1}$ of the generator (i.e. it moves the points $\mathbf{x}_G$ for a given mini-batch $\mathbf{z}_{\mathcal{N}}$).

So a second way to derive a loss function from $KL(p_B||p_G) = KL(q_F||q_{\mathcal{N}})$ would

[53] See Murphy [38], chapter 21, figures 21.1 and 21.2.

[54] $KL(p_G||p_B)$ is sometimes referred to as the "exclusive" KL and is said to be "zero-forcing".

[55] $KL(p_B||p_G)$ is sometimes referred to as the "inclusive" KL and is said to be "zero-avoiding".

[56] Note that the same method can be used to show the equivalence between $KL(p_G||p_B)$ and $KL(q_{\mathcal{N}}||q_F)$.

consist in trying to differentiate the forward pass instead of the inverse pass:

$$KL(p_B||p_G) = \int p_B(x) \log \frac{p_B(x)}{p_G(x)} \, dx \tag{33a}$$

$$= -\int p_B(x) \log \mathcal{Z}_B \, dx + \frac{1}{\mathcal{Z}_B} \int \tilde{p}_B(x) \log \frac{\tilde{p}_B(x)}{p_G(x)} \, dx \tag{33b}$$

$$= -\log \mathcal{Z}_B + \frac{1}{\mathcal{Z}_B} \int \tilde{p}_B(x) \log \frac{\tilde{p}_B(x)}{p_G(x)} \, dx \tag{33c}$$

$$= -\log \mathcal{Z}_B + \frac{1}{\mathcal{Z}_B} \int p_G(x) \frac{\tilde{p}_B(x)}{p_G(x)} \log \frac{\tilde{p}_B(x)}{p_G(x)} \, dx \tag{33d}$$

$$= -\log \mathcal{Z}_B + \frac{1}{\mathcal{Z}_B} \int q_{\mathcal{N}}(z) \frac{\tilde{p}_B(G(z))}{q_{\mathcal{N}}(z) \left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}} \log \left(\frac{\tilde{p}_B(G(z))}{q_{\mathcal{N}}(z) \left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}} \right) dz \tag{33e}$$

$$= -\log \mathcal{Z}_B + \frac{1}{\mathcal{Z}_B} \int q_{\mathcal{N}}(z) \frac{e^{-\beta U_B(G(z))}}{q_{\mathcal{N}}(z) \left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}} \log \left(\frac{e^{-\beta U_B(G(z))}}{q_{\mathcal{N}}(z) \left|\det \left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}} \right) dz \tag{33f}$$

$$= -\log \mathcal{Z}_B + \frac{1}{\mathcal{Z}_B} \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\frac{e^{-\beta U_B(G(z_{\mathcal{N}}))}}{q_{\mathcal{N}}(z_{\mathcal{N}}) \left|\det \left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|^{-1}} \log \left(\frac{e^{-\beta U_B(G(z_{\mathcal{N}}))}}{q_{\mathcal{N}}(z_{\mathcal{N}}) \left|\det \left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|^{-1}} \right) \right] \tag{33g}$$

$$= -\log \mathcal{Z}_B - \frac{1}{\mathcal{Z}_B} \mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[\frac{e^{-\beta U_B(G(z_{\mathcal{N}}))}}{q_{\mathcal{N}}(z_{\mathcal{N}}) \left|\det \left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|^{-1}} \right.$$
$$\left. \times \left(\beta U_B(G(z_{\mathcal{N}})) + \log \left(q_{\mathcal{N}}(z_{\mathcal{N}}) \left|\det \left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|^{-1} \right) \right) \right] \tag{33h}$$

with:

- (33a) by definition of the KL divergence
- (33b) by using $p_B = \tilde{p}_B / \mathcal{Z}_B$
- (33c) by using $\int p_B(x) dx = 1$ (probabilities sum to one)
- (33d) by multiplying by p_G / p_G
- (33e) by using the change of variable formula (from equation 11)
- (33f) by using: $\tilde{p}_B(x) = e^{-\beta U_B(x)}$
- (33g) by definition of expectations: $\mathop{\mathbb{E}}_{z_{\mathcal{N}} \sim q_{\mathcal{N}}} \left[f(z_{\mathcal{N}}) \right] = \int q_{\mathcal{N}}(z) f(z) dz$

Although it *seems* that the unknown $\mathcal{Z}_B$ has successfully been factored out of the expectation, it is not possible to transform equation 33g into a loss suitable for training, for two different reasons:

- The main reason (that we shall call the "additive $\mathcal{Z}_B$ problem") refers to the fact that the first fraction of equation 33g is multiplied by a quantity

which can be either positive or negative depending on an arbitrary additive constant in the energy U_B. This means that factoring out $\mathcal{Z}_B$ between equations 33a and 33b actually has a profound impact on the optimization. For illustration, equation 33f is analyzed here:

- If a large constant is added within the log then the log is positive and the U_B term from the first fraction is always *maximized* during training. Although it can make sense mathematically in the continous limit (and under the constraint that the probabilities must sum to 1), it is extremely detrimental to the optimization process when sampling mini-batches. It results in markedly unstable trainings since the model is being asked to increase the energy of the generated samples *as quickly as possible*.
- On the other hand, if a large constant is subtracted within the log then the log is negative and U_B from the first fraction is always *minimized* during optimization, which $\mathcal{L}_{KLx}$ already does.
- Only a careful balance between these two extremes would truly minimize $KL(p_B||p_G)$, and this balance is achieved when $\mathcal{Z}_B$ is *not* factored out of the log (see equation 33g). As a result, the log would then be either positive or negative, depending on whether points are over- or under-sampled.

- The second reason (that we shall call the "multiplicative $\mathcal{Z}_B$ problem") refers to the fact that the unknown fraction $1/\mathcal{Z}_B$ before the expectation of equation 33g makes it impossible to perfectly balance $KL(p_B||p_G)$ and $KL(p_G||p_B)$ to get a "symmetrical" training objective (in which p_G and p_B can be swapped freely).

To address these issues, Rényi divergences are investigated in section 6.2 and L^2 losses in section 6.3.

6.2 Using $RN_{1/2}(p_G||p_B)$ as a way to symmetrize the objective

KL divergences are a special case of a broader family of divergences called Rényi Divergences[57] which depend on an hyperparameter α, called the *order*, where $\alpha \geq 0$ and $\alpha \neq 1$:

$$RN_\alpha(p_G||p_B) = \frac{1}{\alpha - 1} \log \int \frac{p_G(x)^\alpha}{p_B(x)^{\alpha-1}} dx \tag{34}$$

The limit of $\alpha \to 1$ gives the KL divergence:

$$RN_1(p_G||p_B) = \int p_G(x) \log \frac{p_G(x)}{p_B(x)} \, dx = KL(p_G||p_B) \tag{35}$$

Instead of trying to balance $KL(p_G||p_B)$ and $KL(p_B||p_G)$, one can also use the

[57] See van Erven and Harremoes [54] for an overview as well as Hernandez-Lobato et al. [20] and Li and Turner [34] for applications to Variational Inference.

symmetrical Rényi divergence[58], with $\alpha = 1/2$:

$$RN_{1/2}(p_G, p_B) = RN_{1/2}(p_G||p_B) = -2\log \int \sqrt{p_B(x)p_G(x)}\ dx \tag{36}$$

The associated loss function is obtained like this:

$$\begin{aligned}
RN_{1/2}(p_G, p_B) &= -2\log \int \sqrt{p_B(x)p_G(x)}\ dx && \text{(37a)}\\
&= \log \mathcal{Z}_B - 2\log \int \sqrt{\tilde{p}_B(x)p_G(x)}\ dx && \text{(37b)}\\
&= \log \mathcal{Z}_B - 2\log \int p_G(x) \left(\frac{\tilde{p}_B(x)}{p_G(x)}\right)^{\frac{1}{2}}\ dx && \text{(37c)}\\
&= \log \mathcal{Z}_B - 2\log \int q_{\mathcal{N}}(z) \left(\frac{\tilde{p}_B(G(z))}{q_{\mathcal{N}}(z)\left|\det\left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}\right)^{\frac{1}{2}}\ dz && \text{(37d)}\\
&= \log \mathcal{Z}_B - 2\log \int q_{\mathcal{N}}(z) \left(\frac{e^{-\beta U_B(G(z))}}{q_{\mathcal{N}}(z)\left|\det\left(\frac{\partial G(z)}{\partial z}\right)\right|^{-1}}\right)^{\frac{1}{2}}\ dz && \text{(37e)}\\
&= \log \mathcal{Z}_B - 2\log \mathop{\mathbb{E}}_{z_{\mathcal{N}}\sim q_{\mathcal{N}}} \left[\left(\frac{e^{-\beta U_B(G(z_{\mathcal{N}}))}}{q_{\mathcal{N}}(z_{\mathcal{N}})\left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|^{-1}}\right)^{\frac{1}{2}}\right] && \text{(37f)}
\end{aligned}$$

with:

- (37a) by definition of the Rényi divergence with $\alpha = 1/2$
- (37b) by using $p_B = \tilde{p}_B/\mathcal{Z}_B$ and $\int p_B(x)dx = 1$ (probabilities sum to one)
- (37c) by multiplying by p_G/p_G
- (37d) by using the change of variable formula (from equation 11)
- (37e) by using: $\tilde{p}_B(x) = e^{-\beta U_B(x)}$
- (37f) by definition of expectations: $\mathop{\mathbb{E}}_{z_{\mathcal{N}}\sim q_{\mathcal{N}}}\left[f(z_{\mathcal{N}})\right] = \int q_{\mathcal{N}}(z)f(z)dz$

Note that in equation 37f the expectation is trapped within a log which means that the principle of Stochastic Gradient Descent cannot be applied directly to transform it into a loss function (see section 4.1). Although using the log of an expectation as an empirical loss introduces a bias in the resulting stochastic gradients, that bias is almost negligible [20], especially when sampling large mini-batches. Under this

[58]Divergences are not symmetrical in the general case, i.e. $D(p_G||p_B) \neq D(p_B||p_G)$ in most cases. The divergence $RN_{1/2}(p_G, p_B)$ is a notable exception, hence the use of a comma instead of two vertical bars to separate the two distributions.

approximation, one can derive the following loss function[59]:

$$\mathcal{L}_{RN_{1/2}}(\mathbf{z}_{\mathcal{N}}) = -\log \sum_{i=1}^{n} \left[\frac{1}{n} \cdot \left(\frac{e^{-\beta U_B(G(z_{\mathcal{N},i}))}}{q_{\mathcal{N}}(z_{\mathcal{N},i}) \left| \det \left(\frac{\partial G(z_{\mathcal{N},i})}{\partial z_{\mathcal{N},i}} \right) \right|^{-1}} \right)^{\frac{1}{2}} \right] \tag{38}$$

Symmetrizing the training objective does not prevent the collapse (see figure 30).

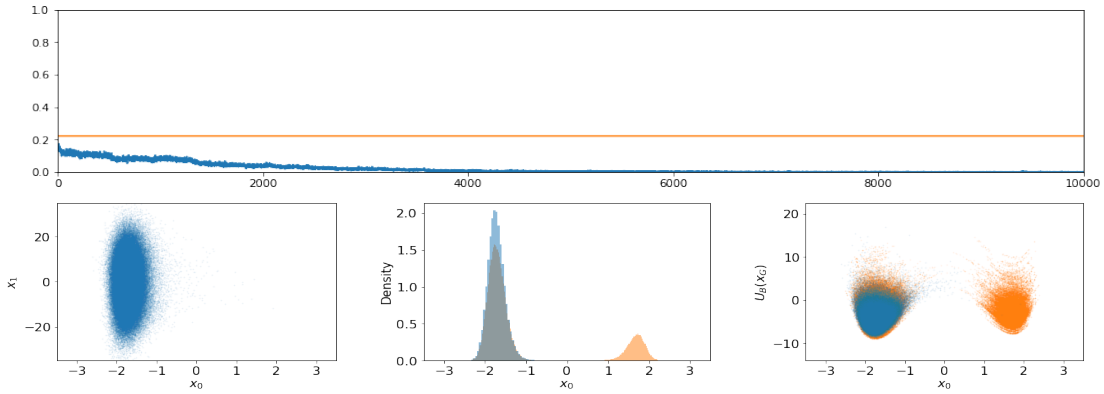


Figure 30: Fine-tuning with $\mathcal{L}_{RN_{1/2}}(\mathbf{z}_{\mathcal{N}})$ on `Double Well Wide 12D` (see caption of figure 7 for more details).

6.3 Using $PL^2(p_B, p_G)$, an L^2 loss on pairs of points

6.3.1 Definition of the $PL^2(p_B, p_G)$ objective

The simplest L^2 criterion that could be applied to log-probabilities is:

$$L^2(p_G, p_B) = \int \left(\log \frac{p_G(x)}{p_B(x)} \right)^2 dx \tag{39}$$

But this criterions cannot be used since it relies on knowing $\mathcal{Z}_B$. As reminded in section 6.1, one major requirement when designing a good training objective is not to rely on the unknown $\mathcal{Z}_B$ when transforming the objective into a usable loss function. Since $\mathcal{Z}_B$ *divides* $\tilde{p}_B$, the simplest way to remove it from an equation is by using ratios of the form $\frac{p_B(x_\gamma)}{p_B(x_\beta)} = \frac{\tilde{p}_B(x_\gamma)}{\tilde{p}_B(x_\beta)}$, where x_γ and x_β are pairs of points in x-space. Following this reasoning, an L^2 loss using pairs of points (denoted PL^2)

[59] In practice, the exponential terms are computed with softmaxes with detached denominators.

can be defined as follows:

$$PL^2(p_G, p_B) = \iint \left(\log \frac{p_B(x_\gamma)}{p_G(x_\gamma)} - \log \frac{p_B(x_\beta)}{p_G(x_\beta)} \right)^2 dx_\gamma dx_\beta \tag{40a}$$

$$= \iint \left(\log \frac{\tilde{p}_B(x_\gamma)}{p_G(x_\gamma)} - \log \frac{\tilde{p}_B(x_\beta)}{p_G(x_\beta)} \right)^2 dx_\gamma dx_\beta \tag{40b}$$

$$= \iint \Big(\big(U_G(x_\gamma) - \beta U_B(x_\gamma)\big) - \big(U_G(x_\beta) - \beta U_B(x_\beta)\big) \Big)^2 dx_\gamma dx_\beta \tag{40c}$$

$$= \mathop{\mathbb{E}}_{(x_{\mathcal{U}\gamma}, x_{\mathcal{U}\beta}) \sim p_{\mathcal{U}}^2} \left[\Big(\big(U_G(x_{\mathcal{U}\gamma}) - \beta U_B(x_{\mathcal{U}\gamma})\big) - \big(U_G(x_{\mathcal{U}\beta}) - \beta U_B(x_{\mathcal{U}\beta})\big) \Big)^2 \right] \tag{40d}$$

with:

- (40a) by defining x_γ and x_β as two different points in x-space such that $p_G(x_\gamma) \neq 0$ and $p_G(x_\beta) \neq 0$.[60]
- (40b) by noticing that the $\log \mathcal{Z}_B$ terms (from $p_B(x_\gamma)$ and $p_B(x_\beta)$) cancel one another
- (40c) by using $\log \tilde{p}_B = -\beta U_B$ and $\log p_G = -U_G$ (see section 0) with:

$$U_G(x) = -\log \left(q_{\mathcal{N}}(F(x)) \left| \det \left(\frac{\partial F(x)}{\partial x} \right) \right| \right) \tag{41}$$

- (40d) by definition of expectations and with $\mathcal{U}$ referring to the uniform distribution

6.3.2 Proof that the optimum of $PL^2(p_B, p_G)$ is correct

Since the double integral of equation 40a is defined over something that is always positive, it means that if $PL^2(p_G, p_B)$ reaches 0 then it is also at its minimum. One can also note that:

$$\forall (x_\gamma, x_\beta) \in \text{supp}(p_G)^2, \quad \left(\log \frac{p_B(x_\gamma)}{p_G(x_\gamma)} - \log \frac{p_B(x_\beta)}{p_G(x_\beta)} \right)^2 = 0 \quad \Longrightarrow \quad \frac{p_B(x_\gamma)}{p_G(x_\gamma)} = \frac{p_B(x_\beta)}{p_G(x_\beta)} = K \tag{42}$$

with K some constant.

This means that, *at 0*, p_G and p_B must be proportional. Moreover, since p_G and p_B are both probability distributions, they must sum up to 1, which implies that $K = 1$ when the support of p_G covers the support of p_B. In that case, this proves that the $PL^2(p_G, p_B)$ objective has only one minimum, which is the correct one. However, if there exists some region of the space with $p_G = 0$ and $p_B \neq 0$, $PL^2(p_G, p_B)$ can still be equal to 0, with $K \neq 1$ (see section 6.4.1).[61]

6.3.3 Proof that $PL^2(p_B, p_G)$ is convex in $U_G = -\log p_G$

PL^2 can also be proven to be convex in $U_G = -\log p_G$. Equation 40d can be generalized since the sampling procedure can be done according to any fixed

[60] Note that the β in x_β has nothing to do with the thermodynamic β defined in section 1.1

[61] This is what happens when a mode is completely ignored by the generator.

distribution p (not just $p_{\mathcal{U}}$) under the condition that p is not 0 anywhere. This leads to:

$$PL^2(p_G, p_B) = \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\Big(U_G(x_\gamma) - U_G(x_\beta) - \beta\big(U_B(x_\gamma) - U_B(x_\beta)\big)\Big)^2\right] \tag{43}$$

x_γ and x_β can even be sampled according to different distributions, but this use-case will not be introduced until section 6.4.3.

The first derivative of equation 43 is:

$$\begin{aligned}
\frac{dPL^2}{dU_G}(\delta U_G) &= 2 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\Big(U_G(x_\gamma) - U_G(x_\beta) - \beta\big(U_B(x_\gamma) - U_B(x_\beta)\big)\Big)\delta U_G(x_\gamma)\right] \\
&\quad - 2 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\Big(U_G(x_\gamma) - U_G(x_\beta) - \beta\big(U_B(x_\gamma) - U_B(x_\beta)\big)\Big)\delta U_G(x_\beta))\right] && \text{(44a)} \\
&= 2 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\Big(U_G(x_\gamma) - U_G(x_\beta) - \beta\big(U_B(x_\gamma) - U_B(x_\beta)\big)\Big)\delta U_G(x_\gamma)\right] \\
&\quad + 2 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\Big(U_G(x_\beta) - U_G(x_\gamma) - \beta\big(U_B(x_\beta) - U_B(x_\gamma)\big)\Big)\delta U_G(x_\beta))\right] && \text{(44b)} \\
&= 4 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\Big(U_G(x_\gamma) - U_G(x_\beta) - \beta\big(U_B(x_\gamma) - U_B(x_\beta)\big)\Big)\delta U_G(x_\gamma)\right] && \text{(44c)}
\end{aligned}$$

with:

- (44a) by differentiating equation 43
- (44b) by putting the negative sign within the second expectation
- (44c) by noticing that x_γ and x_β can be swapped

The second derivative, in turn, is:

$$\frac{d^2PL^2}{df^2}(\delta U_G)(\delta U'_G) = 4 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\big(\delta U'_G(x_\gamma) - \delta U'_G(x_\beta)\big)\delta U_G(x_\gamma)\right] \tag{45}$$

This quantity is a matrix which is positive if and only if when multiplied from the left and from the right by any vector the result is always positive. To pick the same vector, $\delta U'_G$ is replaced with δU_G in the following equation:

$$\begin{aligned}
\frac{d^2PL^2}{dU_G^2}(\delta U_G)(\delta U_G) &= 4 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\big(\delta U_G(x_\gamma) - \delta U_G(x_\beta)\big)\delta U_G(x_\gamma)\right] && \text{(46a)} \\
&= 4 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\delta U_G(x_\gamma)^2\right] - 4 \cdot \mathop{\mathbb{E}}_{(x_\gamma, x_\beta)\sim p^2}\left[\delta U_G(x_\beta)\delta U_G(x_\gamma)\right] && \text{(46b)} \\
&= 4 \cdot \mathop{\mathbb{E}}_{x_\gamma\sim p}\left[\delta U_G(x_\gamma)^2\right] - 4 \cdot \mathop{\mathbb{E}}_{x_\gamma\sim p}\left[\delta U_G(x_\gamma)\right] \mathop{\mathbb{E}}_{x_\beta\sim p}\left[\delta U_G(x_\beta)\right] && \text{(46c)} \\
&= 4 \cdot \mathop{\mathbb{E}}_{x_\gamma\sim p}\left[\delta U_G(x_\gamma)^2\right] - 4 \cdot \left(\mathop{\mathbb{E}}_{x_\gamma\sim p}\left[\delta U_G(x_\gamma)\right]\right)^2 && \text{(46d)} \\
&= 4 \cdot \left(\mathop{\mathbb{E}}_{x_\gamma\sim p}\left[\delta U_G(x_\gamma)^2\right] - \left(\mathop{\mathbb{E}}_{x_\gamma\sim p}\left[\delta U_G(x_\gamma)\right]\right)^2\right) && \text{(46e)} \\
&\geqslant 0 && \text{(46f)}
\end{aligned}$$

with:

- (46c) since $x_{\gamma} \sim p$ and $x_{\beta} \sim p$ are independent.
- (46f) by noticing that equation 46e is a formula for the variance, which is known to always be positive.

This proves the convexity of the $PL^2(p_G, p_B)$ objective in $U_G = -\log p_G$.

6.4 Choosing the distribution p that weights $PL^2(p_B, p_G)$

6.4.1 Picking $p = p_G$ leads to collapse

The expectation from equation 43 can be weighted according to any distribution p, as long as p is not zero anywhere and does not depend on p_G, otherwise the optimality proof from section 6.3.2 does not hold (because one would have to differentiate wrt p also). But what happens if p is chosen to be p_G *anyway* in equation 47? This quantity is always positive (or 0), and $p_G = p_B$ is indeed a global minimum (i.e. $PL^2(p_G = p_B, p_B) = 0$):

$$PL^2_G(p_G, p_B) = \mathop{\mathbb{E}}_{(x_{\gamma}, x_{\beta}) \sim p_G^2} \left[\Big(U_G(x_{\gamma}) - U_G(x_{\beta}) - \beta \big(U_B(x_{\gamma}) - U_B(x_{\beta}) \big) \Big)^2 \right] \tag{47}$$

However, there are plenty of other global minima. This is because:

$$PL^2(p_G, p_B) = 0$$

$$\iff$$

$$\forall (x_{\gamma}, x_{\beta}) \left[p_G(x_{\gamma}) p_G(x_{\beta}) = 0 \quad \text{or} \quad \Big(U_G(x_{\gamma}) - U_G(x_{\beta}) - \beta \big(U_B(x_{\gamma}) - U_B(x_{\beta}) \big) \Big)^2 = 0 \right]$$

Said differently, it means that it is possible for p_G to be equal to 0 in some regions of the space (for example in one of many modes) and to be equal to $K \cdot p_B$ in other regions with $K \geq 1$.[62]

When choosing $p = p_G$, the resulting loss from equation 48 leads to a quick collapse (as shown in figure 31).[63]. This result is interesting, in the sense that the only term that is differentiated w.r.t. θ here is U_G, which means that the collapse behavior is *not* induced by the differentiation of the potential energy term U_B.

$$\mathcal{L}_{PL^2_G}(\mathbf{x}^{\ddagger}_G) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{n^2} \left[\Big(U_G(x^{\ddagger}_{G,i}) - U_G(x^{\ddagger}_{G,j}) - \beta \big(U_B(x^{\ddagger}_{G,i}) - U_B(x^{\ddagger}_{G,j}) \big) \Big)^2 \right] \tag{48}$$

[62] Other training objectives like $KL(p_G || p_B)$ and $RN_{1/2}(p_G, p_B)$ do not have this problem in the limit of a dense sampling. However, once transformed into loss functions, when sampling mini-batches, regions of the space that are not sampled by p_G will not be penalized and therefore never recovered. But, as said before, this observation does not explain why modes are *lost* if p_G already samples them.

[63] The double sum of equation 48 corresponds to the double integral of equation 40c. The first index i refers to x_{γ} points and the second index j refers to x_{β} points.

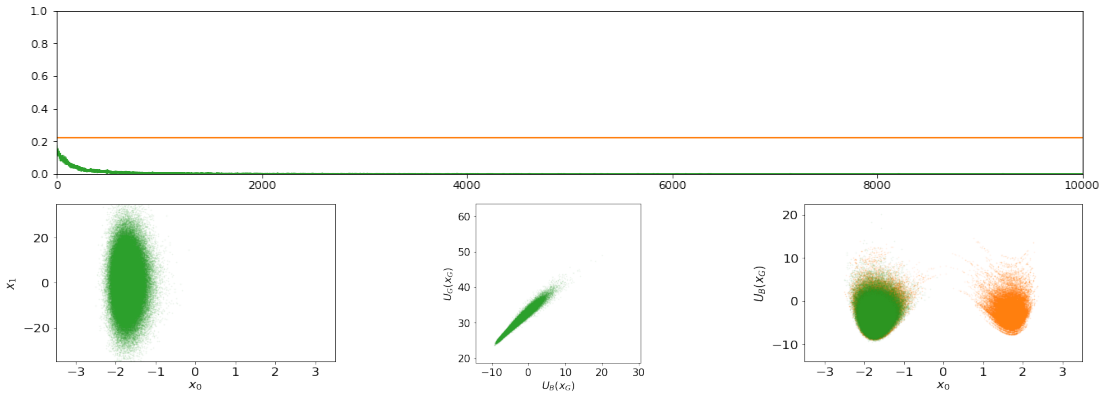


Figure 31: Fine-tuning with $\mathcal{L}_{PL^2_G}$ on `Double Well Wide 12D` (see caption of figure 7 for more details).

6.4.2 Picking $p = p_B$ leads to instabilities

When choosing $p = p_B$, importance sampling weights appear (similar to those from equation 19) and result in the following loss function:

$$\mathcal{L}_{PL^2_B}(\mathbf{x}^{\ddagger}_G) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{n^2}\left[w_i w_j \Big(U_G(x^{\ddagger}_{G,i}) - U_G(x^{\ddagger}_{G,j}) - \beta\big(U_B(x^{\ddagger}_{G,i}) - U_B(x^{\ddagger}_{G,j})\big)\Big)^2\right] \tag{49}$$

with w_i a quantity proportional to $(\tilde{p}_B(x_{G,i})/p_G(x_{G,i}))^{\ddagger}$.

This loss leads to a progressive increase in the potential energy of generated points (i.e. $U_B(x_G)$), until the generated distribution has almost no point in common with p_B (data not shown). Since this objective only works *within a mini-batch*, some points are always considered over-sampled *relative to others* (even when the generated distribution is already excellent). The energy of these points is then maximized, even if it doesn't need to be, and the model ends up generating many high-energy points.

To summarize, this loss only corrects the relative probabilities of p_B points, even if they have a very low probablity of being generated by the model and without making them any more probable as a whole.

Side-note: Having well-defined training objectives leading to losses that behave poorly experimentally is a recurring theme. Although it is only an hypothesis, the main cause of the problem probably comes from the fact that training objectives use integrals whereas empirical losses are estimated over finite sets of points, which *considerably* changes the expected behavior of the training procedure. This point is discussed further in section 9.

6.4.3 Picking $p(x_\gamma) = p_G(x_\gamma)$ and $p(x_\beta) = p_B(x_\beta)$ leads to stable trainings

A promising idea to overcome the problems of section 6.4.2 is to only optimize points that are deemed *under*-sampled within each mini-batch. This can be accom-

plished quite simply by using a mask as follows:

$$\tilde{U}_{\text{diff}}(x^{\ddagger}_{G,i}, x^{\ddagger}_{G,j}) = U_G(x^{\ddagger}_{G,i}) - U^{\ddagger}_G(x^{\ddagger}_{G,j}) - \beta\big(U_B(x^{\ddagger}_{G,i}) - U_B(x^{\ddagger}_{G,j})\big) \tag{50a}$$

$$U_{\text{diff}}(x^{\ddagger}_{G,i}, x^{\ddagger}_{G,j}) = \begin{cases} \tilde{U}_{\text{diff}}(x^{\ddagger}_{G,i}, x^{\ddagger}_{G,j}) & \text{if } \tilde{U}_{\text{diff}}(x^{\ddagger}_{G,i}, x^{\ddagger}_{G,j}) \geq 0 \\ 0 & \text{if } \tilde{U}_{\text{diff}}(x^{\ddagger}_{G,i}, x^{\ddagger}_{G,j}) < 0 \end{cases} \tag{50b}$$

$$\mathcal{L}_{PL^2_{GB}}(\mathbf{x}^{\ddagger}_G) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{n^2}\left[w_j\Big(U_{\text{diff}}(x^{\ddagger}_{G,i}, x^{\ddagger}_{G,j})\Big)^2\right] \tag{50c}$$

with:

- w_j a quantity proportional to $(\tilde{p}_B(x_{G,j})/p_G(x_{G,j}))^{\ddagger}$

Importantly, the variation of PL^2 used here is denoted $PL^2_{GB}(p_G, p_B)$. The subscript GB refers to the fact that the points in each pair of the loss are associated with different probabilities.

- The first points of each pair, with index i, are sampled according to p_G (notice the missing w_i in equation 50c). They produce $U_G(x^{\ddagger}_{G,i})$ which is not detached. It is through this term alone that θ is optimized.
- The second points of each pair, with index j, are sampled according to p_G but have p_B importance sampling weights (notice the presence of w_j in equation 50c). They produce $U^{\ddagger}_G(x^{\ddagger}_{G,j})$ which is detached. Therefore those points are only used as reference points.

In essence, this loss is similar to comparing the probability ratios of points sampled according to p_G with points sampled according to p_B, and leads to excellent results as shown in figure 32, albeit some slightly over-sampled points in the inter-mode.[64]

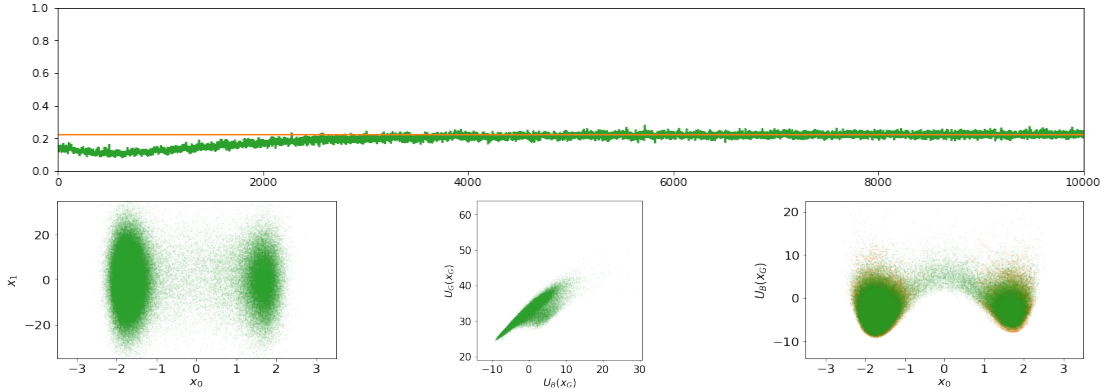


Figure 32: Fine-tuning with $\mathcal{L}_{PL^2_{GB}}$ on `Double Well Wide 12D` (see caption of figure 7 for more details).

Compared to the results of $\mathcal{L}_{PL^2_G}$ in figure 31 there are two changes that may justify the better performance of $\mathcal{L}_{PL^2_{GB}}$ in figure 32. First, a mask is added is equation 50b, and second, batch weights w_j are added to the points used as "reference"

[64]Having slightly over-sampled points in the inter-mode might actually provide more stability during training, since the model ends up tearing the space less in these regions (see figure 44a appendix A.4 for a visualization).

in equation 50c. Among those two changes, it is the mask that brings the most improvement. Simply using the batch weights w_j without the mask leads to collapse (data not shown), whereas using the mask without the batch weights hinders the performance but does not lead to collapse as shown in figure 33. This is a crucial result that is explored further in section 6.5.3.

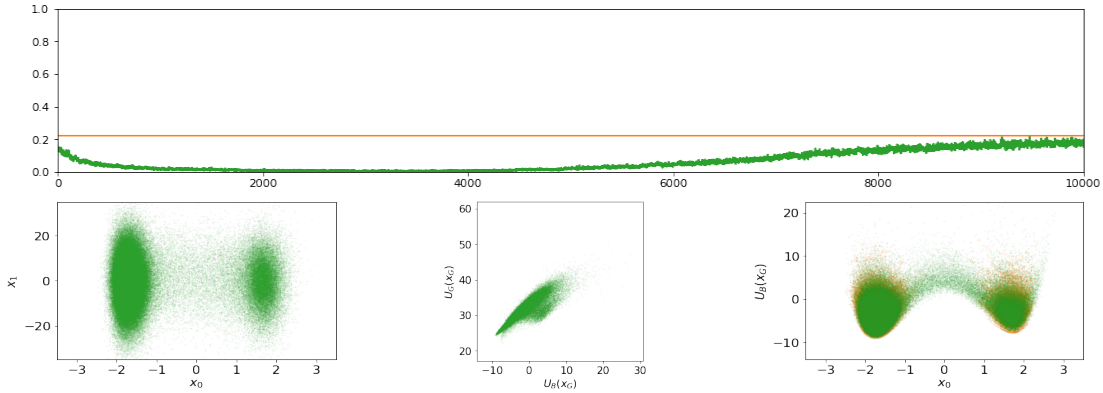


Figure 33: Fine-tuning with $\mathcal{L}_{PL^2_G}$ on `Double Well Wide 12D` but with the mask from equation 50b ensuring that only under-sampled points are optimized (see caption of figure 7 for more details).

6.4.4 Using a cache of the most under-sampled points

This idea of only optimizing points that are deemed under-sampled can be pushed one step further by using a cache. At each timestep during training, half of the points of the mini-batch, those that are the most under-sampled, are kept in memory within the cache and passed on to the next timestep. At the next timestep, this "half-mini-batch" is completed with new random points sampled according to p_G.

In this setup, the first point of each pair (which is differentiated) comes from this sampling method, whereas the second point of each pair (which is not differentiated) is still sampled from p_G with importance sampling weights from p_B. This loss is denoted $\mathcal{L}_{PL^2_{CB}}$ (with "C" refering to the sampling method using the cache).[65]

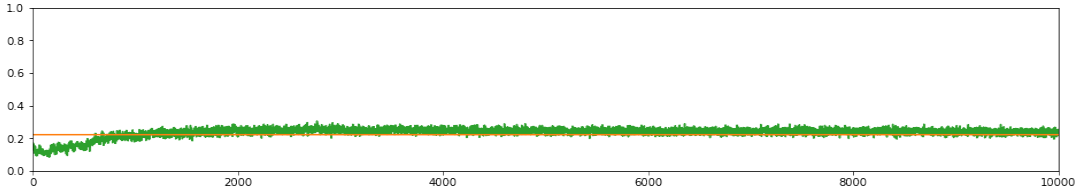


Figure 34: Percentage of points generated by p_G in the minor mode during fine-tuning with $\mathcal{L}_{PL^2_{CB}}$ (in green) compared with the real ratio of ≈ 0.22 from the `Double Well Wide 12D` dataset (in orange).

This simple trick ensures that the model has a chance to recover an entire missing mode from only a single point sampled in that mode at one timestep. Overall, the result shows a slightly improved convergence on `Double Well Wide 12D` (as

[65] To be clear, the mask introduced in section 6.4.3 is still in use here.

shown in figure 34) and a higher quality of generation on `Dialanine` (as shown in section 6.5).[66]

6.5 $\mathcal{L}_{PL^2_{CB}}$ compares favorably with any other loss so far

6.5.1 Data-free fine-tuning with $\mathcal{L}_{PL^2_{CB}}$ on `Dialanine`

$\mathcal{L}_{PL^2_{CB}}$ performs identically to $\mathcal{L}^{\text{df}}_{KLz}$ on `Butane` (see appendix A.7) but outperforms it on `Dialanine`. When comparing figure 36c with figure 24c, one can immediately notice that some points remain in the minor mode when using $\mathcal{L}_{PL^2_{CB}}$ whereas they do not when using $\mathcal{L}^{\text{df}}_{KLz}$. This is also confirmed in figure 37c where points in the minor mode have a low energy of generation, contrary to figure 25c where those points are close to impossible to generate.[67]

6.5.2 Impact of having hydrogen atoms that are deterministically fixed

But although the minor mode of `Dialanine` has not been lost, points in that mode are generated only about 1% of the time at the end of the fine-tuning, when in fact they represent about 6% of the dataset (as illustrated in figure 35). Importantly, this problem is *not* related to the loss, but is a consequence of using a model in two stages, with the second stage placing the hydrogen atoms deterministically (as described in section 3.3). This method drastically simplifies the task that the model has to solve (since it lowers the dimensionality and makes hydrogen permutations irrelevant) but also changes the target distribution. Indeed, a simulation produces different mode ratios depending on whether or not the energy of the hydrogen atoms is continuously minimized (see appendix A.2). The expected percentage of points that would be expected under such a constraint can be estimated to be close to 1.2% by using the estimator from equation 87. It is remarkable to notice that the model did not lose the C7ax mode even though it is represented by so few points.

$\mathcal{L}_{PL^2_{CB}}$ is therefore the first loss that demonstrates perfectly stable fine-tunings across all datasets (`Double Well Wide 12D`, `Butane` and `Dialanine`), and produces correct ratios between modes in a *data-free* regime, with excellent energy correlations between U_G and U_B.

Note that when transforming the training objective into a loss in equation 40, no change of variable was used (contrary to equations 14d and 37d for instance). This is a deliberate choice since using the change of variable produces a loss that leads to collapse, in a similar fashion to $\mathcal{L}_{KLx}$.

[66] Note that using a cache with the KLz slows down the loss of the minor mode on `Dialanine`, but does not prevent it (data no shown).

[67] In fact, those results can also be achieved *without* using the concept of a cache introduced in section 6.4.4 (i.e. by using $\mathcal{L}_{PL^2_{GB}}$ instead of $\mathcal{L}_{PL^2_{CB}}$) as shown in appendix A.8.

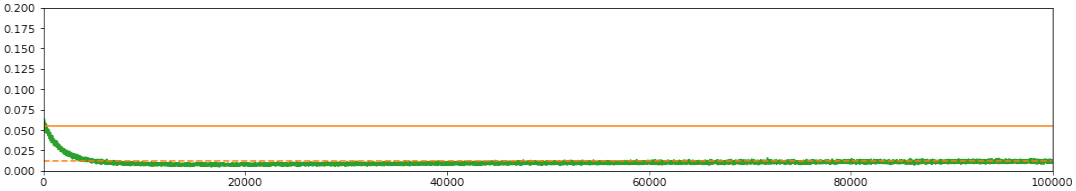
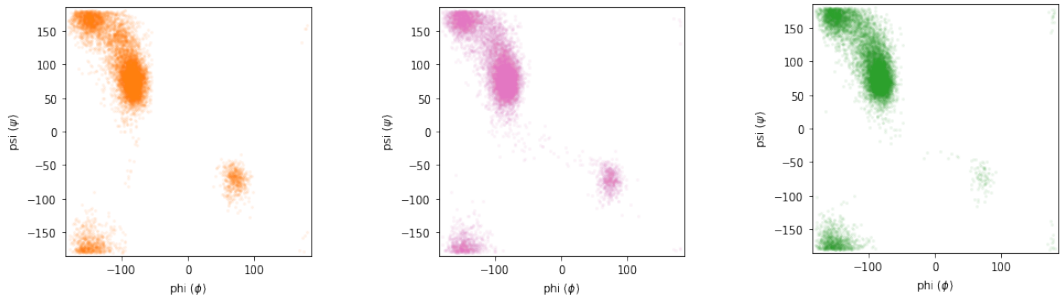
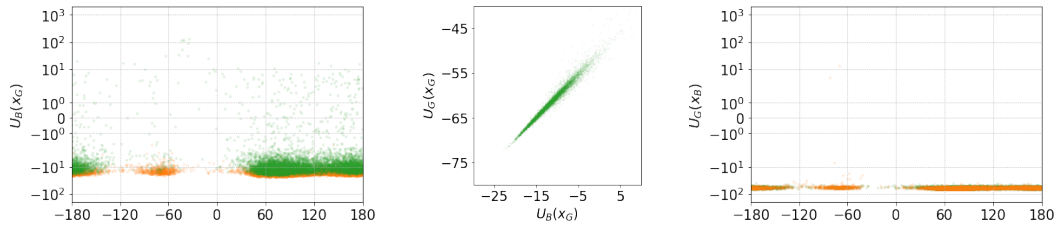


Figure 35: Ratio of generated x_G samples in the minor modes during fine-tuning with $\mathcal{L}_{PL^2_{CB}}$ on `Dialanine`. The ordinates go from 0% to 20%.



(a) ϕ and ψ from the `Dialanine` dataset. (b) ϕ and ψ after pre-training with $\mathcal{L}^{\text{dd}}_{KLz}$. (c) ϕ and ψ after fine-tuning with $\mathcal{L}_{PL^2_{CB}}$

Figure 36: Dihedral angles ϕ and ψ of the dialanine molecule. Figures 36a and 36b are duplicates of figures24a and 24b since the same data and pre-training are used.



(a) U_B energies of samples $x_G \sim p_G$. (b) Correlations. (c) U_G energies of samples $x_B \sim p_B$.

Figure 37: Quality of generation after fine-tuning on `Dialanine` with $\mathcal{L}_{PL^2_{CB}}$.

6.5.3 The likely culprit behind the mode collapse

Every single data-free loss introduced so far optimizes one or several of the four following terms:

- The potential energy in x-space: $\beta U_B(G(z_\mathcal{N}))$
- The expansion factor from z to x: $\log \left|\det \left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}}\right)\right|$
- The potential energy in z-space: $\frac{1}{2\sigma^2} U_\mathcal{N}(F(x^\ddagger_G))$
- The expansion factor from x to z: $\log \left|\det \left(\frac{\partial F(x^\ddagger_G)}{\partial x^\ddagger_G}\right)\right|$

Crucially, $\log \left|\det \left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}}\right)\right|$ is by definition the opposite of $\log \left|\det \left(\frac{\partial F(x^\ddagger_G)}{\partial x^\ddagger_G}\right)\right|$. Therefore, minimizing $\log \left|\det \left(\frac{\partial G(z_\mathcal{N})}{\partial z_\mathcal{N}}\right)\right|$ amounts to maximizing $\log \left|\det \left(\frac{\partial F(x^\ddagger_G)}{\partial x^\ddagger_G}\right)\right|$ and vice versa.

Here is a summary of the losses studied so far:

- $\mathcal{L}_{KLx}(\mathbf{z}_{\mathcal{N}})$ [68] and $\mathcal{L}_{RN_{1/2}}(\mathbf{z}_{\mathcal{N}})$ [69] both minimize $\beta U_B(G(z_{\mathcal{N}}))$ and maximize $\log\left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|$. These losses easily lead to collapse if the pre-training quality is not sufficient.
- $\mathcal{L}^{\text{df}}_{KLz}(\mathbf{x}^{\ddagger}_G)$ [70] minimizes $\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x^{\ddagger}_G))$ and maximizes $\log\left|\det\left(\frac{\partial F(x^{\ddagger}_G)}{\partial x^{\ddagger}_G}\right)\right|$. This loss is stable.
- Unmasked versions of $\mathcal{L}_{PL^2_G}(\mathbf{x}^{\ddagger}_G)$ [71] and $\mathcal{L}_{PL^2_{GB}}(\mathbf{x}^{\ddagger}_G)$ sometimes minimize and sometimes maximize $\log\left|\det\left(\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right)\right|$. These losses easily leads to collapse if the pre-training quality is not sufficient.
- Masked versions of $\mathcal{L}_{PL^2_G}(\mathbf{x}^{\ddagger}_G)$ and $\mathcal{L}_{PL^2_{GB}}(\mathbf{x}^{\ddagger}_G)$ [72] are both stable and only maximize $\log\left|\det\left(\frac{\partial F(x^{\ddagger}_G)}{\partial x^{\ddagger}_G}\right)\right|$.

	$\beta U_B(G(z_{\mathcal{N}}))$	$\log\left\|\det\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right\|$	$\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x^{\ddagger}_G))$	$\log\left\|\det\frac{\partial F(x^{\ddagger}_G)}{\partial x^{\ddagger}_G}\right\|$	
$\mathcal{L}_{KLx}$	$\downarrow$	$\uparrow$			✗
$\mathcal{L}_{RN_{1/2}}$	$\downarrow$	$\uparrow$			✗
$\mathcal{L}^{\text{df}}_{KLz}$			$\downarrow$	$\uparrow$	✓
$\mathcal{L}_{PL^2_{GB}}$			$\downarrow\uparrow$	$\downarrow\uparrow$	✗
$\mathcal{L}_{PL^2_{GB}}$ +mask			$\downarrow$	$\uparrow$	✓

Table 1: Summary of the main losses studied in sections 4, 5 and section 6. The arrows $\uparrow$ and $\downarrow$ indicate whether each loss increases or decreases the column terms respectively. The symbols of the last column indicate whether the loss leads to collapse (✗) or not (✓).

At this stage, a clear pattern emerges. Losses that rely on a maximization of $\log\left|\det\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right|$ (or an equivalent minimization of $\log\left|\det\frac{\partial F(x^{\ddagger}_G)}{\partial x^{\ddagger}_G}\right|$) lead to collapse, whereas losses that only minimize it are stable. Although this is true for every variation of every loss investigated, there is no guarantee that this observation always holds as it is unclear why maximizing $\log\left|\det\frac{\partial G(z_{\mathcal{N}})}{\partial z_{\mathcal{N}}}\right|$ (which is equivalent to minimizing $p_G(x^{\ddagger}_G)$) would lead to collapse when sampling mini-batches, nor why the problem is exacerbated under the conditions explored in this work (i.e. when the pre-training is poor, when sampling mini-batches on a high dimensional problem and when the target distribution has multiple modes).

[68] See equation 16 in section 4.1 and figures 10, 18, 19 and 21 in their respective sections.
[69] See equation 38 in section 6.2 and figure 30.
[70] See equation 20 in section 5.1 and figures 18, 19, 21 and 24/25/26 in their respective sections.
[71] See equation 48 in section 6.4.1 and figure 31.
[72] See equation 50 in section 6.4.3 and figures 32, 33 and 35/36/37 in their respective sections.

7 Using the variance of probability ratios as a loss

7.1 Using the variance of estimators as a loss

With Normalizing Flows, one can compute exactly the probability p_G 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 $\frac{p_B(x)}{p_G(x)}$ (see equation 9). 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 output distribution is always perfect.

An important application of the 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:

$$e^{-\beta \Delta F_{BC}} \coloneqq \mathop{\mathbb{E}}_{x \sim p_B}[f] \qquad \text{with} \qquad f(x) = e^{-\beta(U_C(x) - U_B(x))} \tag{51}$$

The true value of the quantity to estimate is denoted Q:

$$Q \coloneqq \mathop{\mathbb{E}}_{p_B}[f] = \mathop{\mathbb{E}}_{p_G}\left[\frac{p_B}{p_G} f\right] \tag{52}$$

This equality is always exactly true provided that p_G is never 0 where p_B is not. For any p_G, the following quantity $\hat{Q}$ is an unbiased estimator of Q (see appendix A.10):

$$\hat{Q} \coloneqq \frac{1}{n} \sum_{x_{G,i} \in \mathbf{x}_G} \frac{p_B(x_{G,i})}{p_G(x_{G,i})} f(x_{G,i}) \tag{53}$$

where $\mathbf{x}_G = (x_{G,1}, \dots, x_{G,n})$ is a mini-batch of points sampled according to p_G. That is, when averaging over all possible mini-batches, $\hat{Q}$ becomes Q (i.e. $\mathbb{E}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}] = Q$). Yet, for some distributions p_G, the estimate $\hat{Q}$ may converge faster than for others, in terms of number of samples required to reach a given target accuracy. The quality of a generator can thus be quantified through the expected error when estimating Q with n points. This error is proportional to the variance of $\hat{Q}$, and can then be turned into a training loss (see appendix A.11):

$$\mathop{\mathbb{E}}_{x_G \sim p_G}\left[\frac{p_B^2(x_G)}{p_G^2(x_G)} f^2(x_G)\right] \tag{54}$$

If the function f is not fixed and can be any bounded function over x-space, then

one can deduce the following loss function:

$$\mathcal{L}_{L^2}(x_G^\ddagger) = \mathop{\mathbb{E}}_{x_B\sim p_B}\left[\frac{p_B(x_B)}{p_G(x_B)}\right] = \mathop{\mathbb{E}}_{x_G^\ddagger\sim p_G}\left[\frac{p_B^2(x_G^\ddagger)}{p_G^2(x_G^\ddagger)}\right] = e^{RN_2(p_B||p_G)} = \mathop{\text{var}}_{x_G^\ddagger\sim p_G}\left[\frac{p_B(x_G^\ddagger)}{p_G(x_G^\ddagger)}\right] + 1 \tag{55}$$

where $RN_2(p_B||p_G)$ is the Rényi divergence of order 2, thus penalizing high ratios $\frac{p_B}{p_G}$ more strongly. The last equality derives from equation 102 in appendix A.11.[73]

Thus we arrive at $\mathcal{L}_{L^2}$ as a principled loss to minimize the variance of estimators of expectations over the Boltzmann distribution. 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 x-space. Without knowing the value of the target constant, this can still be achieved by minimizing the variance of the ratio, which is precisely the $\mathcal{L}_{L^2}$ loss.

Building on equation 55, we replace ratios $\frac{p_B(x)}{p_G(x)}$ with log-ratios for numerical reasons, since Normalizing Flows actually compute log-probabilities and the exponentiation leads to instability:

$$r(x_G) = \log\frac{p_B(x_G)}{p_G(x_G)} \tag{56}$$

We also note that:

$$\mathop{\text{var}}_{p_G}[r] = \mathop{\mathbb{E}}_{p_G}\left[\left(r - \mathop{\mathbb{E}}_{p_G}[r]\right)^2\right] \tag{57}$$

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 $p_G(x_G^\ddagger)$ explicitly at any point $x_G^\ddagger$ (see section 6.5.3), the loss is modified by masking $(r - \mathbb{E}_{p_G}[r])$ (similarily to the strategy followed in section 6.4). As a consequence, $r(x_G^\ddagger)$ can only be minimized (i.e. $U_G(x_G^\ddagger)$ can only be minimized and $p_G(x_G^\ddagger)$ can only be maximized) whereas $U_B(x_G^\ddagger)$ is not differentiated with respect to θ. The masked loss with detached means is therefore defined as:

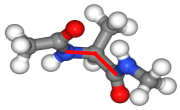
$$\mathcal{L}_{L^2_+}(\mathbf{x}_G^\ddagger) = \sum_{i=1}^{n}\left[\frac{1}{n}\cdot\left[\left(r(x_{G,i}^\ddagger) - K^\ddagger\right)_+^2\right]\right] \tag{58}$$

where $a_+^2 = a^2$ if $a > 0$ and 0 otherwise, and where $K^\ddagger = \left[\sum_{j=1}^n\left[\frac{1}{n}\cdot r(x_{G,j}^\ddagger)\right]\right]^\ddagger$ is not differentiated (so as to ensure that it is never increased).[74] See section 7.2 a proof of convergence.

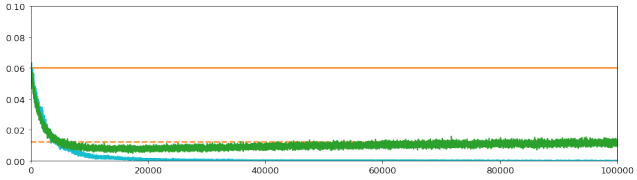
One can prove that, in spite of the non-differentiation of $K^\ddagger$, a pseudo-gradient descent on this loss *will* converge towards p_B, provided the model is expressive

[73] In practice, one knows how to compute $\tilde{p}_B(x) \coloneqq \mathcal{Z}_B p_B(x)$ but not $p_B(x)$ directly. This is not an issue since $\mathcal{Z}_B$ cancels out in equations 57 and 58. Moreover, a model p_G trained with $\mathcal{L}$ will yield by definition a good estimator of $\mathcal{Z}_B = \mathbb{E}_{p_B}[\mathcal{Z}_B] = \mathbb{E}_{p_G}\left[\frac{\tilde{p}_B}{p_G}\right]$

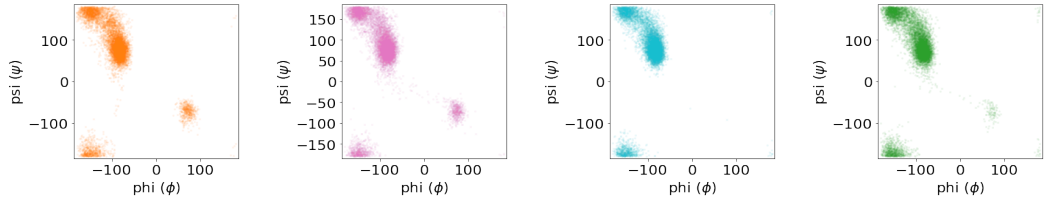
[74] Note that in the continuous limit: $\mathbb{E}_{p_G}[r] = -KL(p_G||p_B) \leqslant 0$.



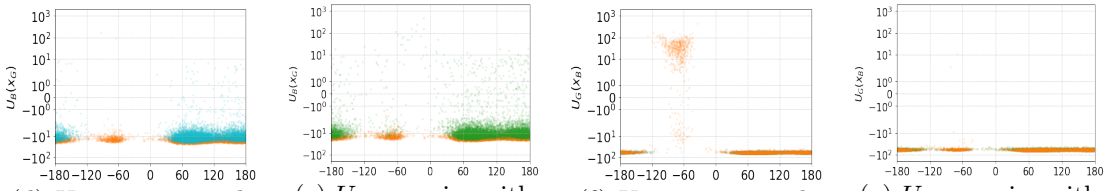
(a) A configuration of dialanine



(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)

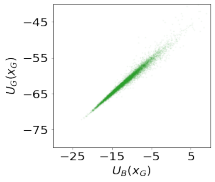


(d) U_B energies with $\mathcal{L}^{\text{df}}_{KLz}$

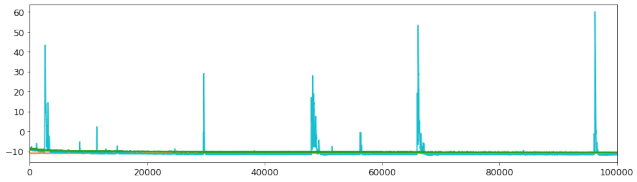
(e) U_B energies with $\mathcal{L}_{L^2_+}$

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

- Figure 38b 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, whereas the dashed orange line corresponds to the same ratio from p_B but with the energy minimized with respect to hydrogen atom coordinates (see section 3.3 and appendix A.2).
- Figure 38c contains 2D projections of the ground truth dataset and generated data.
- Figures 38d and 38e 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 hydrogen atoms is minimized (either by the model or manually). These figures visualize whether or not $p_G \subset p_B$.
- Figures 38f and 38g 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 38h 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 38i 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 $\mathcal{L}_{L^2_+}$.

Note that similar excellent results are achieved with $\mathcal{L}_{L^2_+}$ when changing the targed distribution with $\lambda_{\text{align}} = 1$ or $\lambda_{\text{align}} = 100$ (data not shown).

enough and that the initial p_G is non-zero on the support of p_B (see section 7.2).

Experimentally, $\mathcal{L}_{L^2_+}$ performs extremely well when compared to $\mathcal{L}^{\text{df}}_{KLz}$. It is clear that $\mathcal{L}^{\text{df}}_{KLz}$ completely loses the minor mode (figures 38c and 38f) whereas $\mathcal{L}_{L^2_+}$ does not (figures 38c and 38g) and converges to the expected ratio of $\approx 1.2\%$.[75] 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 $\mathcal{L}^{\text{df}}_{KLz}$ has unstable U_B energies during fine-tuning whereas the $\mathcal{L}_{L^2_+}$ does not (figure 38i).

In addition to those clear qualitative improvements, and unlike $\mathcal{L}^{\text{df}}_{KLz}$, $\mathcal{L}_{L^2_+}$ 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 could be lifted by using a conditional normal flow to generate a Boltzmann distribution of hydrogen atom positions given a set of heavy atom positions generated by the main network.

7.2 Proof of convergence

Introductory remarks. Given a mini-batch of x_i points, $r_i = \log \frac{p_B(x_i)}{p_G(x_i)}$ denotes the log-ratio of the target and generated densities at point x_i. As mentioned in the section 7.1, differences of log-ratios satisfy:

$$r_i - r_j = \log \frac{p_B(x_i)}{p_G(x_i)} - \log \frac{p_B(x_i)}{p_G(x_i)} = \log \frac{\tilde{p}_B(x_i)}{p_G(x_i)} - \log \frac{\tilde{p}_B(x_i)}{p_G(x_i)} \tag{59}$$

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

Interestingly:

$$\mathop{\mathbb{E}}_{x_j \sim p_B}[r_j] = KL(p_B||p_G) \qquad \text{and} \qquad \mathop{\mathbb{E}}_{x_i \sim p_G}[r_i] = -KL(p_G||p_B) \tag{60}$$

The simplest version of the loss (see equation 55) is equal to $e^{RN_2(p_B||p_G)}$ where RN_2 is the Rényi divergence of order 2, a measure of divergence between distributions. As a consequence, the only minimum is the global one, reached at $p_G = p_B$.

The objective of this section is to study the optimization properties of $\mathcal{L}_{L^2_+}$ (defined in equation 58) which uses both a mask and detached means.

[75] Recall that 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 $\approx 1.2\%$ (dashed line in figure 38b, see section 3.3 and appendix A.2). This means that the result of the pre-training with the data-dependent $\mathcal{L}^{\text{dd}}_{KLz}$ produces a ratio ($\approx 6\%$, figure 38c) that is different from the one expected at the end of the fine-tuning ($\approx 1.2\%$).

- $(r_i - K^\ddagger)$ is masked with $()_+$ so that r_i can only be 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 importantly this will not be done by the gradient descent, hence not in the worst possible direction (by making x_j as unlikely as possible as fast as possible), but rather in the smoothest possible way (by pushing 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.

This loss is non-negative, and 0 can be reached if all r_i are equal (note that a 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 said 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 this section below for more details).

Optimization process. However, since only part of the loss is differentiated (although K changes with p_G it is treated as a constant), then strictly speaking the optimization process is not a gradient descent, as the loss function 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 over the total mass of p_G (which remains equal to 1) is handled implicitly by the Normalizing Flow, and so the precise way the gradient with respect to p_G is replaced with a variation δp_G that preserves the total mass depends on the architecture and the weights of the model.

The variation of K induced by a (partial) gradient step is:

$$\delta K = \delta \left(\mathop{\mathbb{E}}_{x \sim p_G}[r] \right) = \delta \left(\int p_G \; r \; dx \right) = \mathop{\mathbb{E}}_{x \sim p_G}[\delta r] + \int r \; \delta p_G \; dx \tag{61}$$

Note that:

$$p_G(x_i) = e^{\log p_G(x_i)} = e^{-r_i + \log p_B(x_i)} = e^{-r_i} p_B(x_i) \tag{62}$$

By taking the integral of equation 62 over x_i, if follows that $\mathbb{E}_{x_B \sim p_B}[e^{-r(x_B)}] = 1$ (since $\mathbb{E}_{p_G}[1] = 1$), and this is true for any p_G or equivalently for any associated r. As a consequence, the variation of $\mathbb{E}_{x_B \sim p_B}[e^{-r(x_B)}]$ with respect to any realizable change δr (p_B being fixed and p_G varying) is necessarily 0:

$$\delta \left(\mathop{\mathbb{E}}_{x \sim p_B}[e^{-r}] \right) = - \mathop{\mathbb{E}}_{x \sim p_B}[e^{-r} \delta r] = 0 \qquad \text{and so:} \qquad \mathop{\mathbb{E}}_{x \sim p_G}[\delta r] = 0 \tag{63}$$

Consequently δK can be simplified as:

$$\delta K = \int r \; \delta p_G \; dx \tag{64}$$

but also rewritten as:

$$\delta K = \int (r - K)\ \delta p_G\ dx \tag{65}$$

since $K \int \delta p_G = 0$ as p_G has a conserved total mass of 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:

$$\delta K \geqslant 0 \tag{66}$$

Consequences K is therefore increasing with time, and since $K = -KL(p_G||p_B)$, 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, one can conclude that the optimization process leads to the desired global optimum.

Another way to see this is that K is an increasing, upper-bounded value (bounded by 0). When K converges (possibly to a non-0 value), then the training criterion becomes stable and the optimization process becomes a real gradient descent with respect to r_i. Therefore a local minimum of this loss (for a fixed limit K) is 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 is strictly less than K, then 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.

8 Optimal Transport and the Sinkhorn algorithm

8.1 Data-dependent case

Losses based on L^2 distances and Rényi Divergences (including KL divergences) all aim to change the probability p_G of the generated points of the mini-batch. A different approach consists in considering the global movement of the mini-batch by interpreting the task as a transportation problem. The goal here is to move p_G's mass closer to p_B but at minimal cost (i.e. by also taking into account the total distance traveled to reach p_B). The training objective that follows this formulation of the problem is called the Wasserstein distance W. See Peyré and Cuturi [42] for a comprehensive review.

In the data-dependent case, two mini-batches are available at each training iteration: $\mathbf{x}_G \sim p_G^n$ and $\mathbf{x}_B \sim p_B^m$.[76] Those mini-batches correspond to discrete probability measures:

$$\gamma = \sum_{i=1}^{n} \mathbf{g}_i \delta_{x_{G,i}} \qquad \text{and} \qquad \beta = \sum_{i=1}^{m} \mathbf{b}_j \delta_{x_{B,j}} \tag{67}$$

with:

- δ_x the Dirac measure at position x
- $\mathbf{g}_i$ the weight associated with point $x_{G,i}$ such that $\sum_i^n \mathbf{g}_i = 1$
- $\mathbf{b}_j$ the weight associated with point $x_{B,j}$ such that $\sum_j^m \mathbf{b}_j = 1$

The objective here is to find the optimal coupling matrix $\mathbf{P} \in \mathbb{R}_+^{n \times m}$ from the set of admissible couplings $\mathbf{U}(\mathbf{g}, \mathbf{b})$, where $P_{i,j}$ describes the amount of mass flowing from bin i to bin j:

$$\mathbf{U}(\mathbf{g}, \mathbf{b}) := \left\{ \mathbf{P} \in \mathbb{R}_+^{n \times n} : \mathbf{P} \mathbb{1}_m = \mathbf{g} \qquad \text{and} \qquad \mathbf{P}^\top \mathbb{1}_m = \mathbf{b} \right\} \tag{68}$$

Formally, this amounts to minimizing the Wasserstein Distance:

$$W(\gamma, \beta) := \min_{\mathbf{P} \in \mathbf{U}(\mathbf{g}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle = \sum_{i,j} \mathbf{C}_{i,j} \mathbf{P}_{i,j} \tag{69}$$

with:

- the operator $\langle ., . \rangle$ denoting the Frobenius Inner Product:

$$\langle \mathbf{A}, \mathbf{B} \rangle := \mathrm{Tr}\left(\mathbf{A}^\top \mathbf{B}\right) := \sum_{i,j} \mathbf{A}_{i,j} \mathbf{B}_{i,j} \tag{70}$$

- $\mathbf{C} \in \mathbb{R}^{n \times m}$ the "Cost Matrix" which describes the cost of moving an elementary mass from x_i to x_j.

[76] Note that the mini-batches do not need to be of equal size.

- $\mathbf{P} \in \mathbb{R}_+^{n \times m}$ the "Coupling Matrix".

In practice, solving the transport problem exactly (i.e. finding the best $\mathbf{P}$) is difficult, but good approximations can be computed efficiently by introducing the concept of entropic regularization. The discrete entropy of a coupling matrix $\mathbf{P}$ is defined as:

$$\mathbf{H}(\mathbf{P}) := -\sum_{i,j} \mathbf{P}_{i,j} \left(\log \left(\mathbf{P}_{i,j} \right) - 1 \right) \tag{71}$$

Adding this term to the objective changes it slightly:

$$W^{\varepsilon}(\gamma, \beta) := \min_{\mathbf{P} \in \mathbf{U}(\mathbf{g}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle - \varepsilon \mathbf{H}(\mathbf{P}) \tag{72a}$$

$$= \min_{\mathbf{P} \in \mathbf{U}(\mathbf{g}, \mathbf{b})} \left\{ \sum_{i,j} \mathbf{C}_{i,j} \mathbf{P}_{i,j} + \varepsilon \sum_{i,j} \mathbf{P}_{i,j} \left(\log \left(\mathbf{P}_{i,j} \right) - 1 \right) \right\} \tag{72b}$$

The solution to this new transport objective is unique and has the form: $\mathbf{P}_{i,j} = \mathbf{u}_i \mathbf{K}_{i,j} \mathbf{v}_j$ for two non-negative vectors $(\mathbf{u}, \mathbf{v}) \in \mathbb{R}_+^n \times \mathbb{R}_+^m$. $\mathbf{u}$ and $\mathbf{v}$, as well as the gradient of $W^{\varepsilon}(\gamma, \beta)$ with respect to each x_G point, can be found efficiently with the Sinkhorn algorithm (see Peyré and Cuturi [42] for a detailed overview and proofs of convergence).

Importantly, the objective of equation 72 cannot be used directly. Indeed, $W^{\varepsilon}(\gamma, \beta)$ is non-zero even when $\gamma = \beta$, causing γ to contract. This phenomenon is called the entropic bias and leads the generated distribution to be a shrunk version of the target distribution. This bias can be canceled by subtracting autocorrelation terms [17, 50]. The final objective becomes:

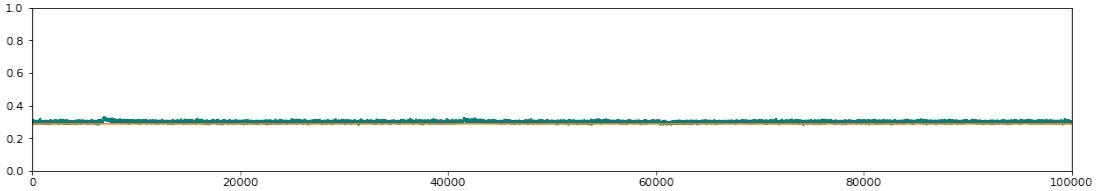
$$OT^{\varepsilon}(\gamma, \beta) = W^{\varepsilon}(\gamma, \beta) - \frac{1}{2} W^{\varepsilon}(\gamma, \gamma) - \frac{1}{2} W^{\varepsilon}(\beta, \beta) \tag{73}$$

with $\quad \forall i, \mathbf{g}_i = 1/n \quad$ and $\quad \forall j, \mathbf{b}_j = 1/m \quad$ in the data-dependent case.

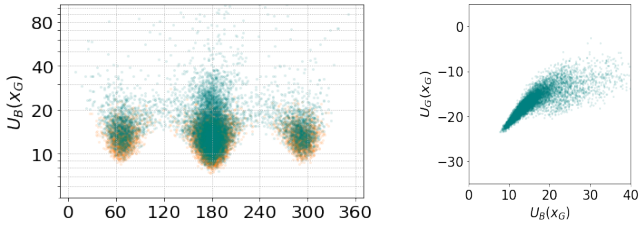
Its associated $\mathcal{L}_{OT^{\varepsilon}}^{\text{dd}}$ loss gives good results on `Double Well Medium 12D` (see appendix A.9) as well as on `Butane` (see figure 39).

One practical word of caution here is that the points of the mini-batches are mostly assigned to themselves in the coupling matrices of the autocorrelation terms. Experimental results seem to suggest that such assignations are undesirable and should be avoided to prevent a failure mode in which the autocorrelation terms ($W^{\varepsilon}(\gamma, \gamma)$ and $W^{\varepsilon}(\beta, \beta)$) do not properly cancel the entropic bias (see figure 40). This is done by tweaking the diagonal of the cost matrix so that there is an infinite distance between each point and itself, ensuring that the coupling matrix maps each point to the average of its neighbors.

Results on `Dialanine` are not as good as those on `Butane`. Although the 2D projection on the angles ϕ and ψ looks correct (see figure 41d), a large portion of the target distribution p_B is not covered by the model (see figure 41e) and some

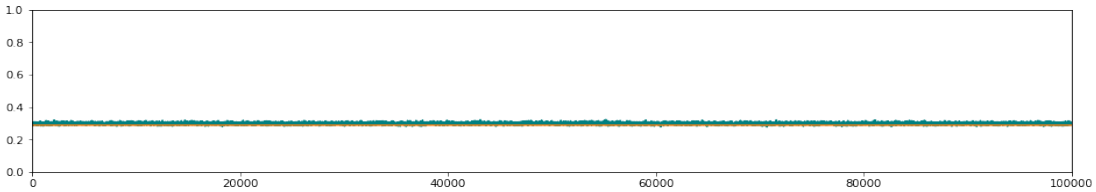


(a) Ratio of generated x_G samples in the minor modes during fine-tuning.

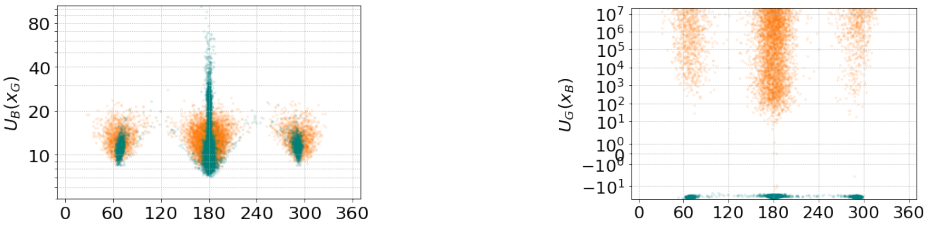


(b) U_B energies of samples $x_G \sim p_G$. (c) Correlations. (d) U_G energies of samples $x_B \sim p_B$.

Figure 39: Fine-tuning with $\mathcal{L}^{\text{dd}}_{OT^\varepsilon}$ on `Butane` (see caption of figure 17 for more details).



(a) Ratio of generated x_G samples in the minor modes during fine-tuning.



(b) U_B energies of samples $x_G \sim p_G$. (c) U_G energies of samples $x_B \sim p_B$.

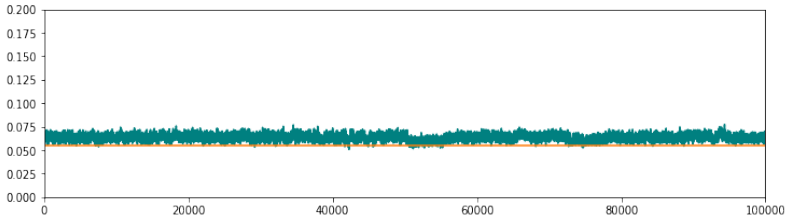
Figure 40: Fine-tuning with $\mathcal{L}^{\text{dd}}_{OT^\varepsilon}$ on `Butane` *without* tweaking the diagonal of the cost matrix of the autocorrelation terms (see caption of figure 17 for more details).

generated samples have a high potential energy U_B (see figure 41c). The cause of this poor perfomance is unclear. It could be that $\mathcal{L}^{\text{dd}}_{OT^\varepsilon}$, as it is currently defined, simply performs worse on more complex problems, and it could also be that some hyperparameters (like ε or the number of Sinkorn iterations) need to be tuned more precisely.

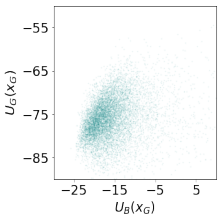
8.2 Data-free case

In the data-free case, the same training objective is used (see equation 73) but β (from equation 67) becomes:

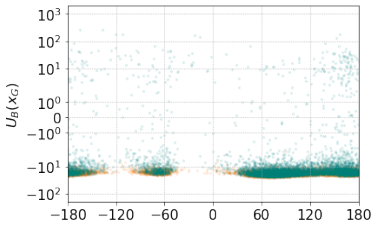
$$\beta = \sum_{i=1}^{m} \mathbf{b}_j \delta_{x_{G,j}} \tag{74}$$



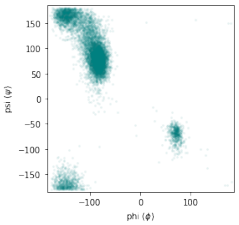
(a) Ratio of generated x_G samples in the minor modes.



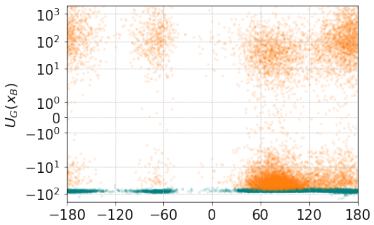
(b) Correlations.



(c) U_B energies of samples $x_G \sim p_G$.



(d) 2D projection



(e) U_G energies of samples $x_B \sim p_B$.

Figure 41: Fine-tuning with $\mathcal{L}^{\text{dd}}_{OT^\varepsilon}$ on `Dialanine` (see captions of figures 35, 36 and 37 for more details).

with $\mathbf{b}_j \propto p_B(x_{G,j})/p_G(x_{G,j})$ to take into account the fact that the points are now sampled according to p_G instead of p_B within the Dirac term $\delta_{x_{G,j}}$.[77]

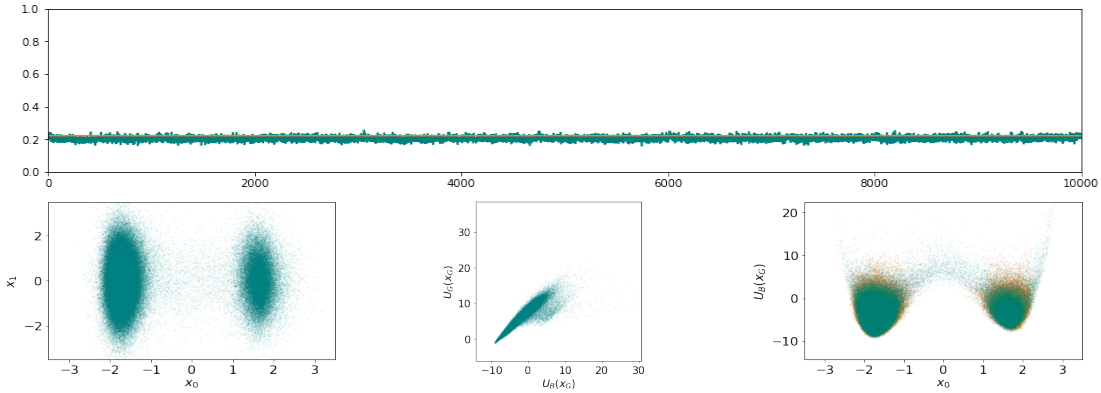


Figure 42: Fine-tuning with $\mathcal{L}^{\text{df}}_{OT^\varepsilon}$ on `Double Well Medium 12D` (see caption of figure 7 for more details).

The associated $\mathcal{L}^{\text{df}}_{OT^\varepsilon}$ loss gives good results on `Double Well Medium 12D` (see figure 42), but fails as soon as the dataset becomes slightly more complex. The potential energy U_B of the generated samples progressively increases by a few orders of magnitude during fine-tuning on `Double Well Wide 12D`, `Butane` and `Dialanine` (data not shown). More work would be needed to investigate the cause of this instability.

[77] Note that in both the data-dependent case and the data-free case $\mathbf{b}$ must verify $\sum_j^m \mathbf{b}_j = 1$.

9 Future Work

9.1 The dream

Every experiment performed in this work relies on data-dependent pre-trainings. While the methods developed here (in particular the $\mathcal{L}_{L^2_+}$ loss from section 7.1) allow for stable optimization of a correctly trained model, lifting the requirement for reference data requires 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 [13], enhanced-sampling simulations [23], or hybrid approaches [18].
- Alternately, the cost of complete training for every new target could be reduced considerably by transferring information between systems using curriculum learning. In the case of molecular targets, this would require a single model able to generate the configurations of several molecules (e.g. one based on graph convolutions [35, 45, 56]).

Ultimately, this last point would be truly transformative. This section 9 discusses what future work would be needed to train on all molecules, fully in data-free and with a model and training procedure that scales:

1. **Data-free trainings with hierachical architectures.** Although data-free fine-tunings have been demonstrated in this work (for the first time, to the best of our knowledge), hydrogen atoms still need to be removed on molecular systems to avoid a combinatorial explosion of the number of modes in the target distribution p_B. This requirement causes the ratio between the modes to change (see section 3.3 and appendix A.2), but it can be lifted with hierachical architectures (see section 9.2) by leveraging the collapsing behavior of $\mathcal{L}_{KLx}$. Hierachical architectures are also a fundamental ingredient of methods that scale to very large systems (like proteins).
2. **Exploration to discover missing modes.** Although the $\mathcal{L}_{L^2_+}$ loss is extremely stable and manages to recover the correct mode ratios from pre-trained models that are imperfect[78], if a mode is *completely* missing, it cannot be discovered. This is a significant problem if the pre-training is particularily poor (or in the context of curriculum learning), but the $\mathcal{L}_{L^2_+}$ loss

[78] This is demonstrated in several experiments. The one from figure 38 shows that correct mode ratios are recovered from incorrect ones at the end of the pre-training. The one that decreases λ_{align} by a factor 10 during the fine-tuning shows that $\mathcal{L}_{L^2_+}$ can expand the width of the distribution within known modes. The one that increases λ_{align} by a factor 10 shows that $\mathcal{L}_{L^2_+}$ can also narrow of the distribution where needed.

seems compatible with many approaches inspired from the field of Molecular Dynamics that could address that problem (see section 9.3).

3. **Scaling with Transfer and Curriculum Learning.** In order to remove the pre-training requirement altogether, curriculum learning offers a good strategy in which models trained on larger systems would be initialized with models trained on smaller systems.[79] A prerequisite to enable this strategy is to have a model architecture that can be applied to several molecules (i.e. with different chemical formulas). This is a form of transfer learning that is explored in section 9.4 with equivariant models (in particular with GraphNN and attention layers).

9.2 Data-free trainings with hierachical architectures

In the context of molecular systems, not being able to generate hydrogen atoms is an important limitation of the current approach (since it distorts the target ratio between the modes, see appendix A.2). As mentioned in section 3.3, a better solution to placing the hydrogens atoms deterministically near their energy minimum would be to place them stochastically, directly with a Normalizing Flow.[80] This is not so easy to do in practice since hydrogen permutations (in CH_2 and CH_3 groups) cause a combinatorial explosion of the number of modes. But since the energy landscape is perfectly symmetrical under hydrogen permutations, generating only one of those "hydrogen modes" would be sufficient. And there is already a tool that enables the convergence of a model toward only one mode: it is the mode collapse induced by the $\mathcal{L}_{KLx}$ loss showcased in section 4.

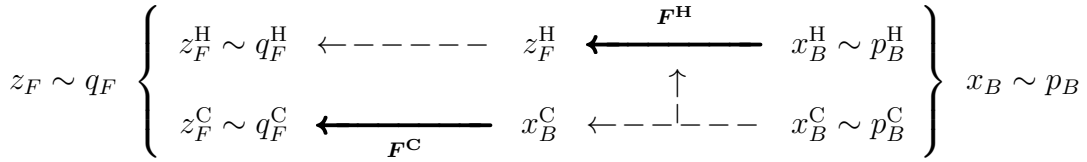
The challenge now is to separate the generative model in two stages (denoted G^{C} and G^{H} to stay consistent with the notation of section 3.3) so that heavy atoms generated by G^{C} are trained with $\mathcal{L}_{L^2_+}$ and hydrogen atoms generated with by G^{H} are trained with $\mathcal{L}_{KLx}$. This would ensure mode collapse *only* on the hydrogen atoms and can be achieved by making the second stage G^{H} *conditional* on the output of the first one. Conditional Normalizing Flows [57] can be implemented with Coupling Blocks by simply ensuring that their parameterized functions (denoted M in section 3.2.4) take two inputs: half of the features of each layer (like before) plus the condition (which in this case would be the position of the heavy atoms).

The following diagrams represent the forward and inverse passes respectively:

$$z_{\mathcal{N}} \sim q_{\mathcal{N}} \left\{ \begin{array}{ccccc} z^{\mathrm{H}}_{\mathcal{N}} \sim q^{\mathrm{H}}_{\mathcal{N}} & - - - - - \rightarrow & z^{\mathrm{H}}_{\mathcal{N}} & \xrightarrow{\boldsymbol{G^{\mathrm{H}}}} & x^{\mathrm{H}}_G \sim p^{\mathrm{H}}_G \\ & & & & \uparrow \\ z^{\mathrm{C}}_{\mathcal{N}} \sim q^{\mathrm{C}}_{\mathcal{N}} & \xrightarrow[\boldsymbol{G^{\mathrm{C}}}]{} & x^{\mathrm{C}}_G & - - - \rightarrow & x^{\mathrm{C}}_G \sim p^{\mathrm{C}}_G \end{array} \right\} x_G \sim p_G$$

[79] Note that this approach is likely to miss modes, hence the need for exploration

[80] At the price of adding some variance to the target energy U_B.



with the thick arrows representing the two flow-based stages of model and the dashed upward arrows representing the conditioning of the second stage.

Interestingly, this hierachical architectures can be extended to any arbitrary number of stages. In the case of proteins for example, one can imagine a three-stage model: one stage for the protein backbone, one stage for the heavy atoms of the amino acids and one stage for the hydrogens.

9.3 Exploration to discover missing modes

When sampling mini-batches, in any data-free training, regions of the space that are not sampled by p_G are not be penalized by the loss function, and are therefore never recovered. This is a fundamental problem that makes data-dependent pre-trainings mandatory, since the model needs to already sample each mode sufficiently well *before* fine-tuning. To lift this contraint, better exploration strategies need to be investigated.

The issue is made even more critical when using $\mathcal{L}_{L^2_+}$, and even $\mathcal{L}^{\mathrm{df}}_{KLz}$, since neither loss nor their variants explore x-space properly beyond the modes that are already covered. Fortunately during training, all these losses use the inverse pass of the model over $\mathbf{x}_G^\ddagger$ mini-batches, which could be a significant advantage during exploration. Indeed one can take advantage of this by simply expanding the mini-batches with additional candidate points generated with any exploration strategy (as long as the distribution that produce those candidate points converges to p_G).[81]

Many exploration methods have been developed over the years in the field of Molecular Dynamics (see section 1.2 and Hénin et al. [23] for a review). Recently, Gabrié et al. [18] has shown how Adaptive Monte Carlo methods can be augmented with Normalizing Flows to enhance the sampling of high-dimensional distributions. Interestingly, this uses the generative model to accelerate the sampling, and uses the newly sampled data to adapt the generative model, thereby showing how the two methods can seamlessly be integrated with one another.

Other exploration strategies may consist in using the local entropy term of $\mathcal{L}_{KLx}$ in some sort of Simulated Annealing strategy or seeding the model with known low-energy points.[82]

[81] A similar approach is already used successfully when using the cache introduced in section 6.4.4, although it is not for exploration but rather to increase stability during training.

[82] Note that, when combining Normalizing Flow losses, they should have the same target. So anchoring at the bottom of each mode should only be temporary.

9.4 Scaling with Transfer and Curriculum Learning

Another promising avenue of research is to design a *single* model to generate the distributions of multiple molecules (i.e. with different chemical formulas). Beyond the obvious benefit of not having to retrain the model on each new molecule, the hope is to transfer knowledge from molecules of known configurations to molecules more poorly described. Given that biomolecules share a common set of substructures, this would leverage the generalization capabilities of Neural Networks via transfer learning.

Such a model would also enable the use of curriculum learning [6, 19, 46] which can be interpreted as a way to constantly fine-tune the same model on larger and larger systems, making the task more complex as training progresses. Efficient exploration methods would be mandatory to discover new modes gradually on the larger systems.

The Normalizing Flow architecture used in work can be modified to work on molecules of different topologies by making the parameterized functions M of Coupling Blocks able to adapt to any number of atoms. This can be achieved by using GraphNN and attention layers (see sections 9.4.1 and 9.4.2 respectively).[83]

9.4.1 GNNs and Path-aware convolutions

Graph Neural Networks are a convenient architectural framework that would support different molecular graphs during generation (see Battaglia et al. [4] for a review). Graph convolutions are conditioned on the structure of the molecule (described by its sparse adjacency matrix A^{mol}) and are applicable on $\left\{z_{\mathcal{N}}, b^{\text{mol}}\right\}$ directly (with b^{mol} the set of additional information about the molecule such as atom types, bond types etc.).

A naïve approach consisting in simply stacking multiple GAT layers [56] is in principle able to solve the generation problem (given a deep enough model). In practice, conventional graph convolutions suffer from a lack of expressivity for two important reasons. First, their receptive field is limited to neighbors of degree one. And second, most of the computation is usually performed on the nodes with a ligthweight aggregation function (such as a sum) to make the nodes communicate.

To address the first problem, one needs to consider the reason why a larger receptive field may be needed. A quick look at the terms of the energy function U_B suggests that the model needs to be able to compute bonds, angles, dihedrals and impropers internally (cf. section 9.4.2 for nonbonded interactions), and therefore requires receptive fields of size at least three with access to the intermediary nodes. This motivates the following *Path-aware convolution* which is adapted to computing

[83]Note that the Schnet [45], although it shows great performance on energy prediction, would not be usable easily for generation since it relies on radial convolutions that depend on the very distances between atoms that the model is supposed to predict.

the energy terms of interest but still general enough to be used in other contexts:

$$\begin{aligned} v_0^{(l+1)} = [&\mathrm{MFF}_0(v_0), \\ &\sum_{v_1} \mathrm{MFF}_1([v_0, v_1]), \\ &\sum_{v_2} \mathrm{MFF}_2([v_0, v_1, v_2]), \\ &\sum_{v_3} \mathrm{MFF}_3([v_0, v_1, v_2, v_3])] \end{aligned}$$

with:

- $v_0^{(l+1)}$ the features of the central node at the next layer
- v_0, v_1, v_2 and v_3 the features of the central node (i.e. atom) and its neighbors of degree 1, 2 and 3
- $[v_0, v_1, v_2, v_3]$ the concatenation of the features of all the nodes that lead to v_3 (aka. the "path" to v_3, hence the name of the convolution)
- MFF a "Multiplicative Feed-Forward" described below
- $[\cdot, \cdot]$ the concatenation operator

This establishes a change of paradigm when compared to conventional graph convolutions in the sense that most of the computation is performed on the *interactions* between *multiple* nodes, which makes this approach more similar to Hypergraph Convolutions [1].

The last thing to consider is to make the *interaction layer* (denoted MFF here) sufficiently expressive by being able to make products (which are needed to compute euclidean distances for example). This can be achieved by using a simple feed-forward Neural Network with GLU non-linearities [12] defined as:

$$\mathrm{GLU}(a, b) = a \otimes S(b) \tag{75}$$

with:

- a and b two feature vectors of the same size (usually coming from two different linear transformations of the features of the layer below)
- $S(x) = \frac{1}{1+e^{-x}}$ the sigmoid function
- $\otimes$ the element-wise product

Note that using Graph layers within Coupling Blocks requires cutting the features *of each atom* in 2 [35], which means cutting 3 features in 1 + 2 at each layer since atoms live in 3D space. It is unclear if this limitation could be a significant problem, but Augmented Normalizing Flows [21] may provide a solution to this.

9.4.2 Long-distance attention

Finally, when working on larger systems like proteins, the model needs to be able to capture long-distance interactions if, for example, one terminus of a polypeptide chain interacts with the other. Those interactions are constrained by the energy

terms $U_{\text{nb}}(r)$ and $U_{\text{coulomb}}(r)$. Since not every section of the graph interacts with every other, an attention mechanism seems particularly adapted to this task.

One major limitation of attention mechanisms is that they are typically in the order of n_k^2 (with n_k the number of nodes). A solution to this problem consists in defining a two-steps attention layer that is in the order of $2n_q n_k$ with n_q a fixed hyperparameter corresponding to the number of hubs through which nodes communicate.

As defined in Vaswani et al. [55], the output matrix of an attention layer is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}}\right)V \tag{76}$$

with:

- Q the matrix of shape $n_q \times d_k$ corresponding to "queries"
- K the matrix of shape $n_k \times d_k$ corresponding to "keys"
- V the matrix of shape $n_k \times d_v$ corresponding to "values"
- n_k and n_q the number of nodes and the number of queries respectively
- d_k and d_v the number of features for the keys and values
- $\text{Attention}(Q, K, V)$ the output of shape $n_q \times d_v$

Ignoring Multi-Head Attention for simplicity, the *Long-Distance Attention* using hubs can be defined as:

$$\begin{aligned} Q^{\text{nodes}}, K^{\text{nodes}}, V^{\text{nodes}} &= \text{Linear}(F^{(l)}) \\ F^{\text{hubs}} &= \text{Attention}(Q^{\text{hubs}}, K^{\text{nodes}}, V^{\text{nodes}}) \\ K^{\text{hubs}}, V^{\text{hubs}} &= \text{FF}(\text{Linear}(F^{\text{hubs}})) \\ F^{(l+1)} &= \text{Attention}(Q^{\text{nodes}}, K^{\text{hubs}}, V^{\text{hubs}}) \end{aligned}$$

with:

- $F^{(l)}$ and $F^{(l+1)}$ the features at layers l and $l+1$ respectively
- Linear() a linear transformation of the input features
- FF() some feed-forward layer
- Q^{nodes}, K^{nodes}, V^{nodes} the queries, keys and values associated with the nodes.
- Q^{hubs}, K^{hubs}, V^{hubs} the queries, keys and values associated with the hubs.

Importantly here, Q^{hubs} is the only matrix that is a learned parameter of the model.

This newly defined Long-Distance Attention is reminiscent of the Agglomerative Attention of Spellings [47] and supports sparse communication between nodes.

9.5 Practical recommendations

Beyond the obvious guidelines that are general to any deep learning training procedure (data should be properly normalized, learning rates should be chosen

carefully, etc.), this sections lists a few practical recommendations that seem to be of importance when training flow-based models on Boltzmann distributions.

9.5.1 Degrees of freedom:

- As illustrated in section 4, avoiding unnecessary symmetries within the target distribution is often beneficial to ease the training. Hydrogen atoms for instance are permutation invariant within CH_3 groups and thus each group multiplies by 6 the total number of modes. 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 hydrogen atoms deterministically near their energy minimum at the cost of intruducing a bias that changes the ratio between modes (see section 3.3). 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 (see section 9.2).
- 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 2.4), but it is also possible to generate configurations in internal coordinates directly (thereby removing 6 degrees of freedom).

9.5.2 Numerical instabilities:

- The $\mathcal{L}_{KLz}^{\mathrm{df}}$ loss 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 (see section 5.3 and figure 38i). It is better to avoid using such weights during the design of new loss functions (as it is the case with the loss $\mathcal{L}_{L_+^2}$ developed in section 7).
- 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 (see section 2.3). This approach is much more fine-grained than using a global capping, directly on U_B.

9.5.3 Minimizing vs. maximizing the energy terms:

- The U_B term should probably never be increased explicitly through gradient descent (which is equivalent to saying that p_B should never be decreased directly). 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 applied to discrete mini-batches (see section 6.1).

- 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 (see section 6.5.3).

9.6 Conclusion

This work explores the conditions necessary for refining flow-based models based on an explicitly known target distribution, rather than pre-determined samples. Several losses that may seem appropriate in theory lead to numerical failures in a discrete setting. In particular, optimizing the KL divergence $KL(p_G||p_B)$ between the generated and target distributions leads to major instabilities, including mode collapse in most cases. 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. Using this insight, a stable data-free loss (denoted $\mathcal{L}_{L^2_+}$) is developed based on an estimator variance minimization approach. This loss is the first one to exhibit stable data-free optimization on the Boltzmann distribution of `Dialanine`, which is a good benchmark for small molecules of pharmacological interest, and a smaller proof of concept for proteins.

In a broader context, this work is an important milestone paving the way toward the training of a single deep learning model used to generate the Boltzmann distributions of multiple molecules without any data requirement (see section 9.1).

Acknowledgments

To Guillaume Charpiat and Jérôme Hénin (in no particular order) for their invaluable insight and precious guidance, scientific and otherwise. To Bruno Raffin for his trust and wise advice. It is truly thanks to the three of them that those four years have been such a precious part of my life.

To every member of the jury, in particular the reviewers Tony Lelièvre and Eric Vanden-Eijnden, for their time and interest.

To Louis Dumont for his work on path-aware convolutions and many rich conversations that helped develop earlier ideas behind this manuscript.

To family and friends for their love and support, and especially to Mathis Felardos for his intuition, fruitful comments and inspiration.

To all of you, thank you. I feel immensely grateful and hope to be able to repay the debt.

It is hard to overstate how much we owe to others.

References

[1] S. Bai, F. Zhang, and P. H. S. Torr. Hypergraph convolution and hypergraph attention, 2019. URL(s): `https://arxiv.org/abs/1901.08150`.

[2] A. Barducci, G. Bussi, and M. Parrinello. Well-tempered metadynamics: A smoothly converging and tunable free-energy method, 2008. URL(s): `https://arxiv.org/abs/0803.3861`.

[3] J. T. Barron. Continuously differentiable exponential linear units, 2017. URL(s): `https://arxiv.org/abs/1704.07483`.

[4] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks, 2018. URL(s): `https://arxiv.org/abs/1806.01261`.

[5] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97. PMLR, 2019. URL(s): `https://arxiv.org/abs/1811.00995` and `https://proceedings.mlr.press/v97/behrmann19a.html`.

[6] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning, 01 2009. URL `https://doi.org/10.1145/1553374.1553380`.

[7] R. T. Q. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL(s): `https://arxiv.org/abs/1906.02735` and `https://proceedings.neurips.cc/paper/2019/file/5d0d5594d24f0f955548f0fc0ff83d10-Paper.pdf`.

[8] J. Comer, J. C. Gumbart, J. Hénin, T. Lelièvre, A. Pohorille, and C. Chipot. The adaptive biasing force method: Everything you always wanted to know but were afraid to ask. *The Journal of Physical Chemistry B*, 119, 2015. doi: 10.1021/jp506633n. https://hal.archives-ouvertes.fr/hal-01238593.

[9] F. Cérou, A. Guyader, and M. Rousset. Adaptive Multilevel Splitting: Historical Perspective and Recent Results. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29, 2019. doi: 10.1063/1.5082247. https://hal.archives-ouvertes.fr/hal-01936611.

[10] F. Cérou, A. Guyader, and M. Rousset. Adaptive multilevel splitting: Historical perspective and recent results. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(4), Apr. 2019. ISSN 1054-1500. doi: 10.1063/1.5082247. URL https://aip.scitation.org/doi/abs/10.1063/1.5082247.

[11] E. Darve, D. Rodríguez-Gómez, and A. Pohorille. Adaptive biasing force method for scalar and vector free energy calculations. *The Journal of chemical physics*, 128, 05 2008. doi: 10.1063/1.2829861.

[12] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70. PMLR, 2017. URL(s): https://arxiv.org/abs/1612.08083 and https://proceedings.mlr.press/v70/dauphin17a.html.

[13] M. Dibak, L. Klein, and F. Noé. Temperature steerable flows and boltzmann generators, 2021. URL(s): https://arxiv.org/abs/2108.01590 and https://arxiv.org/abs/2012.00429.

[14] B. M. Dickson. Approaching a parameter-free metadynamics, 2011. URL(s): https://arxiv.org/abs/1106.4994.

[15] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp, 2016. URL(s): https://arxiv.org/abs/1605.08803 and https://openreview.net/forum?id=SyPNSAW5.

[16] D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives, 2005. URL(s): https://arxiv.org/abs/physics/0508111.

[17] J. Feydy, T. Séjourné, F.-X. Vialard, S.-i. Amari, A. Trouve, and G. Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89. PMLR, 2019. URL(s): https://arxiv.org/abs/1810.08278 and https://proceedings.mlr.press/v89/feydy19a.html.

[18] M. Gabrié, G. M. Rotskoff, and E. Vanden-Eijnden. Adaptive monte carlo augmented with normalizing flows, 2022. URL(s): https://arxiv.org/abs/2105.12603.

[19] G. Hacohen and D. Weinshall. On the power of curriculum learning in training deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97. PMLR, 2019. URL(s): https://arxiv.org/abs/1904.03626 and https://proceedings.mlr.press/v97/hacohen19a.html.

[20] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernandez-Lobato, and R. Turner. Black-box alpha divergence minimization. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48. PMLR, 2016. URL(s): https://arxiv.org/abs/1511.03243 and https://proceedings.mlr.press/v48/hernandez-lobatob16.html.

[21] C.-W. Huang, L. Dinh, and A. Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models, 2020. URL(s): https://arxiv.org/abs/2002.07101 and https://openreview.net/forum?id=FRHsZ62_BTO.

[22] J. Hénin, G. Fiorin, C. Chipot, and M. L. Klein. Exploring multidimensional free energy landscapes using time-dependent biases on collective variables. *Journal of Chemical Theory and Computation*, 6(1), 2010. doi: 10.1021/ct9004432. URL https://doi.org/10.1021/ct9004432. PMID: 26614317.

[23] J. Hénin, T. Lelièvre, M. R. Shirts, O. Valsson, and L. Delemotte. Enhanced sampling methods for molecular dynamics simulations, 2022. URL(s): https://arxiv.org/abs/2202.04164.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. URL(s): https://arxiv.org/abs/1412.6980.

[25] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL(s): https://arxiv.org/abs/1807.03039 and https://proceedings.neurips.cc/paper/2018/hash/d139db6a236200b21cc7f752979132d0-Abstract.html.

[26] A. Krämer, J. Köhler, and F. Noé. Training invertible linear layers through rank-one perturbations, 2020. URL(s): https://arxiv.org/abs/2010.07033 and https://openreview.net/forum?id=RayUtcIlGz.

[27] J. Kästner. Umbrella sampling. *WIREs Computational Molecular Science*, 1(6), 2011. ISSN 1759-0884. doi: 10.1002/wcms.66. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wcms.66.

[28] J. Köhler, L. Klein, and F. Noe. Equivariant flows: Exact likelihood generative learning for symmetric densities. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119. PMLR, 2020. URL(s): https://arxiv.org/abs/2006.02425 and https://proceedings.mlr.press/v119/kohler20a.html.

[29] J. Köhler, A. Krämer, and F. Noe. Smooth normalizing flows. In *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021. URL(s): https://arxiv.org/abs/2110.00351 and https://proceedings.neurips.cc/paper/2021/hash/167434fa6219316417cd4160c0c5e7d2-Abstract.html.

[30] A. Laio and M. Parrinello. Escaping free-energy minima, 2002. URL(s): https://arxiv.org/abs/cond-mat/0208352.

[31] C. A. Laughton. A study of simulated annealing protocols for use with molecular dynamics in protein structure prediction. *Protein Engineering, Design and Selection*, 7(2), 02 1994. ISSN 1741-0126. doi: 10.1093/protein/7.2.235. URL https://doi.org/10.1093/protein/7.2.235.

[32] S. Lawrence, I. Burns, A. Back, A. C. Tsoi, and C. L. Giles. Neural network classification and prior class probabilities, 1998. URL https://ro.uow.edu.au/eispapers/271/.

[33] H. Lee, C. Pabbaraju, A. P. Sevekari, and A. Risteski. Universal approximation using well-conditioned normalizing flows. In *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021. URL(s): https://arxiv.org/abs/2107.02951 and https://proceedings.neurips.cc/paper/2021/hash/69ec5030f78a9b735402d133317bf5f6-Abstract.html.

[34] Y. Li and R. E. Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL(s): https://arxiv.org/abs/1602.02311 and https://proceedings.neurips.cc/paper/2016/hash/7750ca3559e5b8e1f44210283368fc16-Abstract.html.

[35] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky. Graph normalizing flows. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL(s): https://arxiv.org/abs/1905.13177 and https://proceedings.neurips.cc/paper/2019/hash/1e44fdf9c44d7328fecc02d677ed704d-Abstract.html.

[36] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, and M. Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *The Journal of Physical Chemistry B*, 102(18), 1998. doi: 10.1021/jp973084f. URL https://doi.org/10.1021/jp973084f. PMID: 24889800.

[37] B. K. Miller, M. Geiger, T. E. Smidt, and F. Noé. Relevance of rotationally equivariant convolutions for predicting molecular properties, 2020. URL(s): https://arxiv.org/abs/2008.08461.

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

[39] F. Noé, S. Olsson, J. Köhler, and H. Wu. Boltzmann generators – sampling equilibrium states of many-body systems with deep learning, 2018. URL(s): https://arxiv.org/abs/1812.01729.

[40] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2019. URL(s): https://arxiv.org/abs/1912.02762.

[41] R. Parr and Y. Weitao. *Density-Functional Theory of Atoms and Molecules*. International Series of Monographs on Chemistry. Oxford University Press, 1994. ISBN 9780195357738. URL https://books.google.ch/books?id=mGOpScSIwU4C.

[42] G. Peyré and M. Cuturi. Computational optimal transport, 2018. URL(s): https://arxiv.org/abs/1803.00567.

[43] V. G. Satorras, E. Hoogeboom, F. B. Fuchs, I. Posner, and M. Welling. E(n) equivariant normalizing flows, 2021. URL(s): https://arxiv.org/abs/2105.09016.

[44] V. G. Satorras, E. Hoogeboom, and M. Welling. E(n) equivariant graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139. PMLR, 2021. URL(s): https://arxiv.org/abs/2102.09844 and https://proceedings.mlr.press/v139/satorras21a.html.

[45] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL(s): https://arxiv.org/abs/1706.08566 and https://proceedings.neurips.cc/paper/2017/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html.

[46] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe. Curriculum learning: A survey, 2021. URL(s): https://arxiv.org/abs/2101.10382.

[47] M. Spellings. Agglomerative attention, 2019. URL(s): https://arxiv.org/abs/1907.06607.

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

[49] R. Swendsen and J.-S. Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57, 12 1986. doi: 10.1103/PhysRevLett.57.2607.

[50] T. Séjourné, J. Feydy, F.-X. Vialard, A. Trouvé, and G. Peyré. Sinkhorn divergences for unbalanced optimal transport, 2019. URL(s): https://arxiv.org/abs/1910.12958.

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

[52] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude., 2012.

[53] G. M. Torrie and J. P. Valleau. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*, 23(2), Feb. 1977. ISSN 0021-9991. doi: 10.1016/0021-9991(77)90121-8. URL http://www.sciencedirect.com/science/article/pii/0021999177901218.

[54] T. van Erven and P. Harremoes. Rényi divergence and kullback-leibler divergence, 2014. URL(s): https://arxiv.org/abs/1206.2459.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL(s): https://arxiv.org/abs/1706.03762 and https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[56] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2017. URL(s): https://arxiv.org/abs/1710.10903 and https://openreview.net/forum?id=rJXMpikCZ.

[57] C. Winkler, D. Worrall, E. Hoogeboom, and M. Welling. Learning likelihoods with conditional normalizing flows, 2019. URL(s): https://arxiv.org/abs/1912.00042 and https://openreview.net/forum?id=rJg3zxBYwH.

[58] H. Wu, J. Köhler, and F. Noe. Stochastic normalizing flows. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020. URL(s): https://arxiv.org/abs/2002.06707 and https://proceedings.neurips.cc/paper/2020/hash/41d80bfc327ef980528426fc810a6d7a-Abstract.html.

[59] J.-Y. Yi, J. Bernholc, and P. Salamon. Simulated annealing strategies for molecular dynamics. *Computer Physics Communications*, 66(2), Sept. 1991. ISSN 0010-4655. doi: 10.1016/0010-4655(91)90066-T. URL http://www.sciencedirect.com/science/article/pii/001046559190066T.

A Appendix

A.1 Multivariate Normal Distributions

The probability density function $q_{\mathcal{N}}$ (PDF) of a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (sometimes simply referred to as a "multivariate Gaussian") is:

$$q_{\mathcal{N}} = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \cdot e^{-\frac{1}{2}(z-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(z-\boldsymbol{\mu})} \tag{77}$$

with:

- z: a c-dimensional random vector
- $\boldsymbol{\mu}$: a c-dimensional mean vector
- $\boldsymbol{\Sigma}$: a $c \times c$ covariance matrix

If the distribution is *centered*, by definition $\boldsymbol{\mu}$ is equal to the zero vector $\mathbf{0}$ and so the PDF of $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is:

$$q_{\mathcal{N}} = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \cdot e^{-\frac{1}{2}z^\top \boldsymbol{\Sigma}^{-1} z} \tag{78}$$

If the distribution also has a *diagonal covariance matrix* with equal diagonal entries of the form:

$$\boldsymbol{\Sigma} = \operatorname{diag}(\boldsymbol{\sigma}^2) = \begin{pmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{pmatrix} = \sigma^2 \mathrm{Id} \tag{79}$$

then the PDF of $\mathcal{N}(\mathbf{0}, \operatorname{diag}(\boldsymbol{\sigma}^2))$ is:

$$q_{\mathcal{N}} = \frac{1}{\sqrt{(2\pi\sigma^2)^c}} \cdot e^{-\frac{1}{2}z^\top \operatorname{diag}(\boldsymbol{\sigma}^2)^{-1} z} \tag{80a}$$

$$= \frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^c} \cdot e^{-\frac{1}{2}\sum_i^c \left(\frac{z_i}{\sigma}\right)^2} \tag{80b}$$

$$= \frac{1}{\left(\sigma\sqrt{2\pi}\right)^c} \cdot e^{-\frac{1}{2\sigma^2}\sum_i^c z_i^2} \tag{80c}$$

and so:

$$q_{\mathcal{N}}(z) = \frac{1}{\mathcal{Z}_{\mathcal{N}}} \cdot e^{-\frac{1}{2\sigma^2}U_{\mathcal{N}}(z)} \tag{81}$$

with:

- $\mathcal{Z}_{\mathcal{N}} = \left(\sigma\sqrt{2\pi}\right)^c$: the partition function
- $\frac{1}{2\sigma^2}$: a quantity analogous to the thermodynamic beta
- $U_{\mathcal{N}}(z) = \sum_i^c z_i^2$: the energy function of the multivariate Gaussian

A.2 Deterministic hydrogen placement changes target ratios

In the spirit of a coarse-graining 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^{all} with respect to those coordinates:

$$p_B^{\mathrm{C}}(x^{\mathrm{C}}) = \int p_B^{\mathrm{all}}(x^{\mathrm{C}}, x^{\mathrm{H}}) dx^{\mathrm{H}} \tag{82}$$

The Boltzmann probability of any given mode $\mathcal{M} \subset \mathbb{R}^{3N}$ is:

$$\begin{aligned} P_B^{\mathrm{all}}(\mathcal{M}) &= \int_{\mathcal{M}} p_B^{\mathrm{all}}(x^{\mathrm{all}}) dx^{\mathrm{all}} && \text{(83a)} \\ &= \int_{\mathcal{M}} p_B^{\mathrm{all}}(x^{\mathrm{C}}, x^{\mathrm{H}}) dx^{\mathrm{C}} dx^{\mathrm{H}} && \text{(83b)} \\ &= \int_{\mathcal{M}^{\mathrm{C}}} \left[\int p_B^{\mathrm{all}}(x^{\mathrm{C}}, x^{\mathrm{H}}) dx^{\mathrm{H}} \right] dx^{\mathrm{C}} && \text{(83c)} \\ &= \int_{\mathcal{M}^{\mathrm{C}}} p_B^{\mathrm{C}}(x^{\mathrm{C}}) dx^{\mathrm{C}} && \text{(83d)} \end{aligned}$$

where 83c uses the fact that $\mathcal{M}$ is defined solely based on the values of x^{C}, so that it can be written $\mathcal{M} = \mathcal{M}^{\mathrm{C}} \times \mathbb{R}^{3N_{\mathrm{H}}}$, with $\mathcal{M}^{\mathrm{C}}$ a set of heavy atom coordinates, and N_{H} the number of hydrogen atoms.

However, in practice, data-free optimization of G^{C} (using only samples with minimum-energy hydrogen coordinates) minimizes the divergence between p_G^{C} and an auxiliary distribution $\bar{p}_B^{\mathrm{C}}$:

$$\lim_{t\to\infty} p_{G,t}^{\mathrm{C}}(x^{\mathrm{C}}) = \bar{p}_B^{\mathrm{C}}(x^{\mathrm{C}}) = p_B^{\mathrm{all}}(G^{\mathrm{H}}(x^{\mathrm{C}})) \tag{84}$$

In general, p_B^{C} (defined as a marginal of p_B^{all}) differs from $\bar{p}_B^{\mathrm{C}}$ (defined by the energy function $U_B \circ G^{\mathrm{H}}$, i.e. by mapping the space $\mathbb{R}^{3N_{\mathrm{C}}}$ of heavy atom coordinates to a slice of the space $\mathbb{R}^{3N_{\mathrm{all}}}$ of all-atom coordinates). This difference introduces a bias in the generation of heavy atom coordinates. Furthermore, the generation density p_G^{all} of an all-atom configuration x^{all} is optimized towards:

$$\begin{aligned} \bar{p}_B^{\mathrm{all}}(x^{\mathrm{all}}) &= \bar{p}_B^{\mathrm{C}}(x^{\mathrm{C}})\; p_B^{\mathrm{all}}(x^{\mathrm{all}}|x^{\mathrm{C}}) && \text{(85a)} \\ &= p_B^{\mathrm{all}}(G^{\mathrm{H}}(x^{\mathrm{C}}))\; \delta(x^{\mathrm{all}} - G^{\mathrm{H}}(x^{\mathrm{C}})) && \text{(85b)} \\ &= p_B^{\mathrm{all}}(G^{\mathrm{H}} \circ F^{\mathrm{H}}(x^{\mathrm{all}}))\; \delta(x^{\mathrm{all}} - G^{\mathrm{H}} \circ F^{\mathrm{H}}(x^{\mathrm{all}})) && \text{(85c)} \end{aligned}$$

which is non-zero only on the minimum-energy-hydrogen manifold that is the image of $G^{\mathrm{H}} \circ F^{\mathrm{H}}$. Assuming perfect training ($p_G^{\mathrm{C}} = \bar{p}_B^{\mathrm{C}}$), we obtain the probability of the

minor mode as generated:

$$\bar{P}_B^{\text{all}}(\mathcal{M}) = \int_{\mathcal{M}} p_B^{\text{all}}(G^{\text{H}} \circ F^{\text{H}}(x^{\text{all}}))\ \delta(x^{\text{all}} - G^{\text{H}} \circ F^{\text{H}}(x^{\text{all}}))\ dx^{\text{all}} \tag{86a}$$

$$\approx \frac{1}{\sigma\sqrt{2\pi}} \int_{\mathcal{M}} p_B^{\text{all}}(G^{\text{H}} \circ F^{\text{H}}(x^{\text{all}})) \exp\left(-\frac{\|x^{\text{all}} - G^{\text{H}} \circ F^{\text{H}}(x^{\text{all}})\|^2}{2\sigma^2}\right)\ dx^{\text{all}} \tag{86b}$$

where 86b approximates the Dirac distribution with a Gaussian kernel with a well-chosen width σ to pave the way for numerical estimation. Such an estimator can be based on the dataset sampled according to p_B^{all}, thanks to the fact that the conditional Boltzmann distribution of hydrogen atom positions is peaked, that is, p_B^{all} is largest around minimal-energy hydrogen coordinates (i.e. close to where $x = G^{\text{H}} \circ F^{\text{H}}(x)$).

Equations 83a and 86b taken together lead to an importance sampling estimator[84] for this probability $\bar{P}_B^{\text{all}}(\mathcal{M})$ based on samples from the reference dataset:

$$\hat{\bar{P}}_B^{\text{all}}(\mathcal{M}) = \frac{\displaystyle\sum_{x_B^{\text{all}} \sim p_B^{\text{all}} | x_B^{\text{all}} \in \mathcal{M}} \frac{\tilde{p}_B^{\text{all}}(G^{\text{H}} \circ F^{\text{H}}(x_B^{\text{all}}))}{\tilde{p}_B^{\text{all}}(x_B^{\text{all}})} \exp\left(-\frac{\|x_B^{\text{all}} - G^{\text{H}} \circ F^{\text{H}}(x_B^{\text{all}})\|^2}{2\sigma^2}\right)}{\displaystyle\sum_{x_B^{\text{all}} \sim p_B^{\text{all}}} \frac{\tilde{p}_B^{\text{all}}(G^{\text{H}} \circ F^{\text{H}}(x_B^{\text{all}}))}{\tilde{p}_B^{\text{all}}(x_B^{\text{all}})} \exp\left(-\frac{\|x_B^{\text{all}} - G^{\text{H}} \circ F^{\text{H}}(x_B^{\text{all}})\|^2}{2\sigma^2}\right)} \tag{87}$$

[84] In practice, this estimator has a very high variance making it unreliable to assess the expected mode ratio for `Butane`. For `Dialanine` however, since the minor mode (known to biochemists as the C7ax conformation) has a very low probability, it can be estimated a bit more reliably to be close to 1%.

A.3 Functional derivative of $KL(p_G||p_B)$

A KL divergence is a *functional*, which is a rule for going from a function to a number, in the same way that a function is a rule for going from a variable to a number. Functionals are usually expressed in terms of an integral of functions, as is the case with KL divergences (the function being p_G in this case).

The functional *derivative* of the KL divergence with respect to p_G is denoted: $\nabla_{p_G} KL(p_G||p_B)$. It is the part of the difference $KL(p_G + \delta p_G||p_B) - KL(p_G||p_B)$ that depends on δp_G linearly:

$$\begin{aligned} \delta KL(p_G||p_B) &= \int \frac{\delta KL(p_G||p_B)}{\delta p_G(x)}\ \delta p_G(x)\ dx \\ &= \int \nabla_{p_G} KL(p_G||p_B)\ \delta p_G(x)\ dx \end{aligned} \tag{88}$$

One way to compute the functional derivative of $KL(p_G||p_B)$ is to just do a Taylor expansion of $KL(p_G + \delta p_G||p_B)$, keeping only the first-order terms [41]:

$$\begin{aligned} KL(p_G + \delta p_G||p_B) &= \int (p_G + \delta p_G)(x) \log \frac{(p_G + \delta p_G)(x)}{p_B(x)}\ dx \\ &= \int p_G(x) \log \frac{p_G(x)}{p_B(x)}\ dx + \int \left(1 + \log \frac{p_G(x)}{p_B(x)}\right) \delta p_G(x)\ dx + o(\delta p_G) \end{aligned} \tag{89}$$

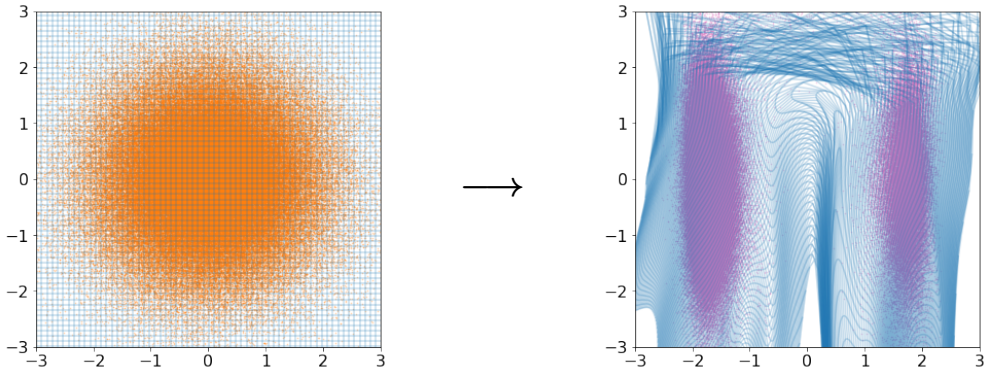
This gives:

$$\begin{aligned} \nabla_{p_G} KL(p_G||p_B) &= 1 + \log \frac{p_G(x)}{p_B(x)} \\ &= \log(p_G) - \log(p_B) + 1 \end{aligned} \tag{90}$$

And since the KL divergence is *minimized*, the quantity of interest is:

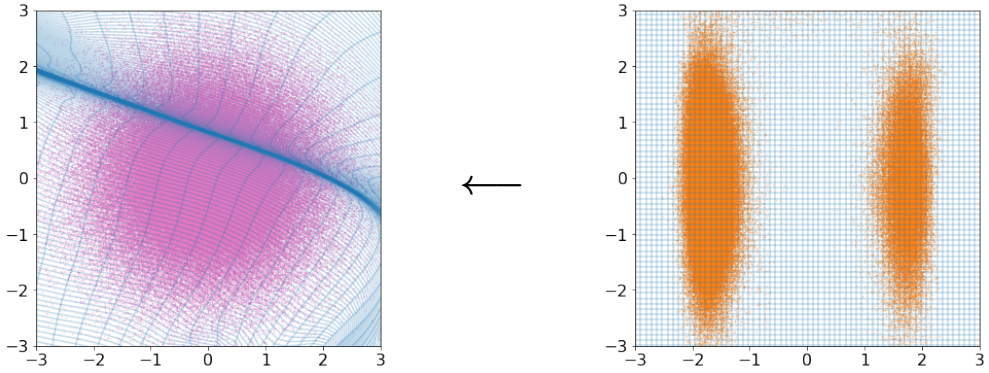
$$-\nabla_{p_G} KL(p_G||p_B) = \log(p_B) - \log(p_G) - 1 \tag{91}$$

A.4 Flow-based Transformations



(a) Input grid in z-space. (b) Output grid in x-space.

Figure 43: Deformation of the space performed by a pre-trained model on `Double Well Medium 2D`. A regular grid is defined in z-space (left) and visualized after transformation by the model in x-space (right).



(a) Output grid in z-space. (b) Input grid in x-space.

Figure 44: Deformation of the space performed by a pre-trained model on `Double Well Medium 2D`. A regular grid is defined in x-space (right) and visualized after transformation by the model in z-space (left).

A.5 Visualizing the collapse

Figure 45 is a visualization of the start of the collapse, after 20 steps of fine-tuning the model from figure 9 with $\mathcal{L}_{KLx}$. A grid is defined in x-space, along 2 dimensions (the multi-modal one in abscissa, and a unimodal one in ordinates), and the value of the 10 remaining dimensions is chosen at random (but the same 10 values are used for all the points of the grid). The result is a clear visualization that the small well moves in the direction of the larger well.

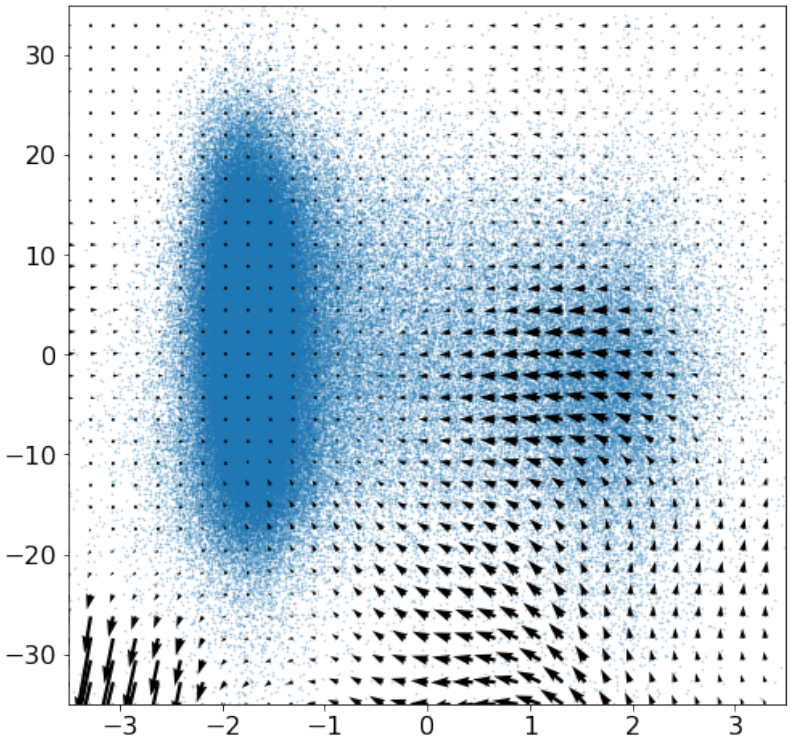


Figure 45: Displacement of the p_G distribution produced by a pre-trained model on `Double Well Wide 12D` after 20 steps of fine-tuning with $\mathcal{L}_{KLx}$.

A.6 Derivation of $KL(q_F||q_{\mathcal{N}})$

$$KL(q_F||q_{\mathcal{N}}) = \int q_F(z) \log \frac{q_F(z)}{q_{\mathcal{N}}(z)}\ dz \tag{92a}$$

$$= \int q_F(z) \log \mathcal{Z}_{\mathcal{N}}\ dz + \int q_F(z) \log \frac{q_F(z)}{\tilde{q}_{\mathcal{N}}(z)}\ dz \tag{92b}$$

$$= \log \mathcal{Z}_{\mathcal{N}} + \int q_F(z) \log \frac{q_F(z)}{\tilde{q}_{\mathcal{N}}(z)}\ dz \tag{92c}$$

$$= \log \mathcal{Z}_{\mathcal{N}} + \int p_B(x) \log \frac{p_B(x) \cdot \left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}}{\tilde{q}_{\mathcal{N}}(F(x))}\ dx \tag{92d}$$

$$= \log \mathcal{Z}_{\mathcal{N}} - S_B + \int p_B(x) \log \frac{\left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}}{\tilde{q}_{\mathcal{N}}(F(x))}\ dx \tag{92e}$$

$$= \log \mathcal{Z}_{\mathcal{N}} - S_B + \int p_B(x) \log \frac{\left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}}{e^{-\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x))}}\ dx \tag{92f}$$

$$= \log \mathcal{Z}_{\mathcal{N}} - S_B + \mathop{\mathbb{E}}_{x_B \sim p_B}\left[\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x_B)) + \log\left|\det\left(\frac{\partial F(x_B)}{\partial x_B}\right)\right|^{-1}\right] \tag{92g}$$

$$= \log \mathcal{Z}_{\mathcal{N}} - S_B + \mathop{\mathbb{E}}_{x_B \sim p_B}\left[\frac{1}{2\sigma^2}U_{\mathcal{N}}(F(x_B)) - \log\left|\det\left(\frac{\partial F(x_B)}{\partial x_B}\right)\right|\right] \tag{92h}$$

with:

- (92a) by definition of the KL divergence
- (92b) by using $q_{\mathcal{N}} = \frac{1}{\mathcal{Z}_{\mathcal{N}}}\tilde{q}_{\mathcal{N}}$
- (92c) by using $\int q_F(z)dz = 1$ (probabilities sum to one)
- (92d) by substitution of $q_F(z)$ by the change of variable formula:

$$\begin{aligned} q_F(z)\ dz &= p_B(F^{-1}(z)) \cdot \left|\det\left(\frac{\partial F^{-1}(z)}{\partial z}\right)\right|\ dz \\ &= p_B(x) \cdot \left|\det\left(\frac{\partial F(x)}{\partial x}\right)\right|^{-1}\ dz \\ &= p_B(x)\ dx \end{aligned} \tag{93}$$

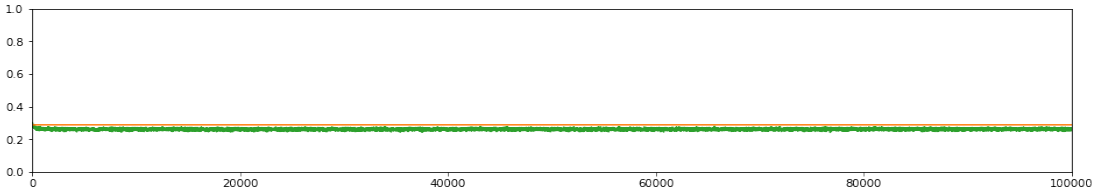
- (92e) by definition of the entropy: $S_B = S(p_B) = -\int p_B(x) \log p_B(x) dx$
- (92f) by using: $\tilde{q}_{\mathcal{N}}(z) = e^{-\frac{1}{2\sigma^2}U_{\mathcal{N}}(x)}$
- (92g) by definition of expectations: $\mathop{\mathbb{E}}_{x_B \sim p_B}\left[f(x_B)\right] = \int p_B(x) f(x) dx$

The development of $KL(p_G||p_B)$ is detailed in section 4.1.

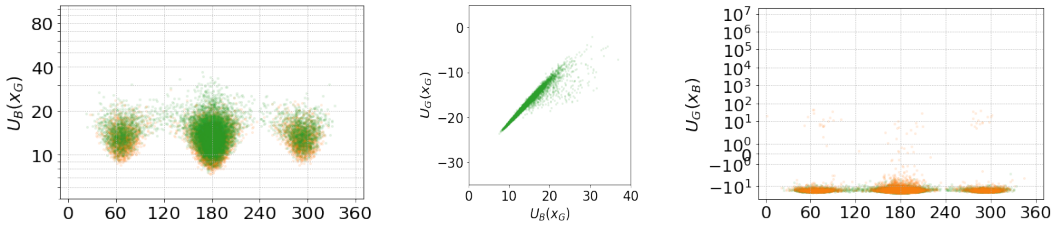
A.7 Applying $\mathcal{L}_{PL^2_{GB}}$ to `Butane`

Results with $\mathcal{L}_{PL^2_{GB}}$ and $\mathcal{L}^{\text{df}}_{KLz}$ are almost identical on `Butane` regardless of the value of λ_{align}.

With $\lambda_{\text{align}} = 0$ figure 46 should be compared with figure 18



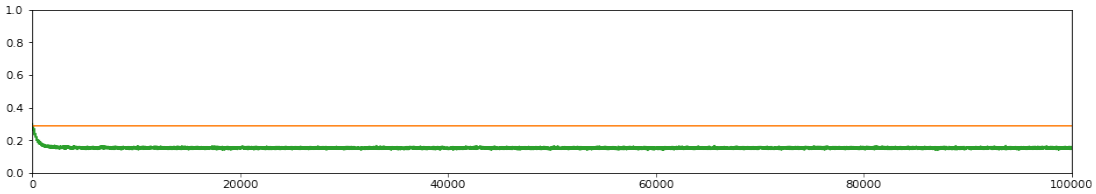
(a) Ratio of generated x_G samples in the minor modes during fine-tuning.



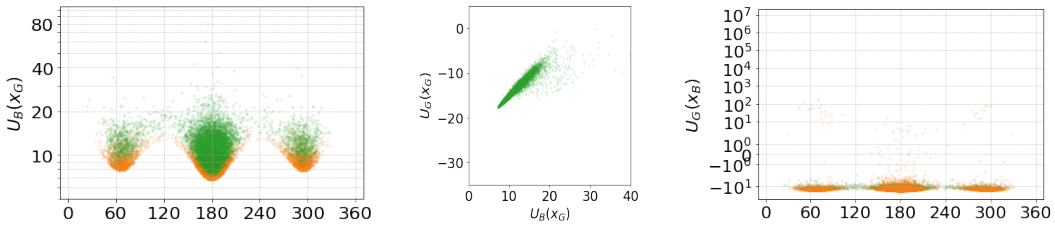
(b) U_B energies of samples $x_G \sim p_G$. (c) Correlations. (d) U_G energies of samples $x_B \sim p_B$.

Figure 46: Data-free fine-tunings on `Butane` with $\lambda_{\text{align}} = 10$. See caption of figure 17 for more details.

With $\lambda_{\text{align}} = 10$ figure 47 should be compared with figure 19.



(a) Ratio of generated x_G samples in the minor modes during fine-tuning.



(b) U_B energies of samples $x_G \sim p_G$. (c) Correlations. (d) U_G energies of samples $x_B \sim p_B$.

Figure 47: Data-free fine-tunings on `Butane` with $\lambda_{\text{align}} = 0$. See caption of figure 17 for more details.

A.8 Applying $\mathcal{L}_{PL^2_{GB}}$ to `Dialanine`

The results of the fine-tuning with $\mathcal{L}_{PL^2_{GB}}$ on `Dialanine` *without* using a cache are shown in figures 48, 49 and 50 (which are almost identical to figures 35, 36 and 37, albeit with slightly less points in the minor mode). This demonstrates that the concept of cache introduced in section 6.4.4 is not necessary to obtain a satisfactory performance with $\mathcal{L}_{PL^2_{GB}}$.

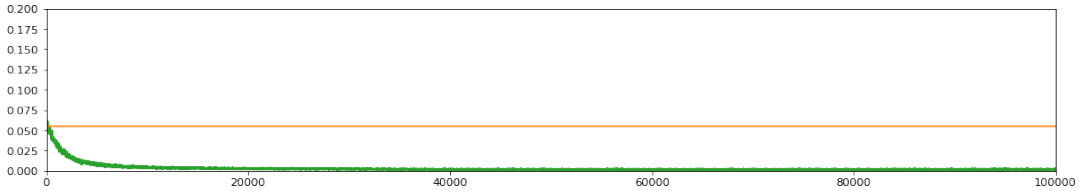
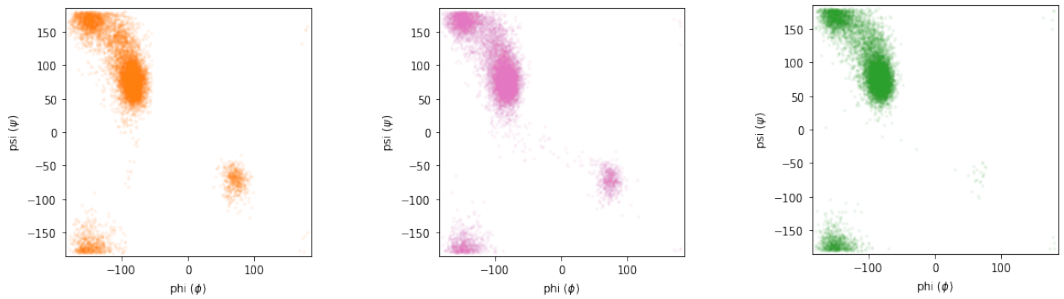


Figure 48: Ratio of generated x_G samples in the minor modes during fine-tuning with $\mathcal{L}_{PL^2_{GB}}$ on `Dialanine`. The ordinates go from 0% to 20%.

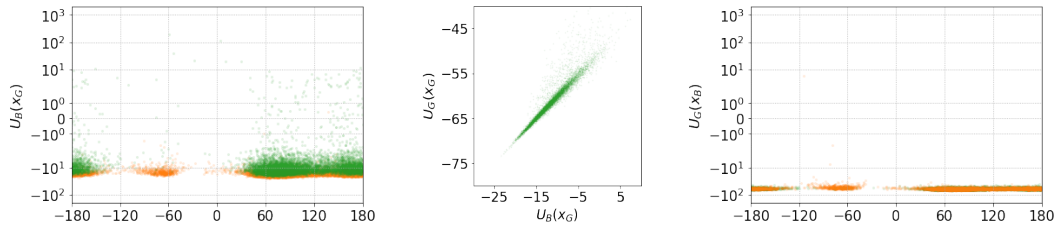


(a) ϕ and ψ from the `Dialanine` dataset.

(b) ϕ and ψ after pre-training with $\mathcal{L}^{\text{dd}}_{KLz}$.

(c) ϕ and ψ after fine-tuning with $\mathcal{L}_{PL^2_{GB}}$

Figure 49: Dihedral angles ϕ and ψ of the dialanine molecule. Figures 36a and 36b are duplicates of figures24a and 24b since the same data and pre-training are used.



(a) U_B energies of samples $x_G \sim p_G$.

(b) Correlations.

(c) U_G energies of samples $x_B \sim p_B$.

Figure 50: Quality of generation after fine-tuning on `Dialanine` with $\mathcal{L}_{PL^2_{GB}}$.

A.9 Applying $\mathcal{L}_{OT^\varepsilon}^{\mathrm{dd}}$ to `Double Well 12D`

The data-dependent $\mathcal{L}_{OT^\varepsilon}^{\mathrm{dd}}$ loss leads to good results on `Double Well Medium 12D` (see figure 51) but fails to give better results on `Double Well Wide 12D` than $\mathcal{L}_{KLz}^{\mathrm{dd}}$ (see figure 52), hinting to the fact that it may be very sensitive to proper normalization of the target distribution.

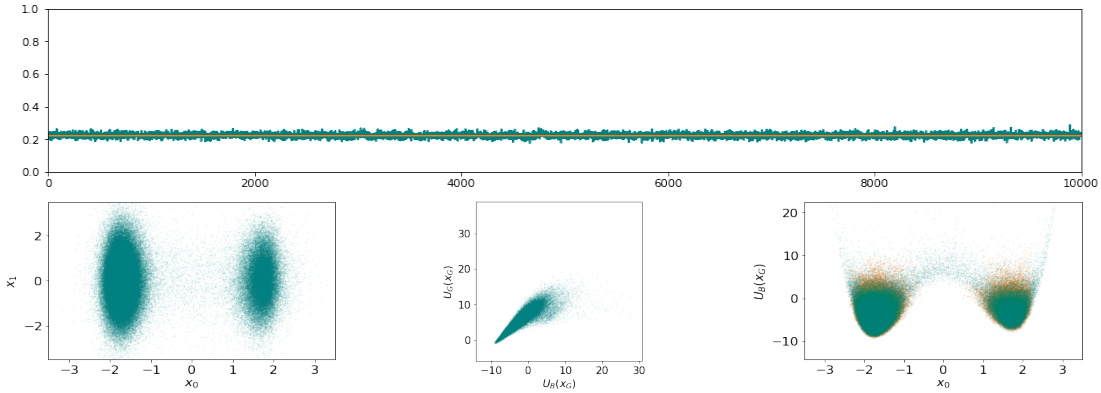


Figure 51: Fine-tuning with $\mathcal{L}_{OT^\varepsilon}^{\mathrm{dd}}$ on `Double Well Medium 12D` (see caption of figure 7 for more details).

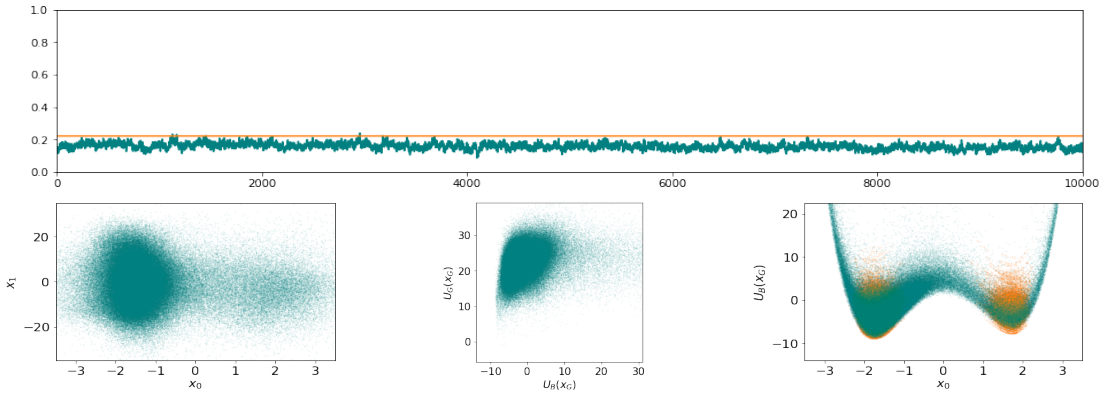


Figure 52: Fine-tuning with $\mathcal{L}_{OT^\varepsilon}^{\mathrm{dd}}$ on `Double Well Wide 12D` (see caption of figure 9 for more details).

The hyperparameters used here and in every other optimal transport experiment are as follows: $\varepsilon = 0.01$, the number of Sinkhorn iterations is 256 and the learning rate is kept at 10^{-5}.

A.10 Unbiased estimators

This section supposes that the use case of the generator G is to estimate integral quantities of the form $\mathbb{E}_{p_B}[f]$ for some given function f and denote by Q its true value[85]:

$$Q \coloneqq \mathop{\mathbb{E}}_{p_B}[f] = \mathop{\mathbb{E}}_{p_G}\left[\frac{p_B}{p_G} f\right] \tag{94}$$

In practice, one estimates Q by sampling:

$$Q \simeq \hat{Q} \coloneqq \frac{1}{n} \sum_{x_{G,i} \in \mathbf{x}_G} \frac{p_B(x_{G,i})}{p_G(x_{G,i})} f(x_{G,i}) \tag{95}$$

where $\mathbf{x}_G = (x_{G,1}, \ldots, x_{G,n})$ is a mini-batch of points sampled according to p_G.[86]

Regardless of the distribution p_G, $\hat{Q}$ is an approximation of Q, in that for very large mini-batches $\mathbf{x}_G$, i.e. large n, the estimate $\hat{Q}$ tends to Q. The estimator is said to be *unbiased* and the convergence rate typically behaves as $O(1/\sqrt{n})$. Indeed:

$$\mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}] = Q \tag{96}$$

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:

$$\mathop{\mathbb{E}}_{x_G \sim p_G}\left[\hat{Q}\right] = \mathop{\mathbb{E}}_{x \sim p_G}\left[\frac{p_B(x)}{p_G(x)} f(x)\right] = \int_{x \in X} p_B(x)\ f(x) dx =: Q \tag{97}$$

For a mini-batch $\mathbf{x}_G$ of arbitrary size n, one gets the average of n such quantities, each of which are Q on expectation, so one recovers Q again:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}] &= \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}\left[\frac{1}{n} \sum_{x_{G,i} \in \mathbf{x}_G} \frac{p_B(x_{G,i})}{p_G(x_{G,i})} f(x_{G,i})\right] && \text{(98a)} \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}\left[\frac{p_B(x_{G,i})}{p_G(x_{G,i})} f(x_{G,i})\right] && \text{(98b)} \\
&= \mathop{\mathbb{E}}_{x_G \sim p_G}\left[\frac{p_B(x_G)}{p_G(x_G)} f(x_G)\right] && \text{(98c)} \\
&=: Q && \text{(98d)}
\end{aligned}$$

[85] Equation 94 is exact provided that p_G is never 0 where p_B is not.

[86] Note however than p_B is unknown but this is not an issue since $\tilde{p}_B = \mathcal{Z}_B p_B$ is sufficient for all practical purposes.

A.11 Estimator variance as a loss

Variance of the estimator. For some distributions p_G, the estimator $\hat{Q}$ 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 $\hat{Q}$:

$$V = \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[(\hat{Q} - Q)^2] \tag{99}$$

that one would like to be as small as possible. Indeed the typical gap between an estimate $\hat{Q}$ for a mini-batch and the real value Q can be expected to be of the order of magnitude of V (by definition). The question is: Can we train p_G so as to minimize V?

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 $\mathbf{x}_G$ is:

$$\begin{aligned}
V &= \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[(\hat{Q} - Q)^2] && \text{(100a)}\\
&= \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}^2 - 2\hat{Q}Q + Q^2] && \text{(100b)}\\
&= \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}^2] - 2 \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}]Q + Q^2 && \text{(100c)}\\
&= \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}^2] - Q^2 && \text{(100d)}
\end{aligned}$$

Since Q^2 is does not depend on p_G, we aim at minimizing only:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}[\hat{Q}^2] &= \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}\left[\left(\frac{1}{n}\sum_{x_{G,i} \in \mathbf{x}_G} \frac{p_B(x_{G,i})}{p_G(x_{G,i})} f(x_{G,i})\right)^2\right] && \text{(101a)}\\
&= \frac{1}{n^2} \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}\left[\sum_{x_{G,i} \in \mathbf{x}_G} \frac{p_B^2(x_{G,i})}{p_G^2(x_{G,i})} f^2(x_{G,i})\right]\\
&\quad + \frac{1}{n^2} \mathop{\mathbb{E}}_{\mathbf{x}_G \sim p_G^n}\left[\sum_{x_{G,i}, x_{G,j} \in \mathbf{x}_G i \neq j} \frac{p_B(x_{G,i})}{p_G(x_{G,i})} f(x_{G,i}) \frac{p_B(x_{G,j})}{p_G(x_{G,j})} f(x_{G,j})\right] && \text{(101b)}\\
&= \frac{1}{n^2} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{x_{G,i} \sim p_G}\left[\frac{p_B^2(x_{G,i})}{p_G^2(x_{G,i})} f^2(x_{G,i})\right]\\
&\quad + \frac{1}{n^2} \sum_{i,j=1, i \neq j}^{n} \mathop{\mathbb{E}}_{x_{G,i} \sim p_G}\left[\frac{p_B(x_i)}{p_G(x_i)} f(x_{G,i})\right] \mathop{\mathbb{E}}_{x_{G,j} \sim p_G}\left[\frac{p_B(x_{G,j})}{p_G(x_{G,j})} f(x_{G,j})\right] && \text{(101c)}\\
&= \frac{1}{n} \mathop{\mathbb{E}}_{x_G \sim p_G}\left[\frac{p_B^2(x)}{p_G^2(x)} f^2(x)\right] + \frac{n(n-1)}{n^2} \mathop{\mathbb{E}}_{x_G \sim p_G}\left[\frac{p_B(x)}{p_G(x)} f(x)\right]^2 && \text{(101d)}\\
&= \frac{1}{n} \mathop{\mathbb{E}}_{x_G \sim p_G}\left[\frac{p_B^2(x)}{p_G^2(x)} f^2(x)\right] + \left(1 - \frac{1}{n}\right) Q^2 && \text{(101e)}
\end{aligned}$$

with:

- (101a) by definition of $\hat{Q}$
- (101c) by noticing that points $x_{G,i}$ and $x_{G,j}$ are sampled identically (according to the same law) and independently.

The variance (without forgetting the constant term $-Q^2$ from equation 100d) thus interestingly rewrites as:

$$V = \frac{1}{n} \mathop{\mathbb{E}}_{x_G \sim p_G} \left[\frac{p_B^2(x_G)}{p_G^2(x_G)} f^2(x_G) \right] - \frac{1}{n} Q^2 \tag{102}$$

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 $O(\frac{1}{\sqrt{n}})$.

Optimizing the variance with respect to p_G. Based on the variance formula above, one can consider that the expected error of the generator G can be quantified by:

$$\mathop{\mathbb{E}}_{x_G \sim p_G} \left[\frac{p_B^2(x_G)}{p_G^2(x_G)} f^2(x_G) \right] \tag{103}$$

and we would like to minimize it with respect to p_G.

If f can be any bounded function in x-space[87], then one can deduce the following optimization criterion:

$$\mathop{\mathbb{E}}_{x_G \sim p_G} \left[\frac{p_B^2(x_G)}{p_G^2(x_G)} \right] = \frac{1}{\mathcal{Z}_B^2} \mathop{\mathbb{E}}_{x_G \sim p_G} \left[e^{2(U_G - \beta U_B)} \right] \tag{104}$$

[87] Note that if the function f that needs to be integrated is known, it could be used explicitly in the criterion to optimize.