UNIVERSITY OF COPENHAGEN
DEPARTMENT OF COMPUTER SCIENCE

Ph.D. Thesis

Marloes Arts

Generative Models for Proteins: Ensembles, Dynamics and Evolution

Modelling Structure and Sequence Variation

Advisor: Wouter Boomsma

This thesis has been submitted to the Ph.D. School of The Faculty of Science, University of Copenhagen on 30 March 2023.

Preface

This Ph.D. thesis is the product of three years and seven months of research conducted at the Department of Computer Science of the University of Copenhagen, with a four month interlude for an internship at Microsoft Research in Amsterdam. This research has been made possible through the financial support of the Novo Nordisk Foundation, both individually and as part of the Center for Basic Machine Learning Research in Life Science (MLLS).

This dissertation is divided into six chapters. Chapter 1 provides an introduction and all necessary background information relevant to the topics presented in this thesis, of which an overview is given in Chapter 2. The main body of the thesis is presented in Chapter 3, Chapter 4 and Chapter 5, which contain the main scientific contributions of my Ph.D. work. Finally, I provide a summary and future research directions in Chapter 6.

Thank you for reading.

Marloes Arts
March 2023

Abstract

The work presented in this Ph.D. thesis focuses on the intersection of machine learning and protein modelling. This field is thriving due to the emergence of more powerful models and growing amounts of data, with AlphaFold2 as a recent success story in static protein structure prediction that even made its way into the global news. Nonetheless, this static view does not accurately represent the complete picture since proteins are dynamic biomolecules. The papers presented in this thesis zoom in on all dynamic aspects of proteins: structure ensembles, protein dynamics, and protein sequence evolution. All proposed methods are based on generative machine learning models such that new data points can be produced that come from approximately the same distribution as the data seen during training.

The main contributions of this dissertation are threefold. Firstly, we propose a general method to simultaneously impose local and global constraints in protein structure ensemble modelling. As a proof of principle, this method is incorporated into a simple variational autoencoder (VAE) and we demonstrate that the generated samples are of high quality, both locally and globally. The second contribution is a denoising diffusion model based method trained on reduced representations of protein structures from molecular dynamics simulations. Not only can this model produce new samples in a one-shot manner, a force field can also cheaply be extracted to perform new simulations. The final contribution is a closer investigation on the use of VAEs to model protein family sequence data. Specifically, we examine the strengths and weaknesses of Bayesian decoders in this context as well as show the potential of hierarchical VAEs to alleviate the mismatch between the commonly used standard Gaussian prior over latent space and the "star-shaped" aggregated posterior for protein family data.

Resumé

Arbejdet præsenteret i denne ph.d. afhandling fokuserer på krydsfeltet mellem maskinlæring og proteinmodellering. Dette felt blomstrer på grund af fremkomsten af mere kraftfulde modeller og voksende mængder af data med AlphaFold2 som en nylig succeshistorie inden for forudsigelse af statisk proteinstruktur, der endda fandt vej til de globale nyheder. Dette statiske billede giver dog ikke et helt retvisende billede af proteiners natur, da proteiner i virkeligheden er dynamiske biomolekyler. Artiklerne præsenteret i denne afhandling tager et nærmere blik på alle dynamiske aspekter af proteiner: proteinstrukturensembler, proteindynamik og evolution af proteinsekvenser. Alle foreslåede metoder er baseret på generative maskinlæringsmodeller, således at der kan produceres nye datapunkter, der kommer fra omtrent den samme fordeling som de data, der ses under træning.

Hovedbidragene i denne afhandling er tredelte. Først foreslår vi en generel metode til at pålægge lokale og globale begrænsninger samtidigt i modellering af proteinstrukturensembler. Som et bevis på princippet er denne metode inkorporeret i en simpel variationel autoenkoder (VAE), og vi demonstrerer, at de genererede strukturer er af høj kvalitet, både lokalt og globalt. Det andet bidrag er en metode baseret på støjreducerende diffusionsmodeller trænet på forenklede repræsentationer af proteinstrukturer fra simuleringer af molekylærdynamik. Denne model kan ikke alene direkte generere nye strukturer, men kan også bruges til billigt at ekstrahere et molekylært kraftfelt, som gør det muligt at udføre nye molekylære simuleringer. Det sidste bidrag er en nærmere undersøgelse af brugen af VAE'er til at modellere sekvensdata fra proteinfamilier. Specifikt undersøger vi styrkerne og svaghederne ved Bayesianske dekodere i denne sammenhæng, samt vise potentialet af hierarkiske VAE'er til at afhjælpe misforholdet mellem den almindeligt anvendte standard normalfordelte prior over det latente rum og den "stjerneformede" aggregerede posterior for protein familiedata.

Table of Contents

Chapter 1

Introduction

Protein sequence, structure and function are inextricably intertwined, and understanding these interconnections is the key to understanding biological processes and disease. Over the last years, a lot of research effort has been invested in solving the long-standing challenge of predicting static protein structures from amino acid sequences. Recently, this has lead to unprecedented successes, set in motion by DeepMind's AlphaFold2 [1] convincingly winning the challenging protein folding competition "Critical Assessment of Techniques for Protein Structure Prediction" in 2020 (CASP14). However, static structures do not represent the full picture of a functional protein. In this thesis, I present different methods based on generative machine learning models to address various aspects of the flexible protein picture, ranging from ensembles of protein structures and dynamics to sequence variance through evolution. The focus of this introduction will be on explaining the concepts needed in Chapter 3, Chapter 4 and Chapter 5, both from a biological and a machine learning perspective.

1.1 Protein Structure & Sequence Variation

Firstly, we will take a closer look at proteins, essentially the dynamic cogs and machines that are built from our DNA blueprint to carry out almost all bodily functions. In the same way a faulty blueprint can lead to a questionable product, DNA mutations can lead to proteins that differ in appearance and behavior. This may result in a useless or even detrimental protein, but could also lead to a molecule that is more stable or better suited for a specific task. In the following sections, I will discuss proteins and their variation in terms of both sequence and structure in more detail.

1.1.1 Protein Structure & Sequence

Proteins are polymer chains folded into a specific three-dimensional shape, of which the surface has distinct physical features as well as chemical properties such as charge and polarity [2, Chapter 2]. For example, proteins can bind *ligands* (small molecules) in *binding pockets*, which are inlets in the protein surface with specific properties optimized to compliment that particular ligand. The 3D protein shape itself is also largely determined by the polarity and charge within the polymer chain, with *hydrophobic* (*i.e.* water-repelling) parts packed away at the inner part of the protein, while *hydrophilic* (*i.e.* water-attracting) parts are exposed on the outside, and electrostatic bonds formed between oppositely charged side chains [2, Chapter 1].

The structure of a single protein, also called a monomer, is organized in three levels [3, Chapter 3], as shown in Fig. 1.1. The first is called the *primary structure*, where a chain of repeating units called *amino acids* is formed [4]. All amino acids share a basic building block consisting of a nitrogen atom and two carbon atoms, where the last carbon forms a so-called peptide bond with the nitrogen atom of the next amino acid to form the polymer chain. The central carbon of each amino acid, termed C_α, is connected to a side chain, also called a residual group. For naturally occurring proteins, there exists twenty different residual groups, each with distinct chemical properties. The repeated N-C_α-C sequence forms the *backbone* of the protein, and the residual groups stick out of this backbone. Each of the twenty different amino acids is associated with a capital letter (Latin alphabet), so that a chain of amino acids can also be represented as a sequence of letters.

The second level of structure arises when the chain of amino acids folds up locally to form *secondary structural elements*, namely *α-helices* and *β-sheets*, where the latter consists of multiple parallel *β-strands* [5]. Hydrogen and sulfur bridges fortify these relatively rigid elements which are connected by flexible loops that exhibit more fluctuations.

Finally, the protein chain folds onto itself to achieve the third level of structural organization: the *tertiary structure* or *global fold*. Ultimately, this three dimensional shape is determined by the location of secondary structure elements and the interplay between the chemical properties of the amino acid chain and the chemical environment [6]. Larger proteins can contain multiple *domains* that fold up locally and independently before the full global fold is completed. It's worth noting that even though most proteins have a preferred folded state, often called the *native state*, they are dynamic molecules that adopt multiple conformations. This will be discussed more elaborately in Section 1.1.3.

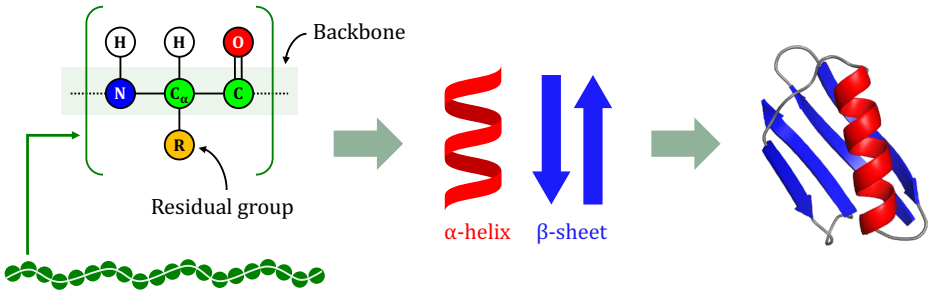


Figure 1.1: Protein monomer structure is organized into three levels: *primary:* the amino acid sequence, *secondary:* the local structure, and *tertiary:* the global fold.

Protein Families

The aforementioned link between protein sequence, structure and function is clearly visible throughout evolution. Similarly to how the evolution of species results in "survival of the fittest", protein evolution is governed by natural selection. Random mutations in DNA can result in substitutions, insertions

or deletions in the amino acid sequence. This gives rise to groups of proteins derived from a common ancestor, also called *homologs*, that are similar in terms of sequence, structure and function [7, 8]. These groups of proteins are known as *protein families*, and members can be found both within and across species. A diverse pool of genes and corresponding proteins leads to more flexibility in protein shape, function, and stability. Beneficial variants will persist, while detrimental variants are either not viable or will not be selected for. Protein evolution is inspirational to fields such as protein design, as it is often easier to adapt an existing protein than to create *de novo* proteins [9]. After choosing a known protein that already possesses some of the desired qualities, single or multiple mutations can be made experimentally to test the change in the properties of the protein. Designing proteins through this experimental route can be time- and resource-consuming. Therefore it is an active field of research to find strategies to guide exploratory experiments, or, ultimately, design new proteins from scratch.

Multiple Sequence Alignments (MSA)

Collections of similar sequences, such as protein families, are often represented as a *multiple sequence alignment* (MSA) [10]. In an MSA, all sequences are scaled to the longest protein in the collection, aligning the amino acids in a way that maximizes the overlap between identical or similar entries and introducing gaps where needed. Not only is this a clear way to organize protein sequences, it can also serve as a valuable input to computational methods. This is especially true regarding protein families, since an MSA can then be viewed as a snapshot of evolution, which provides a lot of information about which mutations were selected for. Examples of such methods are Riesselman et al. [11] and Frazer et al. [12]. While these models are capable learning the evolutionary organization of protein sequences, they have certain limitations that were the inspiration for the work presented in Chapter 5. I will come back to this in Section 1.2.1, since some theoretical background behind variational autoencoders is needed for the reasoning behind our work.

Static Protein Structure

First of all, I would like to provide some historical perspective around experimental protein structure determination. I will not go into much technical detail for the experimental methods mentioned, and refer the reader to the

respective papers for more information. The scientific field that concerns itself with the investigation of protein structure stems from the late 19$^{\text{th}}$ and early 20$^{\text{th}}$ century [13, 14]. These initial experiments established the understanding of proteins as polymer chains of amino acids which in turn led to the notion of higher order structure and the driving force of hydrophobic interaction [5, 15, 16]. Later on, scientists successfully identified different "states" of protein structure by looking at the patterns of diffracted X-rays for a specific material [17]. The issue with this type of analysis was that diffraction patterns are often not very clear for raw materials. A big leap forward was applying X-ray diffraction to crystallized proteins [18]. In this method, called *X-ray crystallography*, proteins are first crystallized to form a regular, fixed structure, for which the diffraction patterns are much cleaner and more consistent. The invention of this technique was awarded with a Nobel Prize in 1915. The successful application of this technique to obtain the first ever protein structure, of muscle myoglobin, was also awarded a Nobel Prize in 1962 [19].

An obvious but important bias of this structure determination method is the need to crystallize the proteins, which is not possible for every protein and will also not be feasible for every possible conformation of the protein. That being said, X-ray crystallography is still, to this day, one of the most used experimental methods to infer protein structure, alongside *nuclear magnetic resonance* (NMR) spectroscopy and *cryogenic electron microscopy* (cryo-EM). In a nutshell, NMR spectroscopy leverages changes in the local electronic environments around atom nuclei to get an ensemble of protein structures in solution [20–22]. The invention of this technique was awarded the Nobel prize in Chemistry in 2002. In contrast, cryo-EM determines protein structure through 3D reconstruction of multiple 2D snapshots of a frozen protein sample [23, 24], which got the Nobel Prize in Chemistry in 2017.

The experimental techniques mentioned above increased the number of known static protein structures. Moreover, sequencing techniques developed such that there was an abundance of proteins for which we know the amino acid sequence. Given the growing number of available structures and sequences, the only thing missing is a mapping from one to the other, *i.e.* how to go from primary to tertiary structure. This connection was first demonstrated in 1953, which resulted in another Nobel Prize in 1972 [25]. Predicting the global structure from the amino acid sequence, also known as the "protein folding problem", has been a long standing challenge in computational biol-

ogy. Following the theory that the native fold of a protein is its most stable state, established methods within the field aim to maximize the stability of the predicted structure through minimization of the free energy [26]. This is commonly done either by taking a known similar structure as a template [27], or by leveraging an MSA of homologous structures through a database search [28], or by assembling small structure fragments with a similar sequence [29, 30].

Over the last years, a lot of progress in sequence-to-structure prediction has been made within the field of machine learning. The first end-to-end differentiable model for protein structure learning was presented by AlQuraishi [31] in 2019. Only a year later, DeepMind's AlphaFold2 [1], a large transformer-based model that also includes evolutionary information, bested its competition at the protein folding competition CASP14, after which some now claim the protein folding problem is "solved". While this last statement is still debatable, AlphaFold2 presented a large step forwards, which was quickly followed up by others [32, 33]. Even more recently, diffusion models (see Section 1.2.2) have started to emerge as promising models for structure prediction and protein design [34, 35].

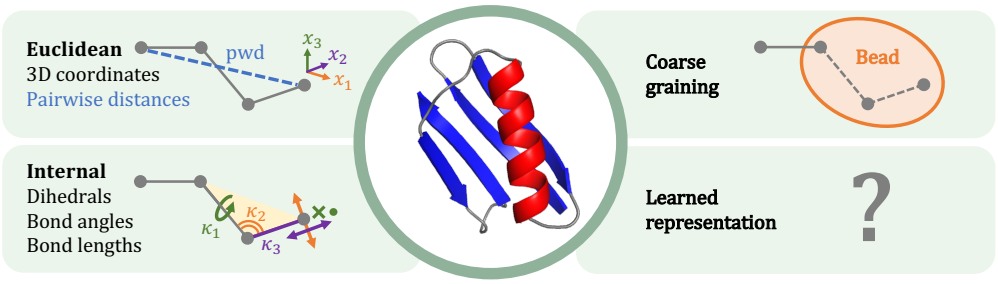


Figure 1.2: An overview of the protein structure representation relevant for this thesis: 3D coordinates (top left) with one pairwise distance (pwd) indicated in blue, internal coordinates (bottom left), coarse-grained representations (top right) and learned representations (bottom right). The protein shown in the middle is protein G (PDB ID: 1pga), which will also be used in subsequent example figures and is one of the test cases in both Chapter 3 and Chapter 4.

1.1.2 Protein Structure Representations

Protein structures can be represented in different ways. Out of many possibilities, I will here discuss the representations relevant for this thesis, of which an overview is depicted in Fig. 1.2. Throughout this section, I will consider only the N-C_α-C backbone of proteins with a length of L amino acids, corresponding to $A = L \times 3$ atoms. Since the focus of this thesis is mostly on the flexible protein picture, I will also highlight how different representations handle perturbations in structure.

Cartesian Coordinates

One of the most intuitive options to represent a protein structure is in *Cartesian coordinates*, *i.e.* in Euclidean space. As this requires an x-, y- and z-coordinate for each atom, this results in $A \times 3$ coordinates. Importantly, working with 3D coordinates corresponds to placing the protein with a predetermined translation and rotation in Euclidean space. Fig. 1.3 shows an example of a 3D protein representation.

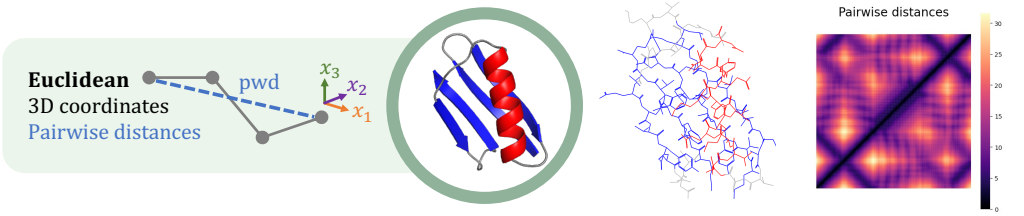


Figure 1.3: An example of a Cartesian coordinate representation for protein G (PDB ID: 1pga) shown as a 3D molecular graph of the full structure (both backbone and side chains) in the middle and pairwise distances for the backbone atoms on the right.

Perturbations It is relatively simple to make a perturbation to a single atom in 3D space, because effectively this will only change the coordinates of the atom in question and leave all other atom coordinates unchanged. Even though perturbations generally do not affect the global structure much, they can violate the local constraints that occur in natural proteins, resulting in *e.g.* bond dissociation and steric clashes (*i.e.* bond crossing).

Pairwise Distances

Pairwise distances are Euclidean distances between all possible pairs of atoms within the protein. Since they is often used in tandem with Cartesian representations, it was chosen to depict them together in Fig. 1.2. Pairwise distances can be organized in an $A \times A$ matrix, exemplified by the matrix in Fig. 1.3 on the right-hand side. This matrix is symmetrical due to the fact that the pairwise distance is the Euclidean norm of the vector from atom i to atom j, which will give the same distance as the norm between atom j and atom i. Moreover, the diagonal will be zero because these are the distances between the atom and itself, and the offset-one diagonals contain bond lengths. While pairwise distances are independent of the rotation and translation of the protein, they are also invariant with respect to reflections. This is not a desirable feature for proteins, since the handedness of structural elements is an important physical and chemical feature. Moreover, pairwise distance presentations are redundant; a subset of all pairwise distances is sufficient to infer the missing values.

Perturbations A pairwise distance representation is hard to work with in terms of perturbations, since one cannot simply change one pairwise distance without affecting others. For this reason, pairwise distances are often used only at evaluation time, while perturbations are done using a different representation such as Cartesian coordinates.

Internal Coordinates

The *internal coordinate* representation for proteins consists of *bond lengths*, *bond angles* and *dihedrals*. Bond lengths are simply the distance between the current atom and its successor, corresponding to the $A - 1$ atomic bonds in the backbone chain. Bond angles are angles between two consecutive bonds, of which there are $A - 2$ in the protein backbone. Finally, dihedrals are torsional angles, which require four consecutive atoms in order to be computed. For this set of four atoms, the dihedral is the angle between the plane defined by the first three atoms and the plane defined by the last three atoms. A dihedral is essentially the torsional angle that twists the protein backbone around the bond between the middle two atoms of the set, and the backbone contains $A-3$ such dihedrals. Fig. 1.4 shows an example of how internal coordinates are distributed for an ensemble of protein structures. Bond lengths

and bond angles follow quite peaked distributions, while dihedrals display the most variance, making them the most important factor in determining the structure of the protein backbone. It is common to represent the most fluctuating dihedrals, namely those around N-C_α bonds (ϕ angles) and C_α-C bonds (ψ angles) together in a *Ramachandran plot*. This ψ-versus-ϕ plot exhibits clusters corresponding to specific secondary structural elements, as indicated in Fig. 1.4. This basic internal representation contains $3 \times A - 6$ coordinates and is redundant, although it is also possible to reduce such representations to a fully non-redundant version [36]. For protein modelling purposes, bond lengths can often be assumed to be fixed due to their small variance, which makes the representation cheaper ($2 \times A - 5$ coordinates). In any form, internal coordinate representations are invariant with respect to rotations and translations.

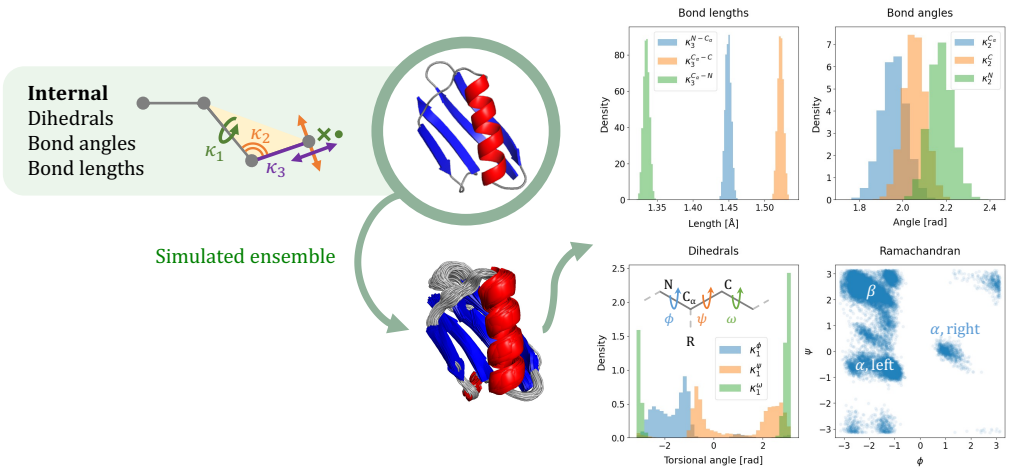


Figure 1.4: An example of distributions over the internal coordinate representation for a short simulation (400 frames, 50ps interval) of protein G (PDB ID: 1pga).

Perturbations The position of each (non-terminal) atom i in the backbone is determined by three corresponding internal coordinates, as shown in Fig. 1.4 on the left hand side. The bond length preceding the atom can move the atom along the bond. For small changes, the bond angle around atom $i-1$ moves the atom approximately perpendicular to the bond and within the bond angle plane. The dihedral around the bond between atom

$i-2$ and $i-1$ moves the atom perpendicular to the previous directions for small perturbations, *i.e.* in and out of the bond angle plane. A considerable advantage of perturbing in internal coordinate space is that local structure is always preserved. This is due to the fact that a change in one internal coordinate will lead to a rigid body movement, where the perturbed atom will move together with all atoms downstream. Even though local physical constraints will not suffer much from perturbations, it is very challenging to uphold global constraints, since this requires a complicated covariance structure over all internal coordinates. In Chapter 3, we present a method that unifies local and global constraints within the internal coordinate parameterization. This is achieved through the incorporation of global constraints in Euclidean space into a full covariance structure over internal coordinates for ensembles of protein structures.

Coarse-Grained Representations

For the representations discussed so far, we have mostly considered the backbone of proteins. This is a simplified view on the full, *fine-grained* protein that also includes all residual groups. The full molecule can be very expensive to use in *e.g.* simulations, since it would require computing the forces acting on each atom as I will come back to in the next section "Protein Dynamics". Therefore, it may be useful to work with simplified representations of the protein to save computational cost, while preserving the relevant features as well as possible despite the loss of fine-grained information. A *coarse-grained* (CG) representation is obtained by grouping certain atoms together into so-called *beads* using a *coarse-graining mapping* [37]. These beads are usually placed at the center of mass of the corresponding atom group or at the location of the heaviest atom in the group. The backbone is an extremely simple example of a CG representation where the mapping is just a slicing operation for all the backbone atoms. Similarly, slicing out the $C_\alpha$atoms gives another simple CG representation, as illustrated in Fig. 1.5. CG mappings can be chosen to be much more sophisticated or even learned [38]. Coarse-grained representations will play a large role in Chapter 4, where we propose a method to generate coarse-grained structures as well as obtain a coarse-grained force field needed for simulations. Both the method and simulation basics will be covered in the sections below: diffusion models are described in Section 1.2.2, and simulations and force fields are discussed in Section 1.1.3.

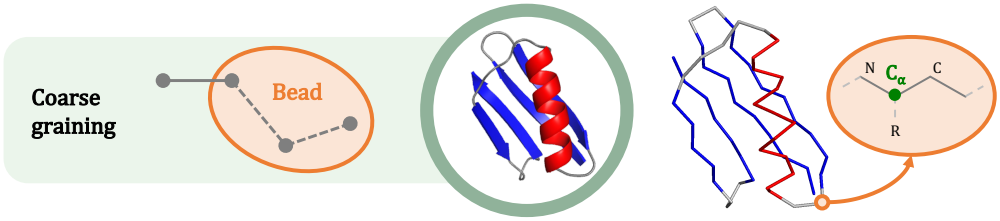


Figure 1.5: An example of a C_α-coarse-grained representation of protein G (PDB ID: 1pga), where amino acids are reduced to beads located at the $C_\alpha$atom position.

Perturbations Coarse-grained representations are not bound to a specific coordinate system. Just like atoms, beads can for example be chosen to be placed in Euclidean space or internal coordinate space, with all corresponding advantages and drawbacks for perturbations. Depending on the coarse-graining mapping, it may be more challenging to satisfy local constraints due to the loss of fine-grained information.

Learned Representations

All methods discussed so far constitute different ways to parameterize a protein, which would still ultimately be in three-dimensional space. In contrast, *learned representations* can be set in any space with an arbitrary number of dimensions. There is a wide variety of models within the field of machine learning that uses such learned representations one way or another, of which generative probabilistic latent variable models are relevant for this thesis and will be elaborately discussed in Section 1.2.

1.1.3 Protein Dynamics

Up until this point, I have discussed proteins in a static setting, and described how perturbations affect the structure. As a final step towards the dynamic protein picture, I will transition to protein dynamics and simulation. Proteins are inherently *dynamic* molecules, which is essential to carry out their function. Changes in structure can occur on short time scales for processes such as domain and hydrogen bond vibrations, or on larger time scales for conformational transitions such as folding and unfolding events, pore opening and closing, and hinge movements [39]. Here, I will touch upon a selection

of protein dynamics tools relevant for the work in this thesis.

Molecular Dynamics

In a molecular dynamics (MD) simulation, the protein is placed in an environment, *e.g.* a block of water molecules as shown in Fig. 1.6. Given this composite biomolecular system, the forces exerted on each atom can be calculated and Newton's laws of motion will dictate how the atoms move in 3D space as a function of time. The *approximated force field* used in this type of simulation is the result of quantum mechanical computations, commonly supplemented with experimental measurements [40]. There is a myriad of options for force field calculation methods, of which AMBER [41], CHARMM [42] and OPLS [43] are typical choices.

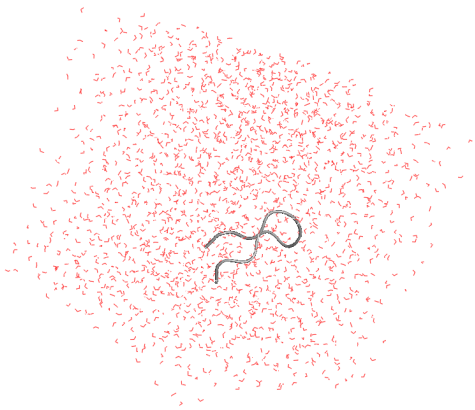


Figure 1.6: Chignolin (grey), a peptide (*i.e.* a small protein) of 10 amino acids, in a block of solvent molecules (red). Simulation frame from Lindorff-Larsen et al. [44].

Langevin Dynamics

In the classic setting, a force field F is calculated as the negative gradient of the interaction potential V over particles $\boldsymbol{x}$: $F(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} V(\boldsymbol{x})$. Correspondingly, the equation of motion, based on Newton's second law, becomes $F = M\frac{\mathrm{d}^2\boldsymbol{x}}{\mathrm{d}t^2}$, with M the particle masses. This is an approximation of reality which may not take into account frictions caused by the solvent molecules or random collision events. Additionally, classic deterministic force fields do not require the temperature to be fixed, even though in most cases it is desired to

analyze the protein's behavior at a specific temperature. *Langevin dynamics* [45] is an example of a *thermostatting* method designed to keep the system at a given temperature [46]. Friction and stochastic forces are incorporated into the Langevin equation of motion as follows:

$$M\frac{\mathrm{d}^2\boldsymbol{x}}{\mathrm{d}t^2} = \underbrace{F(\boldsymbol{x})}_{\substack{\text{particle}\\\text{interactions}}} - \underbrace{\gamma M\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}}_{\text{friction}} + \underbrace{\sqrt{2M\gamma k_B T}\mathbf{w}(t)}_{\text{random events}}, \tag{1.1}$$

where t is time, γ is the friction coefficient, k_B is the Boltzmann constant, T is the temperature, and $\mathbf{w}(t)$ is a random force generated by a delta-correlated stationary Gaussian process [46]. This type of dynamics is relevant for Chapter 4, where we perform Langevin dynamics simulations using a predicted force field.

Coarse-Graining

One can imagine how force calculations on each atom in a system can quickly increase computational cost. To alleviate this problem, a reduced, coarse-grained (CG) representation of the biomolecular system may be used (see "Coarse-grained representations" in Section 1.1.2). Even though this fixes one problem, it creates another: a CG representation requires a corresponding *coarse-grained force field*, without losing thermodynamic consistency. Methods to obtain such CG force fields include variational force matching, relative entropy minimization, and flow-matching. Variational force matching explicitly minimizes the difference between coarse-grained and fine-grained forces [47], while relative entropy minimization performs force-agnostic density estimation [48]. Flow-matching is a combination of both, utilizing a normalizing flow as a density-estimator teacher model, and a separate, more flexible student model that does force-matching with respect to its teacher [49]. The reader is referred to the respective papers for more details. In Chapter 4, we propose a model with a simple setup that outperforms flow-matching in both sampling and dynamics for coarse-grained proteins, while also being capable of scaling to larger systems. The underlying model is based on denoising diffusion, which will be described elaborately in Section 1.2.2.

1.2 Generative Latent Variable Models

All works presented in this thesis feature generative probabilistic latent variable models. These are data-driven methods that model the underlying data distribution, assuming the inputs are observations of the random variable of interest. For the work considered in this thesis, the inputs will be some type of protein representation, for example a multiple sequence alignment or the Cartesian coordinates of the backbone of a protein structure. The observed random variable follows an unknown distribution, which we hope to better approximate by introducing latent variables, which are unobserved. Even though there are no observations for these variables, the intuitive idea is that including them in the model can help explain the behavior of the observed variable. As an example, consider a protein moving through a block of solvent molecules. The observed variable is the location of the protein over time, and the latent variable represents the force field acting on the protein. Of course the force field will greatly influence the movements of the protein, and therefore including it in the model will help predicting where the protein will be next. In this example, the latent variable is an actual interpretable and physical concept, but this does not necessarily need to be the case. The goal of these probabilistic latent variable models is to approximate the true data distribution by maximizing the likelihood that the modeled distribution is the same as the process that generated the observed data. Importantly, the parameterization of the modelled distribution is chosen in such a way that it can be sampled from, hence the term *generative* probabilistic latent variable model.

That being said, there exists a wide variety of these types of models. Here, I will zoom in on variational autoencoders (VAE) and denoising diffusion probabilistic models (DDPM). A simple, single-layer VAE is used in Chapter 3 to demonstrate the potential of the proposed way to incorporate global constraints into the covariance structure over internal coordinates. Chapter 5 investigates the properties of different variants of the VAE in the context of protein sequence (MSA) modelling. More specifically, we use ladder VAEs (LVAEs), which is a type of hierarchical VAE (HVAE), as well as importance weighted samples (IWAE) and a Bayesian decoder, all described below. In Chapter 4, we use a DDPM model to obtain an equilibrium distribution as well as a force field for coarse-grained protein structures from molecular dynamics data.

Since an exhaustive explanation of machine learning and neural networks is beyond the scope of this thesis, I will assume basic familiarity with concepts such as neurons, weights and backpropagation.

1.2.1 Variational Autoencoders

Variational autoencoders (VAEs) are a specific type of autoencoder first introduced by Kingma and Welling [50] and Rezende et al. [51]. Generally, autoencoders are used to reconstruct the input after passing it through a bottleneck, which can be regarded as the learned latent dimensionality-reduced representation of the input [52]. Like regular autoencoders, VAEs consist of an encoder, which maps to the latent space, and a decoder, which maps the latent representation back to the reconstruction of the input. The main difference between a VAE and a regular autoencoder is that the VAE learns a parameterization of a *distribution* over a latent space rather than learning the mapping to the latent space directly. This makes it possible to draw samples from a VAE, while regular autoencoders are not generative. Fig. 1.7 shows a simplified depiction of how a VAE could be used to reconstruct a data point. VAEs are versatile models which have, amongst others, proven useful within the field of protein modelling for applications ranging from representation learning to protein design [11, 12, 53–55].

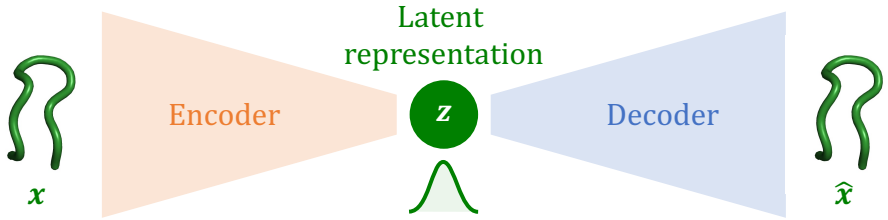


Figure 1.7: Simple illustration of a VAE, where an encoder maps the data point $\boldsymbol{x}$ to a distribution in dimensionality-reduced latent space $\boldsymbol{z}$, and the decoder maps back from $\boldsymbol{z}$ to the reconstructed sample $\hat{\boldsymbol{x}}$. The molecule shown in this example is the central carbon (C_α) trace of chignolin, a peptide (*i.e.* a small protein) of 10 amino acids that will be one of the test cases in Chapter 4.

Let us take a step back and consider why we would need variational autoencoders, and introduce the general notation. Fig. 1.8A depicts the general model, where the latent variable $\boldsymbol{z}$ generates our random variable of interest $\boldsymbol{x}$ with *likelihood* $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, where θ are the distribution parameters learned by a neural network. By updating these model parameters $\boldsymbol{\theta}$, the likelihood can be maximized such that for optimal parameters the data distribution is modelled as closely as possible. Moreover, we choose a *prior* distribution $p(\boldsymbol{z})$ for the latent variable, which does not depend on observed states of $\boldsymbol{x}$. This prior is commonly chosen to be standard Gaussian ($\mathcal{N}(\mathbf{0}, \mathbf{I})$). The distribution over $\boldsymbol{x}$ can be described by the *marginal likelihood*, also called the *model evidence*:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z}. \tag{1.2}$$

The problem with Eq. (1.2) is that it often does not have an analytic solution or a sufficiently efficient estimator, especially for high-dimensional spaces [56]. Instead, let us consider the *posterior* distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ which arises when Bayes' rule is used to update the prior with new observations characterized by the likelihood [57]:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}. \tag{1.3}$$

This *true posterior* would correspond to reversing the blue arrow in Fig. 1.8A. While the nominator terms $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and $p(\boldsymbol{z})$ are tractable to compute, the remaining problem is still the normalizing constant $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, which was previously stated to be intractable. Therefore, we can choose to *approximate* the posterior instead by a new distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ with model parameters $\boldsymbol{\phi}$, represented by the dotted orange line in Fig. 1.8A [50, 51]. The objective is to find an approximate posterior such that $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \approx p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, and therefore we need an appropriate way to compare the approximate and true posterior.

KL-divergence The *Kullback-Leibler divergence* (KL divergence) is a measure of the difference between two distributions [58]. The idea behind the KL divergence is very simple: if we want to compute the dissimilarity between a

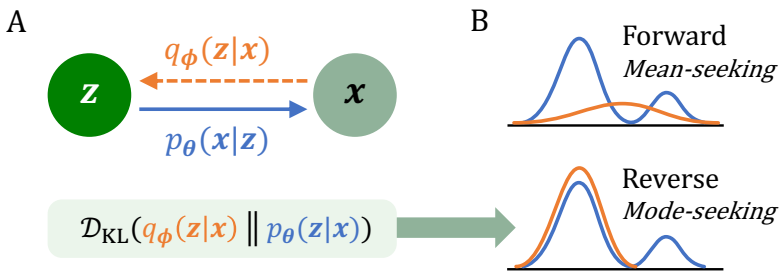


Figure 1.8: Variational autoencoder setup. A: graphical model of the VAE, with the likelihood indicated as a blue arrow and the approximated posterior indicated as a dotted orange arrow. The objective is to match true and approximated posterior. B: Reverse KL-divergence is chosen for model optimization, since a mode-seeking behavior is preferred over a mean-seeking behavior (example shows bimodal p in blue and unimodal approximation q in orange).

reference distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and an approximated distribution $q_{\boldsymbol{\phi}}(\boldsymbol{x})$ (leaving out conditioning on $\boldsymbol{z}$ for now), we can simply take the ratio of these distributions: $\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x_i})}{q_{\boldsymbol{\phi}}(\boldsymbol{x_i})}$ for one input $\boldsymbol{x}_i$. Note that this measure is asymmetrical, *i.e.* it matters if we divide p by q or vice versa, which will be important later on. For numeric stability and convenience we take the difference of the *log*-likelihoods instead: $\log p_{\boldsymbol{\theta}}(\boldsymbol{x_i}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{x_i}) = \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x_i})}{q_{\boldsymbol{\phi}}(\boldsymbol{x_i})}$, which is called the *log-likelihood ratio* (LLR). Of course, rather than using just one sample, we would like to compute the expected value of this ratio, which is the definition of the KL divergence. Since the LLR is a function of a random variable $\boldsymbol{x}$, we can use the law of the unconscious statistician to formulate the expectation:

$$\mathcal{D}_{\mathrm{KL}}(p_{\boldsymbol{\theta}} \,||\, q_{\boldsymbol{\phi}}) := \mathbb{E}_{p_{\boldsymbol{\theta}}}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{x})}\right] = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{x})} d\boldsymbol{x}. \tag{1.4}$$

It was already mentioned that the KL divergence is not symmetric. Eq. (1.4) shows the *forward* KL divergence between reference distribution p and approximation q. However, we could also flip the two distribution, giving rise

to the *reverse* KL divergence:

$$\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}} \,||\, p_{\boldsymbol{\theta}}) := \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\right] = \int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} d\boldsymbol{x}. \tag{1.5}$$

To determine whether to use forward or reverse KL divergence, we can compare their behaviors. Fig. 1.8B shows a simple example of a bimodal reference distribution p in blue and a unimodal approximation q in orange, representing the common situation where we want to model a complex distribution by a simpler one. When minimizing forward KL divergence according to Eq. (1.4), the optimal distribution of q will display *mean-seeking* behavior, which is due to the fact that a high price is paid whenever the approximation has no probability mass in a region covered by p. In contrast, minimizing the reverse KL divergence according to Eq. (1.5) leads to *mode-seeking* behavior, since the highest gain will come from a large overlap between the largest peak of p and the unimodal q.

Now that we know how forward and reverse KL divergence differ in behavior, we need to determine which one is most suitable for the variational autoencoder, where we want to measure how close our approximated posterior is to the true posterior. Since the aim is to capture a simplified version of the true data distribution while still capturing the most important features, mode-seeking behavior is preferred over mean-seeking behavior. Therefore, the reverse KL divergence is chosen to compare the approximated and true posterior:

$$\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) = \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]. \tag{1.6}$$

As indicated by a red box in Eq. (1.6), there is still the problem of the intractable true posterior in the denominator. It will therefore be necessary to rewrite the objective such that we can compute it.

Evidence lower bound Let us take Eq. (1.6) as a starting point:

$$\begin{aligned}
\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) &= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log \underbrace{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}_{\text{Bayes' rule}}] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})] \\
&\quad + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})]}_{\text{expectation}} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})] \\
&\quad + \underbrace{\log p_{\boldsymbol{\theta}}(\boldsymbol{x})}_{\text{independent of } \boldsymbol{z}} \underbrace{\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z}}_{\text{probability density, } = 1} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})] + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}).
\end{aligned}$$

Reorganizing the terms gives

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})] + \mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})). \tag{1.7}$$

There are still two problems here. Firstly, we know that $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is intractable to compute, and the same holds true for $\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$. However, from the definition of KL divergence (see Eq. (1.5)), we know that it is always positive. Therefore, a lower bound on the log evidence can be given:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] =: \mathcal{L}_{\mathrm{ELBO}}. \tag{1.8}$$

Note that even though we got rid of the KL divergence term, maximizing this lower bound will still implicitly minimize $\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$ for Eq. (1.7) to hold true [59]. Additionally, I kept the denominator $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to be conditioned on $\boldsymbol{x}$, but in principle one could also replace this by $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ since removing the conditioning will not change anything in the derivations above apart from the denominator. Eq. (1.8) is called the evidence lower bound (ELBO), where, in the context of a VAE, $\mathcal{L}_{\mathrm{ELBO}}$ is to be maximized

[50]. Correspondingly, the loss function $\lambda_{\text{VAE}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x})$ to be minimized by the VAE will be the negative of $\mathcal{L}_{\text{ELBO}}$. Now, all that is left to be done is obtain $p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, which will be made more intuitive below.

$$
\begin{aligned}
\lambda_{\text{VAE}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x}) &:= -\mathcal{L}_{\text{ELBO}} \\
&= -\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p(\boldsymbol{z})\right] \\
&= -\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right] \\
&= -\underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]}_{\substack{\text{expected} \\ \text{reconstruction error}}} + \mathcal{D}_{\text{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \;||\; p(\boldsymbol{z})). \qquad (1.9)
\end{aligned}
$$

Now we are left with a learning objective where all terms are obtainable from the VAE. As stated before, VAEs have an encoder and a decoder part. The encoder takes the input and maps to a distribution in latent space parameterized by a mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ since we have chosen a Gaussian prior and posterior. The decoder maps back from latent space to the reconstructed output. The expected reconstruction error term in Eq. (1.9) comes from the decoder, and we can also compute the KL divergence between the known prior and the approximated posterior that is learned by the encoder of the network. The full encoder-decoder network can be trained with Eq. (1.9) as its training loss. The architectures of both encoder and decoder can be simple linear layers, but can also chosen to be more complex. Once the network is trained, new samples are generated by sampling from the prior and passing these samples through the decoder network with trained weights to obtain the final samples.

Reparameterization trick Update the model weights through backpropagation requires taking the gradient of the loss function Eq. (1.9) with respect to all model parameters. For the decoder this is not a problem, since we can swap gradient (w.r.t. $\boldsymbol{\theta}$) and expectation (w.r.t. $q_{\boldsymbol{\phi}}$):

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \lambda_{\text{VAE}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x}) &= \nabla_{\boldsymbol{\theta}} \left(-\mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \right] + \mathcal{D}_{\text{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})) \right) \\
&= \nabla_{\boldsymbol{\theta}} \left(-\mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \right] + \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z}) \right] \right) \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}} [\nabla_{\boldsymbol{\theta}} (- \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \underbrace{\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}_{\text{independent of } \boldsymbol{\theta}} - \log p(\boldsymbol{z}))] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}} [\nabla_{\boldsymbol{\theta}} (- \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z}))] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}} [-\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})].
\end{aligned}
$$

The resulting expectation can be estimated easily with a Monte Carlo estimator, where $\boldsymbol{z}$ is drawn from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, *i.e.* the encoder of the network.

However, getting the gradient $\nabla_{\boldsymbol{\phi}} \lambda_{\text{VAE}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x})$ is not that simple, since gradient and expectation both involve $\boldsymbol{\phi}$:

$$
\begin{aligned}
\nabla_{\boldsymbol{\phi}} \lambda_{\text{VAE}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x}) &= \nabla_{\boldsymbol{\phi}} \left(-\mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \right] + \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z}) \right] \right) \\
&\neq \mathbb{E}_{q_{\boldsymbol{\phi}}} [\nabla_{\boldsymbol{\phi}} (- \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z}))].
\end{aligned}
$$

Therefore, backpropagation w.r.t. $\boldsymbol{\phi}$ is not able to flow through stochastic node $\boldsymbol{z}$, as depicted in Fig. 1.9A.

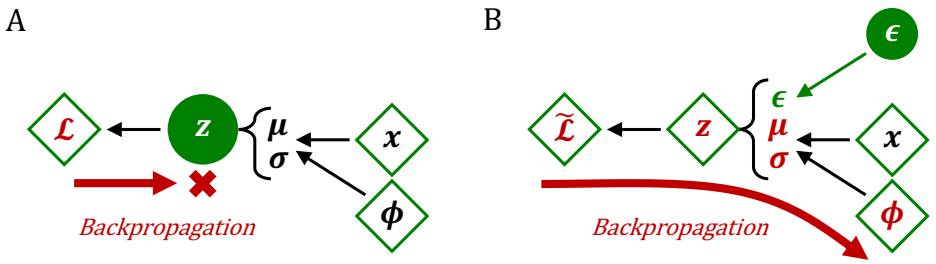


Figure 1.9: The reparameterization trick. Solid green circles are stochastic nodes, diamonds represent deterministic nodes. A: the stochastic node breaks backpropagation. B: the problem is circumvented by the reparameterization trick, where noise sample $\boldsymbol{\epsilon}$ is drawn from a distribution that is independent of $\boldsymbol{\phi}$.

To circumvent this issue, we can apply the reparameterization trick [50, 51, 60], where instead of drawing $\boldsymbol{z}$ from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ we can express $\boldsymbol{z}$ as a transformation T of a newly introduced random variable $\boldsymbol{\epsilon}$: $\boldsymbol{z} = T(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \boldsymbol{x})$. In the

most common case where we have a Gaussian posterior and $\boldsymbol{z} \sim p(\boldsymbol{z}|\boldsymbol{x})$, we can do the following reparameterization:

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}, \tag{1.10}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are predicted by the encoder and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Using the reparameterization trick, the expectations in the loss (Eq. (1.9)) can be taken w.r.t. the distribution over $\boldsymbol{\epsilon}$ instead of $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, which means gradient and expectation can once again be safely swapped in the same way as for the decoder. Fig. 1.9B shows how the backpropagation flow is restored by the reparameterization trick.

Importance Weighted Autoencoders

During VAE training, we commonly draw one sample of $\boldsymbol{z}$ from the approximated posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. However, we can also give a multi-sample version of $\mathcal{L}_{\text{ELBO}}$ from Eq. (1.8) and call it $\mathcal{L}_{\text{ELBO}_{\text{ms}}}$:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\frac{1}{k}\sum_{i=1}^{k}\log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_i, \boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{x})}\right] =: \mathcal{L}_{\text{ELBO}_{\text{ms}}}, \tag{1.11}$$

where $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and number of samples k. Note that $k > 1$ gives $\mathcal{L}_{\text{ELBO}_{\text{ms}}} \geq \mathcal{L}_{\text{ELBO}}$, indicating that drawing more samples will give a better estimation of the ELBO and therefore a closer estimation of the log evidence.

However, Burda et al. [61] propose a way to get even closer to the log evidence, suggesting a tighter bound on the log evidence. In order to understand their reasoning, we need to revisit the ELBO for a bit.

When deriving the ELBO above, we set the KL divergence between the true and approximated posterior as an intuitive starting point to derive the ELBO. However, one could also take the definition of the log evidence as a starting point and derive the ELBO from there, using *importance sampling* (see text box) as a Monte Carlo integrator and swapping log and expectation using Jensen's inequality:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \log \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} \\ &= \log \int \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})d\boldsymbol{z} \\ &= \log \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \quad (1.12) \\ &\geq \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right],\end{aligned}$$

which is the same result as Eq. (1.8).

Importance sampling [62] is essentially a trick where a proxy distribution is used to insert a "probabilistic one", in this case $\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}$. The resulting unnormalized *importance weights* are

$$\boldsymbol{w}_i = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}_i)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{x})}.$$

The likelihood ratio inside the expectation in Eq. (1.12) can be seen as unnormalized importance weights $\boldsymbol{w}_i = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}_i)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{x})}$, whose average is an unbiased estimator of the evidence [61] and therefore

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\frac{1}{k}\sum_{i=1}^{k}\boldsymbol{w}_i\right] \geq \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\frac{1}{k}\sum_{i=1}^{k}\log \boldsymbol{w}_i\right]}_{\mathcal{L}_{\text{ELBO}_{\text{ms}}}}. \quad (1.13)$$

Burda et al. [61] leverage the connection between Eq. (1.12) and importance weights to obtain a new family of lower bounds under the name importance weighted lower bound (IWLB) with corresponding objective $\mathcal{L}_{\text{IWLB}}$ [61, 63]. Compared to $\mathcal{L}_{\text{ELBO}_{\text{ms}}}$, the only difference is that the log and sum are swapped:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{w}_i\right] =: \mathcal{L}_{\text{IWLB}}. \quad (1.14)$$

From here, it is clear from Jensen's inequality that the IWLB provides a tighter bound on the log evidence compared to ELBO:

$$\underbrace{\log \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\frac{1}{k}\sum_{i=1}^{k}\boldsymbol{w}_i\right]}_{\log p_{\boldsymbol{\theta}}(\boldsymbol{x})} \geq \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{w}_i\right]}_{\mathcal{L}_{\text{IWLB}}} \geq \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\frac{1}{k}\sum_{i=1}^{k}\log \boldsymbol{w}_i\right]}_{\mathcal{L}_{\text{ELBO}_{\text{ms}}}}. \quad (1.15)$$

Bear in mind that if the number of samples $k = 1$, the $\mathcal{L}_{\text{IWLB}} = \mathcal{L}_{\text{ELBO}}$ which is the objective for a regular VAE. If we, on the other hand, take the limit of k to infinity, we get $\lim_{k\to\infty} \mathcal{L}_{\text{IWLB}} = \log p_{\boldsymbol{\theta}}(\boldsymbol{x})$. This indicates that for large numbers of k, the true posterior is approached by the approximated posterior. $\mathcal{L}_{\text{IWLB}}$ is the objective for the importance weighted autoencoder (IWAE)[61]. Even though IWAEs have the same type of architecture as VAEs, IWAEs are capable of using the network capacity better and can in principle model more complex distributions [61, 64]. This also makes sense intuitively: a VAE encoder draws one sample for $\boldsymbol{z}$, which will be treated like observed data by the decoder whose weights will be adjusted in a biased manner [63]. In this way, both the encoder and the decoder can be heavily penalized for a "bad" sample. In contrast, the IWAE has more flexibility due to the importance weights, which will put less emphasis on a single sample.

There are also problems associated with optimizing lower bounds and even though it is beyond the scope of this thesis to provide detailed proof, I will briefly mention these limitations and the associated literature. A general problem that occurs in VAEs is that the variance of the gradients of the objective function w.r.t. model parameters can become large, which decreases the signal-to-noise ratio (SNR) and can even prevent the model form converging properly [65, 66]. It has been shown that this problem gets worse for multi-sample estimators [67, 68], and different ways have been proposed to mitigate these problems, both for regular VAEs and IWAEs [65, 66, 69, 70].

Hierarchical VAEs

Up until this point, we have considered a VAE (or IWAE) with a single multidimensional latent variable $\boldsymbol{z}$. Since some prior is assumed over this one latent variable, customarily a standard Gaussian, it is easy to imagine that this can be a large constraint to the model, limiting the complexity of distributions that can be modelled by the VAE. One way to increase the flexibility of the model is to introduce a hierarchy of multiple latent variables, creating a hierarchical variational autoencoder (HVAE) with a more expressive prior and approximate posterior [50, 51, 61, 71–73]. Fig. 1.10 depicts a schematic representation of the encoder (top) and decoder (bottom) of a vanilla HVAE.

In the encoder, corresponding to the inference model as before, the first latent variable $\boldsymbol{z}_1$ is conditioned on the input $\boldsymbol{x}$ and each following latent

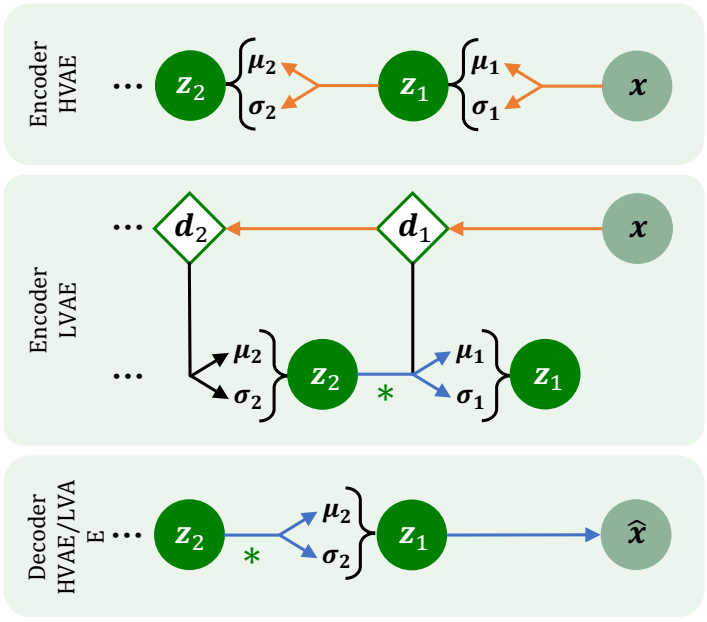


Figure 1.10: Hierarchical VAE (HVAE) structures. Solid green circles are stochastic nodes, diamonds represent deterministic nodes. "Upward" passes are shown as orange arrows, "downward" passes are shown as blue arrows, black lines and arrows indicate transfer without the involvement of a neural network. Top: encoder for a vanilla HVAE. Middle: encoder for a ladder VAE (LVAE). Bottom: the decoder is the same for HVAE and LVAE, apart from the shared weights between the decoder and the downward pass of the LVAE encoder, indicated by a green star.

variable $\boldsymbol{z}_i$ is conditioned on the latent variable preceding it. Therefore the approximate posterior factorizes in a *bottom-up* fashion for a model with N latent variables [61]:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{i=2}^{N} q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{z}_{i-1}). \tag{1.16}$$

In contrast, the decoder represents the generative model and its joint probability distribution factorizes in a *top-down* manner:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_1) \prod_{i=1}^{N-1} p_{\boldsymbol{\theta}}(\boldsymbol{z}_i, \boldsymbol{z}_{i+1}), \tag{1.17}$$

with $\prod_{i=1}^{N-1} p_{\boldsymbol{\theta}}(\boldsymbol{z}_i, \boldsymbol{z}_{i+1}) = p(\boldsymbol{z})$ the prior distribution. Even though this hierarchical structure gives more flexibility to the model, HVAE training can be relatively unstable due to the cumulative effect of conditional stochasticity as more latent variables are added [73]. Moreover, HVAEs with many layers tend to have inactive layers higher up in the hierarchy, *i.e.* closer to $\boldsymbol{z}_N$, which do not have any effect on the learned representation and therefore do not lead to a gain in performance [61, 74, 75].

A solution to this problem is suggested by Sønderby et al. [71], who introduce a method called the ladder variational autoencoder (LVAE). Fig. 2.3 (middle) shows the inference model of the LVAE. As opposed to the encoder of a vanilla HVAE, the LVAE encoder first does a *deterministic* upward pass, followed by *top-down inference*:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_L|\boldsymbol{x}) \prod_{i=N-1}^{1} q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{z}_{i+1}). \tag{1.18}$$

Here, the mean and standard deviation of the approximate posterior at each latent layer i is a combination of the mean and variance from the deterministic node $\boldsymbol{d}_i$ from the upward pass and the stochastic latent layer upstream $\boldsymbol{z}_{i+1}$ [71, 74] as depicted in Fig. 2.3 (middle, see Sønderby et al. [71] for implementation details).

Bayesian Decoder

In a regular VAE model, we assume a single estimate for each parameter in the decoder network, *i.e.* we get a point estimate $\boldsymbol{\theta}^*$ that optimizes the likelihood $p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}^*)$. However, this approach does not incorporate the uncertainty over the selection of the network parameters (weights and biases), also known as the *epistemic uncertainty*. This can lead to overly confident generative models, especially for samples that do not lie within the data distribution [76–78]. Instead of only being probabilistic about $\boldsymbol{z}$, we could also infer a distribution over the parameters of the decoder network and treat

them in a Bayesian way [50]. Similar to before, we seek to approximate the true posterior $p(\boldsymbol{\theta}|\boldsymbol{X})$, with $\boldsymbol{X}$ the full training dataset. In practice, this is often done through minimization of the KL divergence between proxy distribution $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ and prior $p(\boldsymbol{\theta})$, also known as "Bayes by Backprop" [79]. Here, the prior is commonly chosen to be a standard Gaussian such that $p(\boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Moreover, $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ is also usually assumed to be diagonally Gaussian distributed, parameterized by $\boldsymbol{\lambda} = \{\boldsymbol{\mu}, \boldsymbol{\sigma^2}\}$. Bayes by Backprop gives us a new lower bound and corresponding objective for a dataset with N samples (see [79] and appendix F of [50]):

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq\ & N \cdot \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\lambda})q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{\theta}) - \mathcal{D}_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z})\right)]\right] \\ & - \mathcal{D}_{\mathrm{KL}}\left(q(\boldsymbol{\theta}|\boldsymbol{\lambda}) \mid\mid p(\boldsymbol{\theta})\right). \end{aligned} \tag{1.19}$$

In the generative process, the weights and biases are drawn from the predicted distributions, thereby making the model more robust towards epistemic uncertainty. As a side note, it is also possible to be Bayesian over the parameters of the encoder network, but since estimating distributions and sampling from them is more expensive compared to point estimates, the common choice is to only be probabilistic for the generative network.

While being Bayesian about the decoder weights takes epistemic uncertainty into account, having a stochastic estimate for each parameter can also add noise, especially when a standard diagonal Gaussian distribution is assumed such that all parameter distributions are sampled independently. Although there are ways to get a full covariance structure for the parameters, such as Laplace approximation [80], this is usually a very expensive operation. Another way to mitigate noise is by scaling down the variance of the network parameter distributions and making them more "peaked". The scaling factor is often referred to as the *temperature*, where a higher temperature corresponds to a higher variance. Lowering the temperature makes the variance smaller and therefore it becomes more likely that the parameters are sampled close to the mean, which may in many cases increase performance in a phenomenon is called the *cold posterior effect* [81].

Variational Autoencoders: Conclusion

VAEs are probabilistic generative models consisting of an encoder, a latent space with reduced dimensionality, and a decoder. They are trained by optimizing a lower bound on the model evidence. Several flavors of VAEs exist; trying to tighten the lower bound (IWAE), incorporate epistemic uncertainty (Bayesian decoder), increase flexibility by adding more layers of latent space (HVAE/LVAE), and many more. As we move on to Section 1.2.2, consider what would happen if we expand an hierarchical VAE to have many, say a thousand, latent space layers, and replace the encoder by a stochastic process without any learned parameters as well as disregard the need for a bottle neck. The result would, in a very hand-wavy manner, already be quite close to a *diffusion model*, and keeping this in mind will hopefully facilitate reading the next section.

1.2.2 Diffusion Models

Diffusion models are based on the idea that a model can learn how to remove stochastically added noise. In the so-called *forward process*, increasing amounts of noise are added to a data point until a state of pure noise is reached. The generative model learns how to remove noise in the *reverse process*. This way, a corrupted sample can be reconstructed or a completely new sample can be generated by drawing a sample from noise and subsequently *denoising* the sample. Fig. 1.11 shows a simplified picture of a diffusion model, reducing a small protein to noise and denoising it in the opposite direction.

One of the first diffusion models within the domain of machine learning was introduced by Sohl-Dickstein et al. [82], but they started attracting more attention with the publications of Song and Ermon [83] and Ho et al. [84]. Diffusion models have gained momentum in recent years, becoming state-of-the-art in image synthesis [85–87], resulting in popular tools such as StabilityAI's Stable Diffusion [88] and Midjourney [89], and OpenAI's DALL·E 2 [90]. More recently, diffusion models have emerged as a promising tool for modelling proteins [34, 35, 91–93].

Throughout the diffusion model explanation, I have tried to keep the notation and illustrations as close as possible to Section 1.2.1 (Variational Autoencoders) to aid understanding and comparison between both methods.

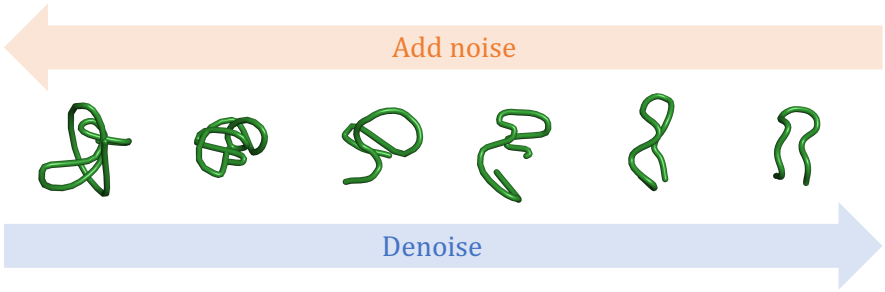


Figure 1.11: Simple illustration of a diffusion model, where noise can be added sequentially to a sample, and the model learns how to denoise the corrupted sample. The molecule shown in this example is the central carbon (C_α) trace of chignolin, a peptide (*i.e.* a small protein) of 10 amino acids that will be one of the test cases in Chapter 4.

The relationship between diffusion models and VAEs will be discussed in Section 1.2.3 ("Diffusion models versus variational autoencoders").

Setup We aim to construct a probabilistic model p with parameters θ that approximates the distribution q over data: $p_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx q(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^d$ denotes a d-dimensional data point and q does not depend on any parameters. If we want to compute the normalized probability density function, we run into a partition function Z, which is a normalizing constant that depends on $\boldsymbol{\theta}$ and usually involves solving an analytically intractable integral:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{\tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x})}{\int_{\boldsymbol{x}\in\mathbb{R}^n} \tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x}} = \frac{1}{Z(\boldsymbol{\theta})} \tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{1.20}$$

where $\tilde{p}$ is the non-normalized version of p. This situation is very similar to the variational autoencoder setting, see Eq. (1.3) in Section 1.2.1.

There are generative models that can estimate exact densities, such as autoregressive models and normalizing flows [94]. Even though discussing these models in detail is beyond the scope of this thesis, it is important to acknowledge that they come with other drawbacks. Autoregressive models use the past values of a data series to predict future values. Many different variants of autoregressive models have been developed over the years for a variety of

applications [95–100]. However, the model requires ordered data, has difficulties is retaining long-term information, and has a slow generation process due to its autoregressive nature [94]. Normalizing flows apply a series of invertible (*i.e. bijective*) transformations to convert a complex data distribution into a simple distribution that can be sampled from. Normalizing flows come in different shapes and sizes and can be applied in different fields [49, 101–105]. Drawbacks of this type of model are slow and difficult training, scaling limitations, and constrained expressivity due to the need for bijective mappings [94]. That being said, both autoregressive and normalizing flow generative models can evaluate exact (normalized) densities.

However, when we do not have a tractable likelihood, there is the alternative option to adopt a non-normalized model, also called an energy based model (EBM). Here, I will focus on a subset of non-normalized models based on *score matching*, which I will explain more elaborately in the next section as we carefully build up to describing diffusion models. Even though we are assuming a non-normalized model, score-based models might still be able to provide an estimate of (a lower bound of) the log-likelihood, similar to VAEs, which I will get back to later.

Score Matching

To understand score-based models such as diffusion models, we first need to define what a *score*, also called the *score function*, is. Essentially, a score is an indicator for how a distribution changes with respect to its inputs. In other words, it is the *gradient* of the probability distribution landscape. See Fig. 1.12 for a simple example of what this could look like.

In *score matching* (SM), the goal is to match the score function of the model density, denoted with $\boldsymbol{s}_{\boldsymbol{\theta}}$, to the score function of the data distribution, denoted with $\boldsymbol{s}$ [106]. The score function corresponds to the gradient of the log-density of the distribution with respect to the inputs:

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log \frac{1}{Z(\boldsymbol{\theta})} \tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log \tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{1.21}$$

$$\boldsymbol{s}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log q(\boldsymbol{x}), \tag{1.22}$$

where we assume that q is differentiable with respect to $\boldsymbol{x}$. Note that the score function does not depend on the normalization constant Z of the distribution. Intuitively, matching these two scores means aligning the first-order

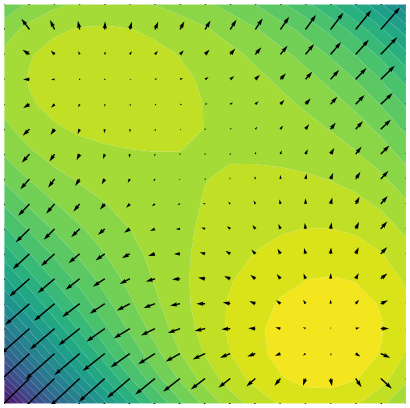


Figure 1.12: A score represents the gradient of the probability distribution, exemplified here by a vector field of a 2D density landscape.

derivatives of the logarithm of both distributions. More formally, the objective is to find the optimal parameters to minimize the expected squared distance between the model and data score functions, *i.e.* we want to minimize the objective function $\mathcal{L}_{\text{SM}_{\text{explicit}}}$:

$$\mathcal{L}_{\text{SM}_{\text{explicit}}} := \frac{1}{2}\mathbb{E}_q \|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \boldsymbol{s}(\boldsymbol{x})\|^2 \tag{1.23}$$

This is known as *explicit* score matching. However, evaluating this objective function requires either knowing the gradient of the data distribution log-density or somehow getting a non-parametric estimate of it, which is often non-trivial. Luckily, this problem can be circumvented by applying integration by parts [106, Theorem 1], resulting in the following objective for *implicit* score matching:

$$\mathcal{L}_{\text{SM}_{\text{implicit}}} := \mathbb{E}_q \left[\text{Tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})) + \frac{1}{2}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] + C, \tag{1.24}$$

where $\nabla_{\boldsymbol{x}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})$ corresponds to the Hessian of the log-density and C is a constant that does not depend on $\boldsymbol{\theta}$.

Even though implicit score matching fixes one problem, it gives rise to another: computing the Hessian of the log-density distribution is often expensive from a computational perspective ($\mathcal{O}(d^2)$ in space and time), especially for high-dimensional settings. Computing the Hessian through *e.g.* backpropagation can be very slow [107]. One way around this issue is to use *sliced* score matching [108], where the high-dimensional problem is transformed into a one-dimensional problem by projecting onto a random direction and comparing scores in that direction.

Denoising Score Matching

An alternative score matching approach that avoids calculating the Hessian altogether is *denoising* score matching (DSM) [109]. This method can be regarded as a transition step from score matching towards diffusion models since the underlying theory and objective are very similar. DSM combines ideas from score matching and denoising autoencoders [110, 111]. In denoising autoencoders, input samples are artificially corrupted in multiple steps, where at each step:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}, \tag{1.25}$$

with $\boldsymbol{x}$ the input sample, $\tilde{\boldsymbol{x}}$ the corrupted sample and $\boldsymbol{\epsilon}$ the added noise, which is usually normally distributed, *i.e.* $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2\mathbf{I})$ where $\boldsymbol{\sigma}^2$ denotes the variance. This yields the following conditional density:

$$q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}\boldsymbol{\sigma}} \exp^{-\frac{1}{2\boldsymbol{\sigma}^2}\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|^2}.$$

A variational autoencoder is then trained such that the encoder learns how to encode the corrupted sample $\tilde{\boldsymbol{x}}$ into a latent representation $\boldsymbol{z}$, while the decoder maps the latent representation to $\hat{\boldsymbol{x}}$, a reconstruction of the input sample. The goal is to minimize the squared difference between the reconstructed sample and the original input $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2$.

In DSM, the gradient of the conditional log-density, $\nabla_{\tilde{\boldsymbol{x}}} \log q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$, is matched to the score function $\boldsymbol{s}_{\boldsymbol{\theta}}$ predicted by a model. This leads to the following objective for the joint density $q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}}|\boldsymbol{x})q_0(\boldsymbol{x})$, with $q_0(\boldsymbol{x})$ the density at zero noise ($\boldsymbol{\sigma} = 0$), *i.e.* the empirical probability density function associated with the data:

$$\mathcal{L}_{\text{DSM}} := \mathbb{E}_{q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}},\boldsymbol{x})}\left[\frac{1}{2}\|\boldsymbol{s}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}) - \nabla_{\tilde{\boldsymbol{x}}} \log q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}}|\boldsymbol{x})\|^2\right]. \tag{1.26}$$

In words, the gradient of the conditional log-density will bring us closer towards the noise-free input, and we want our model to match that as closely as possible. In a setting where the model is highly expressive and enough data is available, the score given optimal parameters $\boldsymbol{s}_{\boldsymbol{\theta}^*}(\tilde{\boldsymbol{x}})$ will approximately match the score $\nabla_{\tilde{\boldsymbol{x}}} \log q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}})$, where the score is no longer conditioned on $\boldsymbol{x}$ [109]. Moreover, at low noise levels we have that $\boldsymbol{s}_{\boldsymbol{\theta}^*}(\tilde{\boldsymbol{x}}) \approx \nabla_{\boldsymbol{x}} \log q_0(\boldsymbol{x})$, since the joint density $q_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ will resemble the data distribution $q_0(\boldsymbol{x})$ [83].

Diffusion Models: SMLD and DDPM

Diffusion models learn a denoising process for samples to which noise has been added slowly in multiple steps and are thereby able to generate new samples from noise, as illustrated in Figure 1.11. A more detailed view on diffusion models is shown in Fig. 1.13. The increasingly noisy representations of the input can be seen as a diffusion model's "latent space", which is why from now on I will denote noisy samples by $\boldsymbol{z}_i$, with $i = 0, \ldots, L$ the noise-adding step, $\boldsymbol{z}_0 = \boldsymbol{x}$ the original sample and $\boldsymbol{z}_L$ random noise where all information has been destroyed.

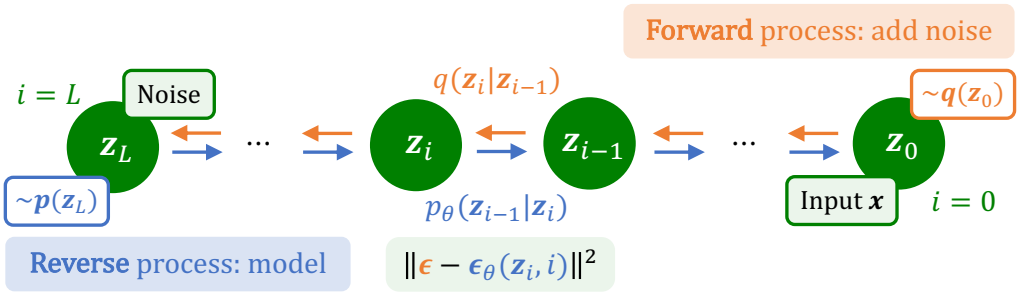


Figure 1.13: Detailed schematic representation of a diffusion model. Noise is added during the forward process, while in the reverse process the model learns how to denoise the sample. The training objective for DDPMs, which is to minimize the difference between the predicted noise and the ground truth noise, is shown at the bottom of the figure in a light green box.

There are multiple variants of diffusion models, of which the two main approaches I will focus on here are denoising diffusion probabilistic models (DDPM) [84] and score matching with Langevin dynamics (SMLD) [83]. I will discuss both models, but DDPM will be explained in more detail since it is the model that is relevant for Chapter 4 of this thesis. Both DDPM and SMLD gradually add Gaussian noise to the input sample in the forward process and both methods in one way or another compute denoising scores. However, there are distinct differences between both models and especially their equilibrium distributions. SMLD is a DSM-based method (see "Denoising score matching") that adds noise according to Eq. (1.25). As the number of noise-adding steps i goes to infinity, SMLD therefore has a finite mean (which is equivalent to the original input) and an exploding variance:

$$q(\boldsymbol{z}_i|\boldsymbol{z}_{i-1}) = \mathcal{N}(\boldsymbol{z}_{i-1}, \sigma_i^2\mathbf{I}) \tag{1.27}$$

$$\lim_{i\to\infty} q(\boldsymbol{z}_i|\boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_0, \infty\cdot\mathbf{I}) \tag{1.28}$$

In contrast, DDPM applies a scaling factor to the input while also adding noise, both dependent on an increasing noise scale $0 < \beta_i < 1$, such that the limit distribution becomes standard Gaussian:

$$q(\boldsymbol{z}_i|\boldsymbol{z}_{i-1}) = \mathcal{N}(\sqrt{\bar{\alpha}_{i-1}}\boldsymbol{z}_{i-1}, (1-\bar{\alpha}_{i-1})\mathbf{I}) \tag{1.29}$$

$$\lim_{i\to\infty} q(\boldsymbol{z}_i|\boldsymbol{z}_0) = \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1.30}$$

with $\alpha_i = 1 - \beta_i$ and $\bar{\alpha}_i = \prod_{j=1}^{i} \alpha_j$. There is also a variance preserving variant of SMLD, mentioned in Song et al. [112, Section 3.4]. Both for SMLD and DDPM, q does not depend on any network parameters, since the added noise is just sampled from a known distribution and there are no learned parameters involved.

SMLD and DDPM also differ in their learning objectives. In SMLD, the loss is based on denoising score matching (see Eq. (1.26)), meaning that in optimal circumstances (*i.e.* for optimal parameters, small noise levels and an infinite amount of time steps) the learned score will approximate the true scores. Subsequently, sampling is done using Langevin dynamics steps (see [83] and Section 1.1.3). In DDPM, on the other hand, the objective is to maximize the evidence lower bound (ELBO) for every step, treating the

corrupted data as latent variables:

$$\begin{aligned}\mathcal{L}_{\text{DDPM}} &:= \sum_{i=0}^{L} \mathcal{L}_{\text{ELBO},i} \\ &= \mathbb{E}_q \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_{0:L})}{q(\boldsymbol{z}_{1:L}|\boldsymbol{z}_0)} \right] \\ &= \underbrace{\mathbb{E}_q \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_0|\boldsymbol{z}_1) \right]}_{\mathcal{L}_{\text{I}}} - \sum_{i=1}^{L-1} \underbrace{\mathcal{D}_{\text{KL}} \left(q(\boldsymbol{z}_i|\boldsymbol{z}_{i+1}, \boldsymbol{z}_0) || p_{\boldsymbol{\theta}}(\boldsymbol{z}_i|\boldsymbol{z}_{i+1}) \right)}_{\mathcal{L}_{\text{II}}} \\ &\quad - \underbrace{\mathcal{D}_{\text{KL}}(q(\boldsymbol{z}_L|\boldsymbol{z}_0)||p(\boldsymbol{z}_L))}_{\mathcal{L}_{\text{III}}}. \end{aligned} \tag{1.31}$$

Here, $\mathcal{L}_{\text{I}}$ is the expected reconstruction error for the first diffusion step, $\mathcal{L}_{\text{II}}$ are computable KL divergences between Gaussians, and $\mathcal{L}_{\text{III}}$ is the divergence between the distribution over noisy samples given the input samples at final step L and the "prior" we have over noise. $\mathcal{L}_{\text{III}}$ can be disregarded since it has no learned parameters and is therefore constant over training. Moreover, $\mathcal{L}_{\text{III}}$ approaches zero for a sufficient number of diffusion steps L, since large amounts of added noise will eventually always converge to the standard Gaussian distribution in a DDPM as shown in Eq. (1.30). In the end, we are left with the by now familiar reconstruction term and a number of KL divergences. Here it is important to note that even though the so-called *forward process posterior* $q(\boldsymbol{z}_{i-1}|\boldsymbol{z}_i)$ is intractable, it becomes tractable when conditioned on $\boldsymbol{z}_0$:

$$\begin{aligned} q(\boldsymbol{z}_{i-1}|\boldsymbol{z}_i, \boldsymbol{z}_0) &= \mathcal{N}(\tilde{\boldsymbol{\mu}}_i(\boldsymbol{z}_i, \boldsymbol{z}_0), \tilde{\beta}_i \mathbf{I}) \\ \text{where } \tilde{\boldsymbol{\mu}}_i(\boldsymbol{z}_i, \boldsymbol{z}_0) &= \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{z}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\boldsymbol{z}_0 \\ \text{and } \tilde{\beta}_i &= \frac{1-\bar{\alpha}_{i-1}}{1-\bar{\alpha}_i}\beta_i. \end{aligned}$$

Next, we can remove the dependency of $\boldsymbol{\mu}_i$ on $\boldsymbol{z}_0$. Through the reparameterization trick we can write $\boldsymbol{z}_0$ as a function of $\boldsymbol{z}_i$:

$$\begin{aligned} \boldsymbol{z}_i &= \sqrt{\bar{\alpha}_i}\boldsymbol{z}_0 + \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}_i, \text{with } \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \Rightarrow \boldsymbol{z}_0 &= \frac{1}{\sqrt{\bar{\alpha}_i}} \left(\boldsymbol{z}_i - \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}_i \right) \end{aligned}$$

Substituting $\boldsymbol{z}_0$ in $\boldsymbol{\mu}_i$ with the expression above gives the following result:

$$\tilde{\boldsymbol{\mu}}_i = \frac{1}{\sqrt{\alpha_i}} \left(\boldsymbol{z}_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}} \boldsymbol{\epsilon}_i \right).$$

In a similar way, we can parameterize the model mean $\boldsymbol{\mu}_\theta$ as a function of $\boldsymbol{z}_i$ and a neural-network parameterized noise source $\boldsymbol{\epsilon}_\theta$:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{i-1}|\boldsymbol{z}_i) &= \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta},i}, \sigma_i^2) \\ \text{where } \boldsymbol{\mu}_{\boldsymbol{\theta},i} &= \frac{1}{\sqrt{\alpha_i}} \left(\boldsymbol{z}_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_i, i) \right) \end{aligned}$$

with σ_i either learned by the model ($\sigma_{\boldsymbol{\theta},i}$) or set as a hyperparameter, which will be assumed here. For fixed σ_i, $\mathcal{L}_{\text{II}}$ can be reformulated such that the effective objective becomes minimizing the difference between $\tilde{\boldsymbol{\mu}}_i$ and $\boldsymbol{\mu}_{\boldsymbol{\theta},i}$ [84]:

$$\mathcal{L}_{\text{II}} = \mathbb{E}_{q(\boldsymbol{z}_0), \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{\beta_i^2}{2\alpha_i(1-\bar{\alpha}_i)\|\sigma_i\|_2^2} \|\boldsymbol{\epsilon}_i - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\underbrace{\sqrt{\bar{\alpha}_i}\boldsymbol{z}_0 + \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}_i}_{\boldsymbol{z}_i}, i)\|^2 \right]. \tag{1.32}$$

Ho et al. [84] suggest a simplified version of this expression to use in DDPM training, which also covers the reconstruction term $\mathcal{L}_{\text{I}}$ when $i = 1$:

$$\lambda_{\text{DDPM},i} = \mathbb{E}_{q(\boldsymbol{z}_0), \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\|\boldsymbol{\epsilon}_i - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_i, i)\|^2 \right] + C, \tag{1.33}$$

where C can be omitted during training since it does not depend on $\boldsymbol{\theta}$ when σ_i is set and not learned. For $i = 1$, this loss function corresponds to $\mathcal{L}_{\text{I}}$: the reconstruction loss for a Gaussian distribution with fixed sigma becomes the mean squared error (MSE) between the ground truth and the reconstruction, which is in turn equivalent to the MSE between the predicted noise at $i = 1$ and the true noise that was added in the forward process. In other words, a perfect reconstruction is obtained when the predicted noise equals the forward process noise. For $i > 1$, Eq. (1.33) corresponds to an unweighted version of Eq. (1.32), which often leads to better sample quality in practice, possibly because less weight is given to the more simple denoising tasks at small values of i [84].

The effective objective for the model according to Eq. (1.33) is to predict the level of noise for a given $\boldsymbol{z}_i$. During training of a DDPM model, a noise level i is picked at random. The predicted noise is then matched to the ground truth noise, where obtaining the ground truth is trivial since we can make any jump in the forward process due to the fact that all added noise is Gaussian distributed. Here, it is important to note that the parameters for every step in the generative process are shared. Moreover, it has been shown that the loss in Eq. (1.33) implicitly minimizes the DSM loss with weights γ_i [83, 84]:

$$\lambda^{\text{DSM}}_{\text{DDPM},i} = \mathbb{E}_{q(\boldsymbol{z}_0)}\mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{z}_0)}\gamma_i \|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_i) - \nabla_{\boldsymbol{z}_i} \log q(\boldsymbol{z}_i|\boldsymbol{z}_0)\|^2, \tag{1.34}$$

which is indeed very similar to $\mathcal{L}_{\text{DSM}}$ in Eq. (1.26). For this DSM objective, we know that for a highly expressive model, enough data and low noise levels, the predicted score approximates the true score [83]. An equivalent statement holds for optimizing $\lambda^{\text{DSM}}_{\text{DDPM},i}$ with low noise levels for each noise-adding step i, which can be ensured by taking the limit to infinity for the number for noise-adding steps.

Score versus conservative energy gradient Given optimal parameters and a sufficient number of noise adding steps, diffusion models aim to estimate a score, either explicitly (SMLD) or implicitly (DDPM). However, a straightforward parameterization of the score with a neural network does not enforce the constraint that $\nabla_{\boldsymbol{z}_i} \log q(\boldsymbol{z}_i)$ is a conservative vector field, *i.e.* that any line integral between two points within the vector field is independent of the path taken. Enforcing this constraint would be plausible since we are modeling a Boltzmann distribution where energy is conserved: $q(\boldsymbol{z}_0) \sim \exp -V(\boldsymbol{z}_0)/(k_B T)$, with $V(\boldsymbol{z}_0)$ the potential energy, k_B the Boltzmann constant and T the temperature. The gradient of the log density corresponds to $-\nabla_{\boldsymbol{z}_i} V(\boldsymbol{z}_0) = F(\boldsymbol{z}_0)$, with $F(\boldsymbol{z}_0)$ the conservative forces. This property could be realized by parameterizing the learned score such that we explicitly calculate the gradient of a scalar energy function: $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_i) = -\nabla_{\boldsymbol{z}_i} V_{\boldsymbol{\theta}}(\boldsymbol{z}_i)$. Although this parameterization could serve as a strong inductive bias, it could also substantially change the optimization landscape through the added constraints. In Salimans and Ho [113], it is argued that as long as you take care of the way you build in these constraints, similar results can be obtained with both parameterizations.

Diffusion Models: Conclusion

Diffusion models learn how to denoise samples to which noise has been added in a multi-step process. These models are flexible and expressive, and can provide an estimate of the lower bound of the log-likelihood. What I have not touched upon here is that it is also possible to generalize the number of noise steps to infinity, giving rise to the stochastic differential equation (SDE) formulation of diffusion models, which can be used to compute exact log-likelihoods [112, Appendix D.2].

Apart from being expressive, diffusion models are relatively fast to train and scale well to large inputs. A drawback is that the generative process of the simplest version of a diffusion model is slow due to the step-wise sampling process. However, tricks can be applied to speed up generation [114].

1.2.3 Diffusion Models versus Variational Autoencoders

As I have hinted at earlier, diffusion models are in many ways similar to deep hierarchical VAEs. I have illustrated the similarities and differences in Fig. 1.14. First of all, both models have a generative process where a latent variable is sampled and subsequently transformed over multiple steps into outputs that resemble the training data. Secondly, looking at Fig. 1.14 and comparing Fig. 1.10 to Fig. 1.13, there is a strong resemblance between the encoder of a VAE and the forward process of a diffusion model, since in both cases a data point gets converted into a sequence of latent representations. In addition, both VAEs and DDPMs are optimized using the same objective, namely maximizing the lower bound on the log evidence.

Of course, there are also differences between the two approaches. Diffusion models have shared parameters for each step in the generative chain, whilst in hierarchical VAEs a different network is typically trained to transition from one latent space layer to another. This makes diffusion models a bit more flexible, since they are trained to handle many noise levels with one set of parameters given noise-step i. Moreover, while there are similarities between the encoder of a VAE and the forward process of a diffusion model, the latter does not have any learned parameters. Finally, all intermediate representations in a diffusion model have to be of the same size as the input, while a VAE allows for dimensionality reduction (*i.e.* a bottleneck). This difference is linked to how much value is assigned to the latent representation

in both models. In VAEs, the latent space is often intended as a reduced representation of the input, which still contains essential information about the data point. This representation can then in principle be used in different downstream tasks. In contrast, diffusion models mostly aim to get plausible samples, and all information about the input gets destroyed as it is transformed into random noise. Even though this means the latent representations cannot easily be used for downstream tasks, it also gives the model a lot of flexibility and, in most cases, good performance.

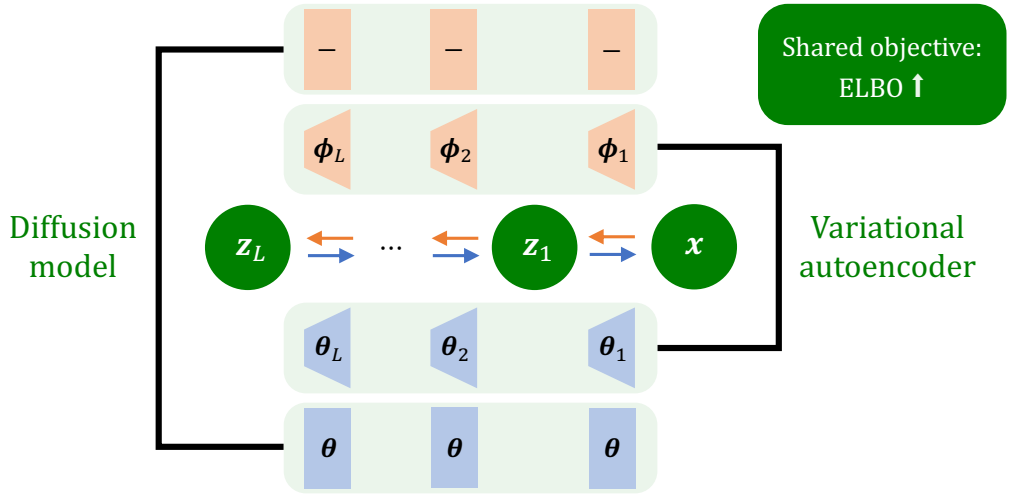


Figure 1.14: Conceptual similarities and differences between variational autoencoders and diffusion models.

Chapter 2

Overview

My Ph.D. journey took some unexpected turns. When I started, we set out to develop new methods for protein folding using a variety of different machine learning models (amongst others LSTMs, residual neural networks, autoregressive networks and dilated convolutional networks). However, after the release of Alphafold2 [1], we decided to move away from static structure prediction and redirect our attention towards ensembles of structures and protein dynamics. Moreover, we ended up focusing on generative models, which was a natural choice since modeling ensembles is much more valuable if one can also sample from the learned distribution. Below, I provide a short description for each of the paper chapters.

Chapter 3: Internal-Coordinate Density Modelling of Protein Structure: Covariance Matters

From our own experiences as well as existing protein structure modeling literature, it is clear that the choice of structure representation is often of great importance. The two most common parameterizations are Cartesian coordinates and internal coordinates. Cartesian coordinates model global structure well, but struggle with local chemical integrity and require either a rotation and translation invariant model or data augmentation. Internal coordinates, on the other hand, satisfy local constraints, but it is non-trivial to capture global structural integrity for this representation since perturbations lead to rigid-body movements downstream of the perturbed atom. Therefore, a

complex covariance structure is needed to satisfy global structural constraints when modelling protein structure using internal coordinates. In Chapter 3, this is exactly the problem we address, as illustrated in Fig. 2.1 (under review for ICML 2023). Inspired by Favrin et al. [115] and earlier work of my supervisor Wouter Boomsma concerning local moves [116], we derived a full covariance structure over internal coordinates for ensembles of full proteins by placing constraints in 3D space. As a proof-of-concept, we incorporate our method into a variational autoencoder and show high-quality generated samples, both locally locally and globally.

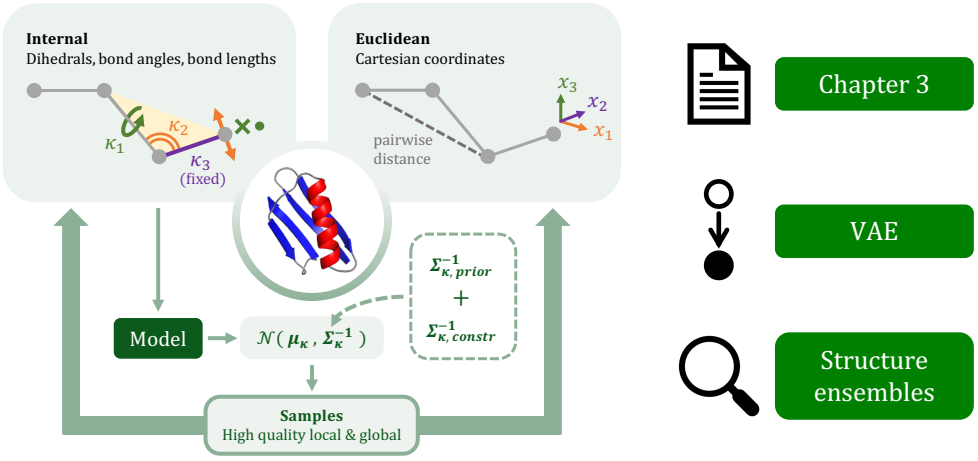


Figure 2.1: The paper in Chapter 3, currently under review for ICML 2023, describes a method to capture a rich covariance structure over internal coordinates that incorporates global constraints for ensembles of proteins. A VAE is presented as an example of how the covariance structure could be leveraged in practice. Figure adapted from Fig. 1 in the corresponding paper.

Chapter 4: Two for One: Diffusion Models and Force Fields for Coarse-Grained Molecular Dynamics

In the final year of my Ph.D., I got the opportunity to go to Amsterdam, the Netherlands, for a four month internship at Microsoft Research (AI4Science). My own research interests and the extensive machine learning expertise that was present in the group culminated in a project where a diffusion model is

used for density modeling of coarse-grained protein dynamics data, inspired by Köhler et al. [49] for certain modeling and evaluation choices. The resulting paper (under review for ICML 2023) is presented in Chapter 4. We exploit the fact that diffusion models implicitly learn a score such that we can, in addition to using the learned equilibrium distribution to draw samples i.i.d., also extract a coarse-grained force field to use in molecular dynamics simulations, as illustrated in Fig. 2.2. We show that our method outperforms baselines for equilibrium metrics as well as transition probability dynamics metrics for small- to medium-sized proteins exhibiting folding and unfolding events [44].

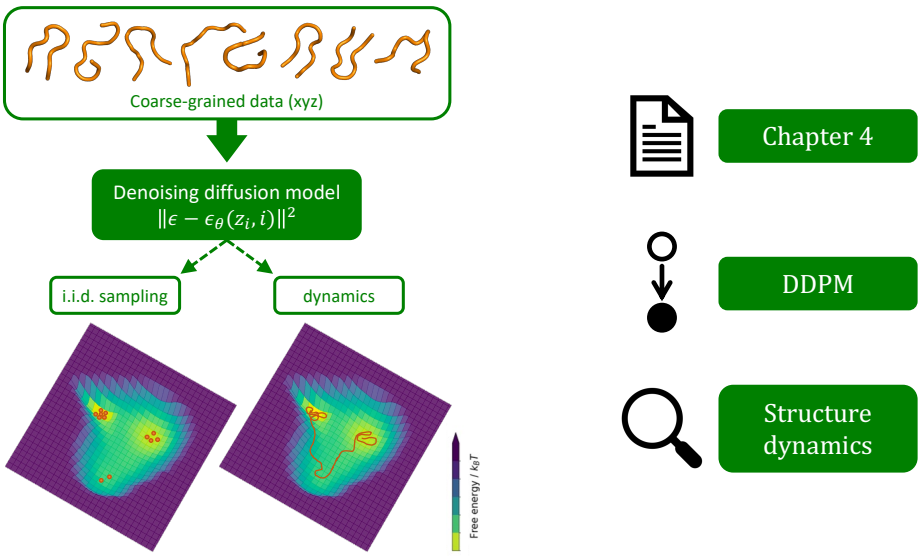


Figure 2.2: The paper in Chapter 4, currently under review for ICML 2023, presents a diffusion model that takes coarse-grained molecular dynamics data as input and learns both an equilibrium distribution from which samples can be drawn i.i.d. and a coarse-grained force field that can be used for molecular dynamics. Figure adapted from Fig. 1 in the corresponding paper.

Chapter 5: Sampling quality in deep generative models of protein sequences

Although generative models can be a powerful tool, they are not without flaws. The final paper of this thesis, under review for UAI 2023, investigates

latent representations of protein families in VAE-based models. First shown by Riesselman et al. [11], training a one-layer VAE on protein family sequence (MSA) data results in a "star-shaped" aggregated posterior where different species separate in latent space. However, a standard Gaussian prior hardly seems appropriate for this type of representation considering the considerable amount of areas where probability mass will be placed despite the absence of observed data points, as depicted in Fig. 2.3. This has a negative effect on the quality of the samples generated by the model. We investigate this issue in Chapter 5 and show that using a multi-layer LVAE somewhat alleviates this problem. We further show that the use of a Bayesian decoder results in lower sample quality and report on the effect of introducing a temperature scale on the posterior of the weights of the decoder. This work was partly done during my change of environment at DTU.

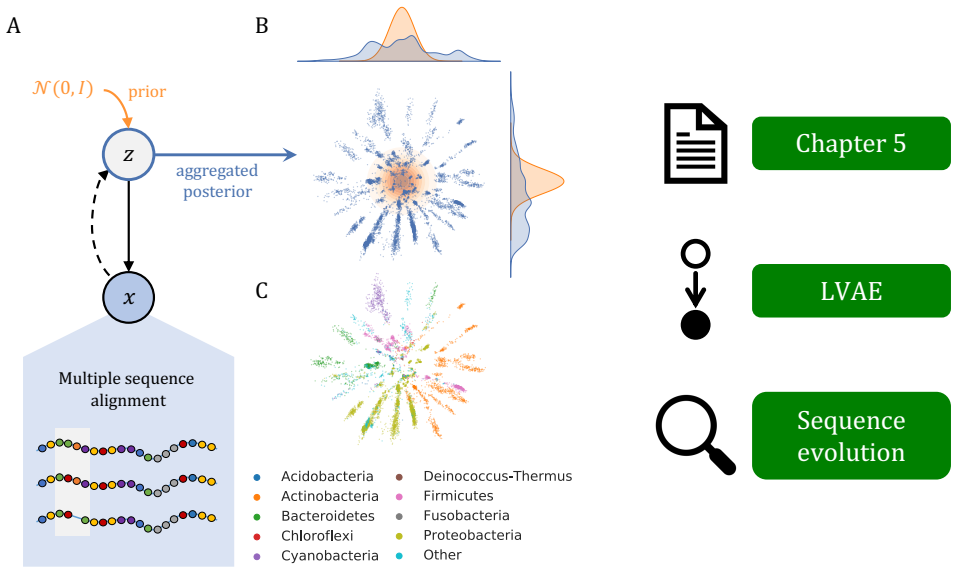


Figure 2.3: The paper in Chapter 5, currently under review for UAI 2023, investigates sampling quality in VAE models trained on protein family sequence data. We show how the use of an LVAE somewhat reduces the discrepancies between the prior and the aggregated posterior, and investigate the effects of a Bayesian decoder on generated samples. Figure adapted from Fig. 1 in the corresponding paper.

Chapter 3

Internal-Coordinate Density Modelling of Protein Structure: Covariance Matters

Marloes Arts, Jes Frellsen, Wouter Boomsma

The work presented in this chapter was submitted to ICML 2023 and is currently under review. A preprint is available on arXiv: `https://arxiv.org/abs/2302.13711`.

Internal-Coordinate Density Modelling of Protein Structure: Covariance Matters

Marloes Arts [1] **Jes Frellsen** [2] **Wouter Boomsma** [1]

Abstract

After the recent ground-breaking advances in protein structure prediction, one of the remaining challenges in protein machine learning is to reliably predict distributions of structural states. Parametric models of small-scale fluctuations are difficult to fit due to complex covariance structures between degrees of freedom in the protein chain, often causing models to either violate local or global structural constraints. In this paper, we present a new strategy for modelling protein densities in internal coordinates, which uses constraints in 3D space to induce covariance structure between the internal degrees of freedom. We illustrate the potential of the procedure by constructing a variational autoencoder with full covariance output induced by the constraints implied by the conditional mean in 3D, and demonstrate that our approach makes it possible to scale density models of internal coordinates to full-size proteins.

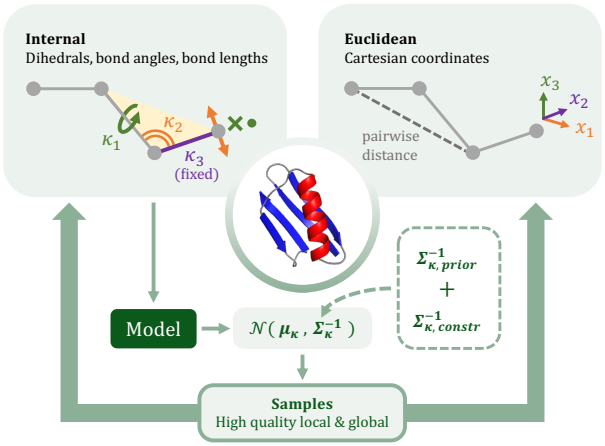


Figure 1. Proteins can be represented in local and global coordinates. Internal coordinates are fed as input to a model (in our case a VAE), which predicts a mean and full covariance structure over internal coordinates based on a prior plus specific constraints on atom fluctuations. The resulting model can generate high-quality samples, in terms of both local and global structure.

1. Introduction

Proteins are macro-molecules that are involved in nearly all cellular processes. Most proteins adopt a compact 3D structure, also referred as the *native state*. This structure is a rich source of knowledge about the protein, since it provides information about how the protein can engage biochemically with other proteins to conduct its function. The machine learning community has made spectacular progress in recent years in the prediction of the native state from the amino acid sequence of a protein (Jumper et al., 2021; Senior et al., 2020; Wu et al., 2022b; Baek et al., 2021; Wu et al., 2022a). However, the static picture of the structure of a protein is misleading: in reality a protein is continuously moving, experiencing both thermal fluctuations and larger conformational changes, both of which affect its function. One of the remaining challenges in machine learning for structural biology is to reliably predict these distributions of states, rather than just the most probable state. We discuss the state of the density modelling field in Section 5 (Related work).

Modelling the probability density of protein structure is non-trivial, due to the strong constraints imposed by the molecular topology. The specific challenges depend on the chosen structural representation: if a structure is represented by the 3D coordinates of all its atoms, these atom positions cannot be sampled independently without violating the physical constraints of e.g. the bond lengths separating the atoms. In addition, an arbitrary decision must be made about how the structure is placed in a global coordinate system, which implies that operations done on this representation should preferably be invariant or equivariant to this choice. An alternative representation is to parameterize the structure using internal coordinates, i.e. in terms of bond lengths, bond angles and torsion/dihedral angles (rotations around the bonds). The advantage of this representation is that

[1]Department of Computer Science, University of Copenhagen, Copenhagen, Denmark [2]Department of Applied Mathematics and Computer Science, Technical University of Denmark, Copenhagen, Denmark. Correspondence to: Marloes Arts <ma@di.ku.dk>.

Preprint.

internal degrees of freedom can be sampled independently without violating the local bond constraints of the molecule. It also makes it possible to reduce the number of degrees of freedom to be sampled – for instance fixing the bond lengths to ideal values, since they fluctuate much less than the torsion angles and bond angles.

For the reasons given above, an internal coordinate representation would appear to be an attractive choice for density modelling. However, one important problem reduces the appeal: small fluctuations in internal coordinates will propagate down the chain, leading to large fluctuations of atoms remotely downstream in the protein chain. As a consequence, internal-coordinate density modelling necessitates careful modelling of the covariance structure between the degrees of freedom in order to ensure that small-scale fluctuations in internal coordinates result in small perturbations of the 3D coordinates of the protein. Such covariance structures are typically highly complex, making direct estimation difficult.

In this paper, we investigate whether density modelling of full-size proteins in internal coordinates is feasible. We empirically demonstrate the difficulty in estimating the covariance structure of internal coordinates from data, and instead propose a technique for *inducing* the covariance structure by imposing constraints on downstream atom movement using the Lagrange formalism. Rather than estimating the covariance structure from scratch, we can instead modulate the covariance structure by choosing appropriate values for allowed fluctuations of downstream atoms. We demonstrate the procedure in the context of a variational autoencoder (Fig. 1). Given a prior on the internal coordinate fluctuations and a predicted mean, we impose constraints on the atom fluctuations in 3D coordinates to obtain a full covariance structure over the internal coordinates. We show that this allows us to generate valid structures in terms of both internal and Cartesian coordinates. Our method is validated in two very different test cases: a small set of nmr structures on an α-helical protein and a molecular dynamics dataset on the larger and more complex protein G. We anticipate that this method could serve as a building block applicable more generally for internal-coordinate density estimation, for instance for internal-coordinate denoising diffusion models.

Our main contributions are:

- We formulate a procedure for inducing full-protein covariance structures in internal coordinates (bond angles and torsional angles), based on constraints on atom fluctuations in 3D space.
- We design a variational autoencoder which models fluctuations for full-length proteins in internal coordinates. Despite the fact that constraints are expressed in terms of Cartesian coordinates, the model is not dependent on a global reference frame (i.e. it is rotationally invariant).
- We demonstrate that our model provides meaningful density estimates on ensemble data for proteins obtained from experiment and simulation. To our knowledge, it is the first model to reliably estimate internal coordinate densities at this scale.

Scope. Our focus in this paper will be on modelling distributions of protein structure states in internal coordinates. We are thus concerned with thermodynamic ensembles, rather than the detailed dynamics that a molecule undergoes. Dynamics could potentially be modelled on top of our approach, for instance by fitting a discrete Markov model to describe transitions between states, and using our approach to model the thermal fluctuations within a state, but this is beyond the scope of the current work.

Another perspective on our approach is that we wish to describe the aleatoric uncertainty associated with a structure deposited in the Protein Data Bank, or a structure predicted by a protein structure prediction procedure such as Alphafold (Jumper et al., 2021).

2. Background

2.1. Cartesian vs internal coordinates

As stated before, Cartesian coordinates and internal coordinates each have advantages and disadvantages. Assume we have a 3D protein structure in Euclidean space with atom positions $\boldsymbol{x}$. Throughout this paper, we only consider backbone atoms N, C_α and C, which means that the total number of atoms M is equal to three times the number of amino acids. The Euclidean setting thus results in $3 \times M$ coordinates. Even though in this setting each of the atoms can fluctuate without affecting other atoms in the backbone chain, there is no guarantee for chemical integrity, i.e. conservation of bond lengths and respecting van der Waals forces. This can lead to backbone crossings and generally unphysical protein structures.

One way to ensure chemical integrity is by parameterizing the protein structure in internal coordinate space using dihedrals $\boldsymbol{\kappa_1}$, bond angles $\boldsymbol{\kappa_2}$ and bond lengths $\boldsymbol{\kappa_3}$. Here, dihedrals are torsional angles that twist the protein around the bond between two consecutive atoms, bond angles are angles within the plane that is formed by two consecutive bonds, and bond lengths are the distances between two consecutive backbone atoms. Since bond length distributions have very little variance, we choose to fix them, thereby reducing the number of variables over which we need to estimate the covariance. We will refer to the remaining two internal coordinates together as $\boldsymbol{\kappa}$ to avoid notation clutter.

As dihedrals can only be defined by four points (where the dihedral is the angle between the plane defined by the first three points and the plane defined by the last three points) and bond angles can only be defined by three points, the resulting protein structure representation will have $(2\times M)-5$ coordinates. Not only does this system result in less coordinates to determine a full covariance structure over, the coordinates are also automatically rotation and translation invariant, as opposed to Cartesian coordinates.

The remaining problem is that small changes in one internal coordinate can have large consequences for the global structure of the protein, since all atoms downstream of the internal coordinate will move together, acting like a rigid body. It is therefore challenging to preserve global structure while altering internal coordinates, since they are mostly descriptive of local structure.

2.2. Standard precision estimators do not capture global structure fluctuations

Because of the limitations of internal coordinates mentioned in Section 2.1 it is a highly non-trivial task to capture a full covariance structure over $\boldsymbol{\kappa}$ which also conforms to constraints in Euclidean space that are inherent to the protein. As an example, we use a standard estimator to get a precision matrix (i.e. the inverse of the covariance matrix) over $\boldsymbol{\kappa}$ for a short molecular dynamics simulation on "1pga", also known as "protein G" (Fig. 2). Details about the simulation can be found in Appendix A. We see that when we take samples from a multivariate Gaussian over $\boldsymbol{\kappa}$ with the true mean (based on the dataset) and the estimated precision, the samples exhibit atom fluctuations that are much higher than the original simulation, and with a very different patterns.

3. Internal-coordinate density modelling with constraints

To overcome the limitations that regular covariance and precision estimators have, we incorporate constraints on atom fluctuations in Euclidean space.

3.1. Setup

We parameterize a 3D protein structure in terms of internal coordinates (i.e. dihedrals and bond angles, while bond lengths are kept fixed), which together will be referred to as $\boldsymbol{\kappa}$. Our aim is to obtain a multivariate Gaussian distribution over the deviations from the mean $p(\Delta\boldsymbol{\kappa})$, centered at zero, with a full precision structure. This target distribution is subject to constraints over atom fluctuations, enforcing the preservation of global structure. We have a prior $q(\Delta\boldsymbol{\kappa})$ over the internal coordinate distribution, where the mean is zero and the precision is a diagonal matrix with the diagonal filled by the inverse variance over all $\Delta\boldsymbol{\kappa}$ values, estimated

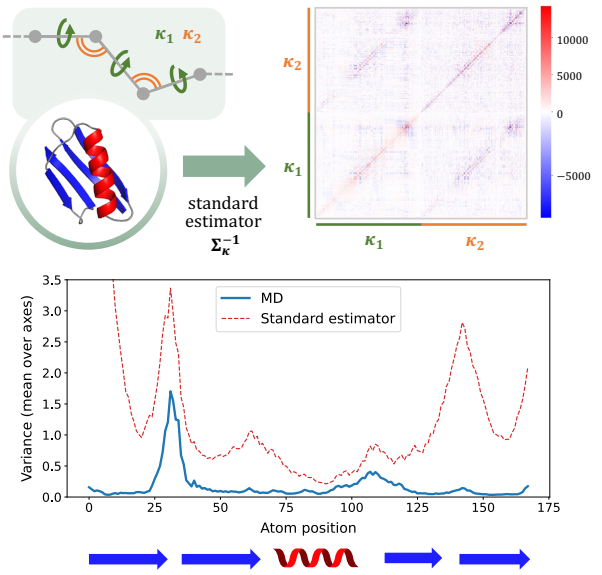


Figure 2. When a standard estimator is used to get the precision structure over internal coordinates (top row; dihedrals and bond angles, bond lengths are kept fixed), the atom fluctuations significantly deviate from MD simulations (bottom row). Blue arrows and red helices represent secondary structural elements. The variance is calculated as the mean of the variances over the x, y and z axis, in Å^2.

from our input data. The prior is defined as

$$q(\Delta\boldsymbol{\kappa}) = \frac{1}{Z_q}\exp\left(-\frac{1}{2}\Delta\boldsymbol{\kappa}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}_{\kappa,\mathrm{prior}}\Delta\boldsymbol{\kappa}\right), \tag{1}$$

where $Z_q = \boldsymbol{\sigma}_{\mathrm{data}}\sqrt{2\pi}$ is the normalization constant for the prior distribution and $\boldsymbol{\Sigma}^{-1}_{\kappa,\mathrm{prior}} = a\cdot\mathrm{diag}(\sigma^{-2}_{\kappa,\mathrm{data}})$ with a a hyperparameter that determines the strength of the prior. Our approach will be to construct a new distribution p which is as close as possible to q, but which fulfills a constraint that prohibits the downstream 3D coordinates from fluctuating too much. We thus wish to minimize the Kullback-Leibler divergence between the objective distribution and prior:

$$\mathcal{D}_{\mathrm{KL}}(p|q) = \int p(\Delta\boldsymbol{\kappa})\ln\frac{p(\Delta\boldsymbol{\kappa})}{q(\Delta\boldsymbol{\kappa})}\mathrm{d}\Delta\boldsymbol{\kappa}. \tag{2}$$

, adding constraints on the expected value over squared atom displacements of each downstream atom m:

$$\mathbb{E}_{\Delta\boldsymbol{\kappa}\sim p(\Delta\boldsymbol{\kappa})}\left[\Delta x_m^2\right] = C_m \tag{3}$$

where $\mathbb{E}_{\Delta\boldsymbol{\kappa}\sim p(\Delta\boldsymbol{\kappa})}\left[\Delta x_m^2\right]$ is the expected value for the m^{th} squared displacement and C_m is a constant equivalent to the variance of the atom position $\sigma^2_{\kappa,x}$ assuming equal variance in all directions. Since every Δx_m is a function of $\Delta\boldsymbol{\kappa}$ with probability density function $p(\Delta\boldsymbol{\kappa})$, we can use the law of

the unconscious statistician to reformulate the constraints as follows:

$$\mathbb{E}_{\Delta\boldsymbol{\kappa}\sim p(\Delta\boldsymbol{\kappa})}\left[\Delta x_m^2\right] = \int \Delta x_m^2 p(\Delta\boldsymbol{\kappa})\mathrm{d}\Delta\boldsymbol{\kappa} = \mathrm{C_m} \quad (4)$$

3.2. Lagrange formalism to incorporate constraints

We use the Lagrange formalism to incorporate M of these constraints, where M is the number of atoms. Because we need the integral over $p(\Delta\boldsymbol{\kappa})$ to sum to one, we also add this as a constraint.

$$\begin{aligned}\tilde{\mathcal{D}}_{\mathrm{KL}}(p|q) = &\int p(\Delta\boldsymbol{\kappa}) \ln \frac{p(\Delta\boldsymbol{\kappa})}{q(\Delta\boldsymbol{\kappa})}\mathrm{d}\Delta\boldsymbol{\kappa} \\ &+ \lambda_0 \left(\int p(\Delta\boldsymbol{\kappa})\mathrm{d}\Delta\boldsymbol{\kappa} - 1\right) \\ &+ \sum_{n=1}^{M} \lambda_n \left(\int \Delta x_n^2 p(\Delta\boldsymbol{\kappa})\mathrm{d}\Delta\boldsymbol{\kappa} - \mathrm{C_n}\right) \quad (5)\end{aligned}$$

This is the objective we want to minimize. Taking the derivative of $\tilde{\mathcal{D}}_{\mathrm{KL}}(p|q)$ with respect to $p(\Delta\boldsymbol{\kappa})$ and setting to zero:

$$\begin{aligned}0 &= 1 + \lambda_0 + \ln \frac{p(\Delta\boldsymbol{\kappa})}{q(\Delta\boldsymbol{\kappa})} + \sum_{n=1}^{M} \lambda_n \Delta x_n^2 \\ p(\Delta\boldsymbol{\kappa}) &= \frac{1}{Z_p} q(\Delta\boldsymbol{\kappa}) \exp\left(-\sum_{n=1}^{M} \lambda_n \Delta x_n^2\right) \quad (6)\end{aligned}$$

with Z_p being the normalization constant of the target distribution. Note that $\frac{\partial^2 \tilde{\mathcal{D}}_{\mathrm{KL}}(p|q)}{\partial p(\Delta\boldsymbol{\kappa})^2} = \frac{1}{p(\Delta\boldsymbol{\kappa})}$ is positive, therefore we know our solution will indeed be a minimum.

3.3. First order approximation for atom fluctuations

In order to use Eq. (6) to satisfy the imposed constraints, we need to express Δx^2 in terms of $\Delta\boldsymbol{\kappa}$. To first order, we can express the displacement vectors $\boldsymbol{\Delta x}_m^i$ of each atom with respect to the i^{th} internal coordinate as

$$\boldsymbol{\Delta x}_m^i = \frac{\partial \boldsymbol{x}_m}{\partial \kappa_i} \Delta\kappa_i \quad (7)$$

where $\boldsymbol{x}_m$ is the position of the m^{th} atom, under the condition that the atom is *post-rotational*, i.e. the location of atom m is downstream of the i^{th} internal coordinate. From Eq. (7) it follows that the squared distance can be defined as

$$\Delta x_m^2 = \sum_{ij} \frac{\partial \boldsymbol{x}_m}{\partial \kappa_i}\Delta\kappa_i \cdot \frac{\partial \boldsymbol{x}_m}{\partial \kappa_j}\Delta\kappa_j = \Delta\boldsymbol{\kappa}^{\mathrm{T}} \boldsymbol{G}_m \Delta\boldsymbol{\kappa} \quad (8)$$

where $\boldsymbol{G}_m^{i,j} = \frac{\partial \boldsymbol{x}_m}{\partial \kappa_i} \cdot \frac{\partial \boldsymbol{x}_m}{\partial \kappa_j}$ is a symmetric and positive-definite matrix.

Substituting Eq. (8) and our prior expression from Eq. (1) into our target distribution from Eq. (6) gives a new Gaussian distribution:

$$\begin{aligned}p(\Delta\boldsymbol{\kappa}) &= \frac{1}{\tilde{Z}} \exp\left(-\frac{1}{2}\Delta\boldsymbol{\kappa}^{\mathrm{T}} \left(\boldsymbol{\Sigma}_{\kappa,\mathrm{prior}}^{-1} + \boldsymbol{\Sigma}_{\kappa,\mathrm{constr}}^{-1}\right)\Delta\boldsymbol{\kappa}\right) \\ &= \mathcal{N}(0, \tilde{\boldsymbol{\Sigma}}) \quad (9)\end{aligned}$$

where $\tilde{Z}$ is the new normalization constant, $\boldsymbol{\Sigma}_{\kappa,\mathrm{constr}}^{-1} = 2\sum_{n=1}^{M} \lambda_n \boldsymbol{G}_n$ and the covariance matrix of the new Gaussian distribution $\tilde{\boldsymbol{\Sigma}} = \left(\boldsymbol{\Sigma}_{\kappa,\mathrm{prior}}^{-1} + \boldsymbol{\Sigma}_{\kappa,\mathrm{constr}}^{-1}\right)^{-1}$.

3.4. Satisfying the constraints

The final step in the constrained optimization is to establish the values for the Lagrange multipliers. A closed form solution for this is not readily available, but we can rewrite the constraints as

$$C_m = \mathbb{E}_{\Delta\boldsymbol{\kappa}\sim\mathcal{N}(0,\tilde{\boldsymbol{\Sigma}})}\left[\Delta\boldsymbol{\kappa}^{\mathrm{T}} \boldsymbol{G}_m \Delta\boldsymbol{\kappa}\right] = \mathrm{tr}(\tilde{\boldsymbol{\Sigma}}\boldsymbol{G}_{\mathrm{m}}) \quad (10)$$

Although it is nontrivial to express Lagrange multipliers λ in terms of atom fluctuations C, we thus see that it is possible to evaluate C given a set of Lagrange multipliers λ. In the following, we will therefore construct our models such that our networks predict λ, directly.

3.5. VAE pipeline

VAE model architecture. To demonstrate how our method works within a modelling context, we choose a one-layer variational autoencoder (VAE), for which the architecture is shown in Fig. 3. The VAE has a simple linear encoder that takes internal coordinates $\boldsymbol{\kappa}$ (dihedrals and bond angles, bond lengths are kept fixed) as input and maps to latent space $\boldsymbol{z}$, where we have a standard Gaussian as a prior on the latent space. The decoder outputs the mean over $\boldsymbol{\kappa}$, which is converted into Cartesian coordinates using pNeRF (AlQuraishi, 2018). This mean structure in 3D coordinates is used for two purposes. First, using the structure we can evaluate the partial derivatives of atom positions with respect to the individual κ, and thereby construct a $\boldsymbol{G}_m$ matrix for all M atoms. Second, the predicted mean over $\boldsymbol{\kappa}$ is used to get a pairwise distance matrix $\boldsymbol{d}$ that serves as the input to a U-Net (Ronneberger et al., 2015), from which we estimate values for the Lagrange multipliers for each constraint. This allowed the variational autoencoder, conditioned on the latent state z, to modulate the allowed fluctuations. Implementation-wise, the U-net is concluded with an average pooling operation that for each row-column

combination computes one Lagrange multiplier λ. Together with our fixed-variance prior over $\boldsymbol{\kappa}$ and hyper parameter a determining the strength of this prior, a new precision matrix is formed according to Eq. (9). The model can generate new structures through simple ancestral sampling: first generating $\boldsymbol{z}$ from the standard normal prior, and subsequently sampling from a multivariate Gaussian distribution with the decoded mean and the constructed precision matrix. For specific model settings see Appendix A.

Loss. We customarily optimize the evidence lower bound (ELBO) using the Gaussian likelihood on the internal degrees of freedom as constructed above. This likelihood does not ensure that the predicted Lagrange multipliers are within the range within which our first order approximation of the fluctuations is valid. To ensure this, we add an auxiliary loss in the form of a mean absolute error over λ^{-1}, which prevents the $\boldsymbol{\kappa}$-prior from dominating. By tuning the weight w_{aux} on the auxiliary loss, we can influence the strength of the constraints.

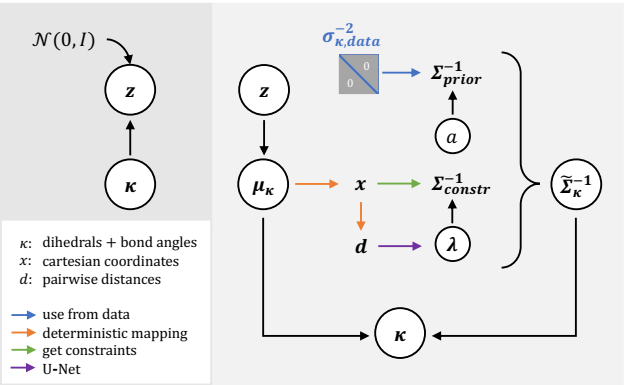


Figure 3. Model overview. The encoder (left) embeds internal coordinates into the latent space. The decoder predicts a mean, from which constraints are extracted that are weighed by predicted Lagrange multipliers to obtain a precision matrix. Together with the prior over the precision matrix based on the input data, a new precision matrix is formed which can be used to sample from a multivariate Gaussian.

4. Experiments

4.1. Test cases

1unc: solution nmr dataset. 1unc corresponds to the solution structure of the human villin C-terminal headpiece subdomain. This protein contains 36 residues, corresponding to 108 backbone (N, C_α and C) atoms. This solution nuclear magnetic resonance (nmr) dataset is freely available from the Protein Data Bank and contains 25 conformers.

1pga: molecular dynamics dataset. We have a short in-house molecular dynamics (MD) simulation for 1pga, corresponding to B1 immunoglobulin-binding domain protein G. This is a 56 amino acid long protein with 168 backbone atoms. The simulated trajectory is 20ns long, where structures are saved with a 50ps interval, resulting in 400 structures for this protein. See Appendix A for more details about the simulation.

4.2. Metrics

Local. In order to get a good overview of the local structure of protein, we visualize the distributions over internal coordinates. For bond angles, we present simple histograms, where we name the bond angles around the different backbone atoms θ_N, θ_{C_α} and θ_C, respectively. For the dihedrals, we show a Ramachandran plot. These are a well-known visualization tool in the context of protein structures, where ϕ and ψ dihedrals, which are the torsional angles around the $N - C_\alpha$ and $C_\alpha - C$ bonds, are plotted against each other. Different types of secondary structure (α helices and β sheets) cluster together in different areas within this plot.

Global. In this paper, we are mostly interested in preserving constraints in Euclidean space, even though we are modelling our distribution in internal coordinate space. The constraints that we formulate in Section 3 act on atom fluctuations in Cartesian coordinates. We accordingly report the variance over atom positions across superposed samples to evaluate global structure fluctuations. Additionally, we can visualize how well our first-order approximation is holding by calculating C_m according to Eq. (10) given the predictions of the model. Since these constraints assume the same absolute starting point for all proteins, this metric can only be evaluated on structures directly sampled by the model without superposing them.

Quality. To assess the quality of sampled structures, we utilize the Qualitative Model Energy ANalysis (QMEAN) server to evaluate the QMEAN6 score, which is a combination of six different potentials, representing both local and global interactions (Studer et al., 2020; Benkert et al., 2011). Each sample is assigned a QMEAN6 score, and we calculate the mean over all samples.

Baselines. Apart from comparing the generated samples from our model to the ground truth distributions that come from MD or nmr, we also include two baselines. The first are samples from our prior over fluctuations in $\boldsymbol{\kappa}$, which corresponds to independently sampling each internal coordinate from a univariate Gaussian based on their individual variances as estimated from the dataset. The final baseline are samples from a multivariate Gaussian with a mean based on the dataset and a precision matrix computed by a standard estimator. We use an empirical estimator for 1pga, but since this method led to a non-invertible matrix for 1unc,

we resort to an Oracle Approximating Shrinkage (OAS) estimator (Chen et al., 2010) for this system.

4.3. Internal-coordinate density modelling results

1unc. Our first test case is the small solution nmr dataset 1unc, which is a mostly α-helical protein with a compact global structure (Fig. 4A). In the precision matrix our VAE predicts, as reported in Fig. 4B, one can detect block-type behavior along different secondary structure elements, for dihedrals as well as bond angles. Fig. 4C shows that the distributions over internal coordinates are similar to the reference nmr distribution, although the distributions are slightly more peaked. This is inherent to our method, since our first-order approximation holds for smaller steps. As shown in Fig. 4D (top), the atom fluctuations in the samples from the VAE mostly follow the same pattern as C_m, our imposed constraints. This demonstrates that our approximation is holding well, and even though samples sometimes deviate from C_m in one place, mostly around more flexible parts of the protein, the approximation can still be "recovered" further downstream. The atom fluctuations compared to the reference distribution (Fig. 4D, bottom) also conform to our hypothesis; overall the sample fluctuations are smaller compared to the reference, but they follow a similar pattern. In other words, our model takes a smaller step with the same characteristics. In contrast, both baselines have fluctuations that are much larger than the reference.

In terms of sample quality, Table 1 shows that samples from the VAE are on par with the reference structures. Samples from the standard estimator baseline do worse, but samples from the $\boldsymbol{\kappa}$-prior also do well. This is probably due to the fact that 1unc largely consists of α-helices with very few loops, and consequently good local structure can more easily lead to acceptable global structure. However, from Fig. 4D (bottom) it is clear that the atom fluctuations of the prior are not representative of the nmr distribution. This is further illustrated in Appendix B.1, where we show 3D representations for some of the sampled structures.

1pga. The second test case, 1pga, is a larger and more complex protein containing two β-strands, and α-helix and two more β-strands that form a β-sheet with the first two strands (Fig. 1A). The global fold is therefore very important in this protein, which should be well-reflected in the atom fluctuations. The precision matrix outputted by our VAE (Fig. 5B) shows strong blocks in both dihedrals and bond angles, corresponding to different secondary structure elements. Similar to 1unc, the bond angle and dihedral distributions shown in Fig. 5C are following the same pattern as the reference distributions, but less broadly distributed. Fig. 5D (top) shows that the atom fluctuations in VAE samples are following the imposed constraints C_m, and deviations from our first-order approximation mostly occur in loop regions, exemplified by the large peak between the first two β-strands. From Fig. 5D (bottom), it is again clear that samples from our model follow the pattern of a smaller-step reference distribution, while prior and standard estimator baseline samples have fluctuations that are much larger than the reference distribution.

That the fluctuations in the baselines are too large is also reflected in the sample quality as reported in Table 1. While VAE sample quality is comparable to the reference data quality, the prior and standard estimator samples are of lower quality. This demonstrates that for a globally more complex protein such as 1pga, a good distribution over internal coordinates does not guarantee good global structure fluctuations, which is compensated for in our model by the constraints we impose. This is also clearly visible when we visualise the samples in 3D in Appendix B.1, where baseline samples even show crossings of the backbone. Moreover, Appendix B.2 shows examples of how the prior weight a and the auxiliary loss weight w_{aux} influence atom fluctuations, demonstrating that tuning these hyperparameters is important for good performance.

5. Related work

There is a large body of work on models for analyzing trajectories of molecular dynamics simulations, either through Markov state models (Chodera & Noé, 2014; Singhal & Pande, 2005; Sarich et al., 2013; Schütte et al., 1999; Prinz et al., 2011), or more complex modelling strategies (Mardt et al., 2018; Hernández et al., 2018; Sultan et al., 2018; Mardt et al., 2020; Xie et al., 2019). Typically, these focused on dimensionality reduced representations of the molecular structures, and are therefore not density models from which samples can be drawn.

To our knowledge, the first generative density model of full protein coordinates was the Boltzmann generator (Noé et al., 2019), a normalizing flow over the Cartesian coordinates of protein ensembles. This approach was later used to estimate coarse-grained force fields for molecular dynamics simulations, which demonstrated the ability of flows to sample structural ensembles for small proteins (Köhler et al., 2022). Other approaches involve latent variable models. One example is the IG-VAE, which generates structures in 3D coordinates but expresses the loss in terms of distances and internal coordinates to maintain SE(3) invariance. Similar approaches have been used to analyze cryo-EM data, where the task is to generate ensembles of structures given the observed cryo-EM image data. Since cryo-EM data provides information at slightly lower resolution than the full-atomic detail we discuss here, the output of these approaches are often density maps in 3D space (Zhong et al., 2021; Punjani & Fleet, 2021). One example of atomic-level modelling in this space is (Rosenbaum et al., 2021), which decodes deter-

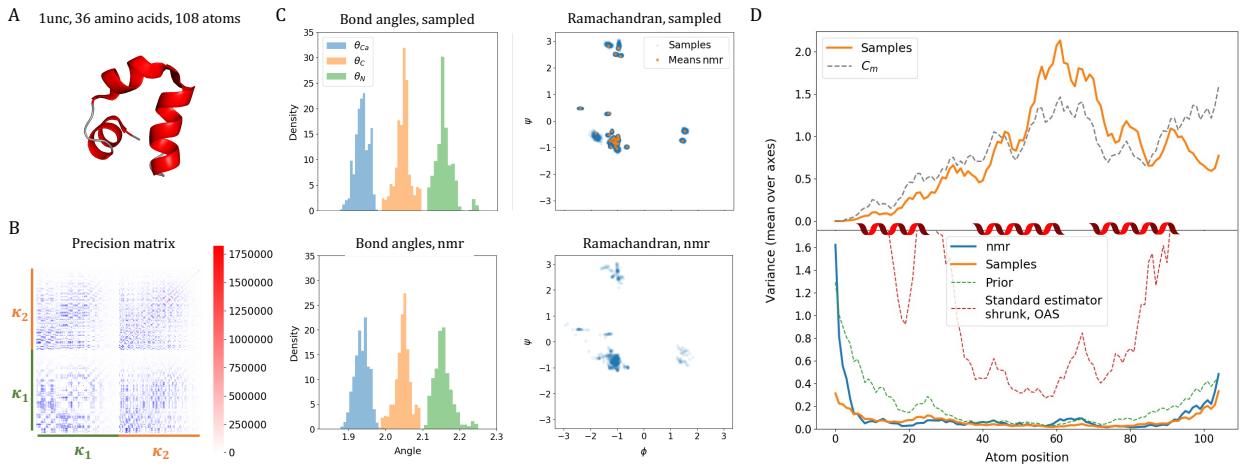


Figure 4. Internal-coordinate density modelling for 1unc. A: 3D structure of 1unc visualized using PyMOL (Schrödinger & DeLano), with α helices in red. B: precision matrix predicted by our VAE model, organized in blocks over dihedrals and bond angles, respectively. C: distributions over bond angles and dihedrals, for both our VAE (top, with nmr means indicated in orange) and the reference nmr distribution (bottom). D: atom fluctuations over atom positions for samples from our model compared to computed constraints for non-superposed structures (top) and atom fluctuations compared to baselines on superposed structures (bottom). The variance is calculated as the mean of the variances over the x, y and z axis, in Å^2. Secondary structure elements are indicated along the atom position axis.

Table 1. Sample quality using QMEAN server for samples from the VAE model and baselines: reference data, samples from the prior, and samples from a standard estimator. Based on 200 random samples, except the nmr reference which is based on 25 data points.

	Reference	**VAE samples**	**Prior samples**	**Standard estimator samples**
1unc	0.60 ± 0.05	0.60 ± 0.03	0.60 ± 0.03	0.50 ± 0.07 (empirical)
1pga	0.54 ± 0.03	0.57 ± 0.03	0.49 ± 0.06	0.41 ± 0.07 (OAS)

ministically into 3D coordinates, but describes the variance in image space (Rosenbaum et al., 2021). Finally, diffusion models have recently provided a promising new approach to density modelling, with impressive examples of density modelling at the scale of full-size proteins (Watson et al., 2022; Ingraham et al., 2022; Anand & Achim, 2022).

The primary objective in our paper is to investigate whether density modelling is feasible in internal coordinates. Internal-coordinate probabilistic models of proteins have traditionally focused on protein local structure, i.e. correct modelling of angular distributions of the secondary structure elements in proteins. Early work was based on hidden Markov models of small fragments (Camproux et al., 1999; 2004; de Brevern et al., 2000; Benros et al., 2006). The discrete nature of the fragments meant that these models did not constitute a complete probabilistic model of the protein structure. Later models solved this issue by modelling local structure in internal coordinates, using different sequential models and angular distributions (Edgoose et al., 1998; Bystroff et al., 2000; Hamelryck et al., 2006; Boomsma et al., 2008; 2014; Thygesen et al., 2021). Due to the downstream effects of small internal-coordinate fluctuations, these models are not by themselves capable of modelling the distribution of entire protein structures, but they are useful as proposal distributions in Markov chain Monte Carlo (MCMC) simulations of proteins (Irbäck & Mohanty, 2006; Boomsma et al., 2013). Using deep learning architectures to model the sequential dependencies in the protein chain, recent work has pushed the maximum length of fragments that can be reliably modelled (Thygesen et al., 2021). To our knowledge, however, no internal-coordinate density model has yet been able to robustly model angular densities beyond fragments of length 15, due to the challenges in estimating the necessary covariance structure.

Our work was inspired by methods used for constrained Gaussian updates in MCMC simulation, first introduced by (Favrin et al., 2001), and later extended by (Bottaro et al., 2012). Our method generalizes the approach to global updates of proteins, derives the relationship between the Lagrange multipliers and corresponding fluctuations in Euclidean space, and uses neural networks to govern the level of fluctuations in order to modulate the induced covariance structures.

Recent work has demonstrated that internal-coordinate modelling can also be done using diffusion models (Jing et al.,

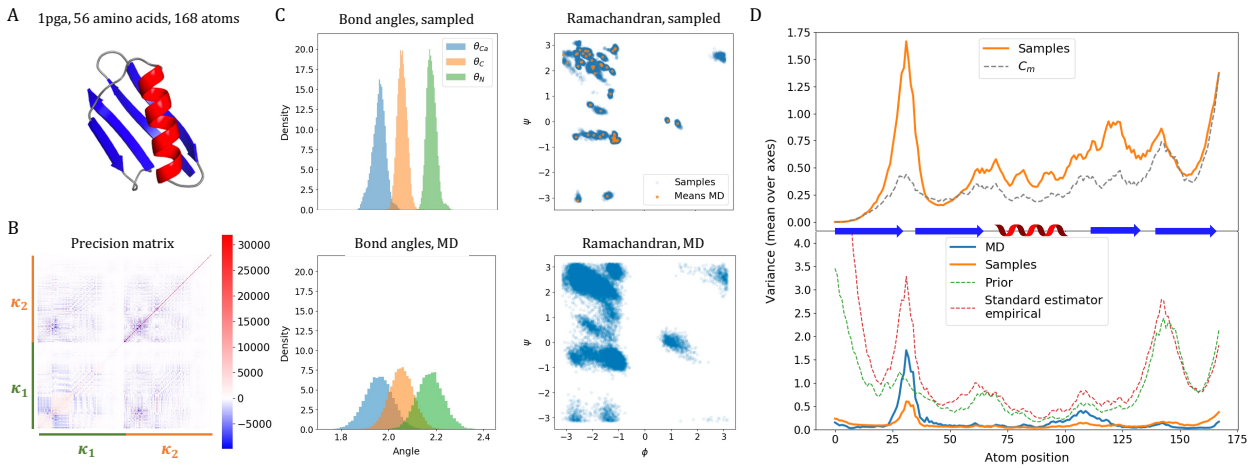


Figure 5. Internal-coordinate density modelling for 1pga. A: 3D structure of 1pga visualized using PyMOL (Schrödinger & DeLano), with α helices in red and β sheets in blue. B: precision matrix predicted by our VAE model, organized in blocks over dihedrals and bond angles, respectively. C: distributions over bond angles and dihedrals, for both our VAE (top, with nmr means indicated in orange) and the reference MD distribution (bottom). D: atom fluctuations over atom positions for samples from our model compared to computed constraints for non-superposed structures (top) and atom fluctuations compared to baselines on superposed structures (bottom). The variance is calculated as the mean of the variances over the x, y and z axis, in Å^2. Secondary structure elements are indicated along the atom position axis.

2022). So far this method has been demonstrated only on small molecules. We believe the method we introduce in this paper might help scale these diffusion approaches to full proteins.

6. Discussion

Although protein structure prediction is now considered a solved problem, fitting the density of structural ensembles remains an open problem. Many recent activities in the field focus on diffusion models in the Cartesian coordinate representation of a protein. In this paper, we take a different approach, and investigate how we can describe small-scale fluctuations in terms of a distribution over the internal degrees of freedom of a protein. The main challenge in this context is the complex covariance between different parts of the chain. Failing to model this properly results in models that produce disruptive changes to the global structure even for fairly minor fluctuations in the internal coordinates. Instead of estimating the covariance matrix from data, we show that it can be induced by imposing constraints on the Cartesian fluctuations. In a sense, this represents a natural compromise between internal and Cartesian coordinates: we obtain samples that are guaranteed to fulfill the physical constraints of the protein topology (e.g. bond lengths, and bond angles), while at the same time producing meaningful fluctuations globally.

We implement the idea in the decoder of a variational autoencoder on two protein systems. This is primarily a proof of concept, and this implementation has several limitations. First of all, the standard deviations of the individual degrees of freedom in the prior of the internal degrees of freedom are currently set as a hyperparameter. These could be estimated from data, either directly, or using a preexisting model of protein local structure. In the current implementation, we place an auxiliary loss on the inverse of the Lagrange multiplier, to ensure that the fluctuations stay within a range where our first-order approximation is valid. A more elegant implementation would be to use the relationship between the Lagrange multiplier and the fluctuations in Cartesian coordinates (10), such that the auxiliary loss could be expressed directly as a likelihood on the Cartesian fluctuations. We leave both these extensions for future work. Another limitation is the current model is that the produced fluctuations are generally too small to match the target densities. This can be solved by constructing a hierarchical VAE, where samples are constructed as a multi-step process, similar to the generation process in diffusion models. In fact, we believe that our fundamental approach of induced covariance matrices could be a fruitful way to make diffusion models in internal coordinates scale to larger systems, by allowing for larger non-disruptive steps.

7. Code and data availability

Code and data will be made available upon acceptance.

8. Acknowledgements

The work was supported by the Novo Nordisk Foundation (project grant nr NNF18OC0052719) and conducted within the Center for Basic Machine Learning Research in Life Science (MLLS, grant nr NNF20OC0062606).

References

AlQuraishi, M. pnerf: Parallelized conversion from internal to cartesian coordinates. *bioRxiv*, pp. 385450, 2018.

Anand, N. and Achim, T. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. *arXiv preprint arXiv:2205.15019*, 2022.

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

Benkert, P., Biasini, M., and Schwede, T. Toward the estimation of the absolute quality of individual protein structure models. *Bioinformatics*, 27(3):343–350, 2011.

Benros, C., de Brevern, A., Etchebest, C., and Hazout, S. Assessing a novel approach for predicting local 3D protein structures from sequence. *Proteins*, 62:865–880, 2006.

Boomsma, W., Mardia, K., Taylor, C., Ferkinghoff-Borg, J., Krogh, A., and Hamelryck, T. A generative, probabilistic model of local protein structure. *Proc Natl Acad Sci USA*, 105(26):8932–8937, 2008.

Boomsma, W., Frellsen, J., Harder, T., Bottaro, S., Johansson, K. E., Tian, P., Stovgaard, K., Andreetta, C., Olsson, S., Valentin, J. B., et al. Phaistos: A framework for markov chain monte carlo simulation and inference of protein structure. *Journal of computational chemistry*, 34 (19):1697–1705, 2013.

Boomsma, W., Tian, P., Frellsen, J., Ferkinghoff-Borg, J., Hamelryck, T., Lindorff-Larsen, K., and Vendruscolo, M. Equilibrium simulations of proteins using molecular fragment replacement and nmr chemical shifts. *Proceedings of the National Academy of Sciences*, 111(38):13852–13857, 2014.

Bottaro, S., Boomsma, W., E. Johansson, K., Andreetta, C., Hamelryck, T., and Ferkinghoff-Borg, J. Subtle monte carlo updates in dense molecular systems. *Journal of Chemical Theory and Computation*, 8(2):695–702, 2012.

Bystroff, C., Thorsson, V., and Baker, D. HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. *J Mol Biol*, 301(1):173–190, 2000.

Camproux, A., Tuffery, P., Chevrolat, J., Boisvieux, J., and Hazout, S. Hidden Markov model approach for identifying the modular framework of the protein backbone. *Protein Eng Des Sel*, 12(12):1063–1073, 1999.

Camproux, A., Gautier, R., and Tufféry, P. A hidden Markov model derived structural alphabet for proteins. *J Mol Biol*, 339(3):591–605, 2004.

Chen, Y., Wiesel, A., Eldar, Y. C., and Hero, A. O. Shrinkage algorithms for mmse covariance estimation. *IEEE transactions on signal processing*, 58(10):5016–5029, 2010.

Chodera, J. D. and Noé, F. Markov state models of biomolecular conformational dynamics. *Current opinion in structural biology*, 25:135–144, 2014.

de Brevern, A., Etchebest, C., and Hazout, S. Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks. *Proteins*, 41(3):271–287, 2000.

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simmonett, A. C., Harrigan, M. P., Stern, C. D., et al. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, 13(7): e1005659, 2017.

Edgoose, T., Allison, L., and Dowe, D. An MML classification of protein structure that knows about angles and sequence. *Pac Symp Biocomput*, 3:585–596, 1998.

Favrin, G., Irbäck, A., and Sjunnesson, F. Monte Carlo update for chain molecules: Biased Gaussian steps in torsional space. *J. Chem. Phys.*, 114:8154–8158, 2001.

Hamelryck, T., Kent, J., and Krogh, A. Sampling realistic protein conformations using local structural bias. *PLoS Comput Biol*, 2(9):e131, 2006.

Hernández, C. X., Wayment-Steele, H. K., Sultan, M. M., Husic, B. E., and Pande, V. S. Variational encoding of complex dynamics. *Physical Review E*, 97(6):062412, 2018.

Ingraham, J., Baranov, M., Costello, Z., Frappier, V., Ismail, A., Tie, S., Wang, W., Xue, V., Obermeyer, F., Beam, A., et al. Illuminating protein space with a programmable generative model. *bioRxiv*, pp. 2022–12, 2022.

Irbäck, A. and Mohanty, S. Profasi: a monte carlo simulation package for protein folding and aggregation. *J. Comput. Chem.*, 27(13):1548–1555, 2006.

Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. Torsional diffusion for molecular conformer generation. *arXiv preprint arXiv:2206.01729*, 2022.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Köhler, J., Chen, Y., Krämer, A., Clementi, C., and Noé, F. Force-matching coarse-graining without forces. *arXiv preprint arXiv:2203.11167*, 2022.

Maier, J. A., Martinez, C., Kasavajhala, K., Wickstrom, L., Hauser, K. E., and Simmerling, C. ff14sb: improving the accuracy of protein side chain and backbone parameters from ff99sb. *Journal of chemical theory and computation*, 11(8):3696–3713, 2015.

Mardt, A., Pasquali, L., Wu, H., and Noé, F. Vampnets for deep learning of molecular kinetics. *Nature communications*, 9(1):1–11, 2018.

Mardt, A., Pasquali, L., Noé, F., and Wu, H. Deep learning markov and koopman models with physical constraints. In *Mathematical and Scientific Machine Learning*, pp. 451–475. PMLR, 2020.

Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Prinz, J.-H., Wu, H., Sarich, M., Keller, B., Senne, M., Held, M., Chodera, J. D., Schütte, C., and Noé, F. Markov models of molecular kinetics: Generation and validation. *The Journal of chemical physics*, 134(17):174105, 2011.

Punjani, A. and Fleet, D. J. 3d flexible refinement: structure and motion of flexible proteins from cryo-em. *BioRxiv*, 2021.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Rosenbaum, D., Garnelo, M., Zielinski, M., Beattie, C., Clancy, E., Huber, A., Kohli, P., Senior, A. W., Jumper, J., Doersch, C., et al. Inferring a continuous distribution of atom coordinates from cryo-em images using vaes. *arXiv preprint arXiv:2106.14108*, 2021.

Sarich, M., Banisch, R., Hartmann, C., and Schütte, C. Markov state models for rare events in molecular dynamics. *Entropy*, 16(1):258–286, 2013.

Schrödinger, L. and DeLano, W. Pymol. URL http://www.pymol.org/pymol.

Schütte, C., Fischer, A., Huisinga, W., and Deuflhard, P. A direct approach to conformational dynamics based on hybrid monte carlo. *Journal of Computational Physics*, 151(1):146–168, 1999.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W., Bridgland, A., et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792): 706–710, 2020.

Singhal, N. and Pande, V. S. Error analysis and efficient sampling in markovian state models for molecular dynamics. *The Journal of chemical physics*, 123(20):204909, 2005.

Studer, G., Rempfer, C., Waterhouse, A. M., Gumienny, R., Haas, J., and Schwede, T. Qmeandisco—distance constraints applied on model quality estimation. *Bioinformatics*, 36(6):1765–1771, 2020.

Sultan, M. M., Wayment-Steele, H. K., and Pande, V. S. Transferable neural networks for enhanced sampling of protein dynamics. *Journal of chemical theory and computation*, 14(4):1887–1894, 2018.

Thygesen, C. B., Steenmans, C. S., Al-Sibahi, A. S., Moreta, L. S., Sørensen, A. B., and Hamelryck, T. Efficient generative modelling of protein structure fragments using a deep markov model. In *International Conference on Machine Learning*, pp. 10258–10267. PMLR, 2021.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *bioRxiv*, pp. 2022–12, 2022.

Wu, K. E., Yang, K. K., Berg, R. v. d., Zou, J. Y., Lu, A. X., and Amini, A. P. Protein structure generation via folding diffusion. *arXiv preprint arXiv:2209.15611*, 2022a.

Wu, R., Ding, F., Wang, R., Shen, R., Zhang, X., Luo, S., Su, C., Wu, Z., Xie, Q., Berger, B., et al. High-resolution de novo structure prediction from primary sequence. *BioRxiv*, pp. 2022–07, 2022b.

Xie, T., France-Lanord, A., Wang, Y., Shao-Horn, Y., and Grossman, J. C. Graph dynamical networks for unsupervised learning of atomic scale dynamics in materials. *Nature communications*, 10(1):1–9, 2019.

Zhong, E. D., Bepler, T., Berger, B., and Davis, J. H. Cryodrgn: reconstruction of heterogeneous cryo-em structures using neural networks. *Nature methods*, 18(2):176–185, 2021.

A. Experiment details

Model. The VAE encoder linear layer sizes are $[60, 30, 15]$ and decoder linear layer sizes are $[15, 30, 60]$. We use a standard U-Net that takes a pairwise distance matrix of size $M \times M$ (where M is the number of atoms) and 1 channel, and scales the channels up to 1024 channels in four steps before scaling back down to one channel in four steps. All models were run for 1200 epochs, with batch size 16 and learning rate 0.01. All datasets were split 90%-10% into a training and validation set. The validation set is tracked during training to prevent overtraining. Prior and auxiliary loss weights were explored with grid search, values chosen for the models reported in the main paper are shown in Table A1. All final metrics are calculated on structures sampled from the model. At sampling time, 100 samples are drawn from the latent space prior, decoded, and subsequently 4 samples are drawn from the final multivariate Gaussian to get a total of 400 structures. Models were trained on a Nvidia Titan Xp (12GB) GPU.

Table A1. Weight settings for test cases.

	$\mathbf{a}$	$\mathbf{w}_{\text{aux}}$
1unc	10	5
1pga	50	50

Molecular dynamics details. The molecular dynamics simulation was done in OpenMM (Eastman et al., 2017), using an Amber forcefield (Maier et al., 2015), water type TIP3P, box geometry "rhombic dodecahedron" and a padding of 1 nm on each side of the solvated protein (i.e. 2 nm in total). The simulation is 20ns in total with a 50ps time lag, giving 400 structures.

B. Additional results

B.1. Visualization of sampled structures.

Fig. A1 shows ten randomly chosen superposed samples for our model, the reference, and the prior and standard estimator baselines. This demonstrates that VAE samples tend to have slightly smaller, but globally consistent fluctuations compared to the reference data, while the baselines show larger fluctuations that can lead to unphysical structures containing crossings.

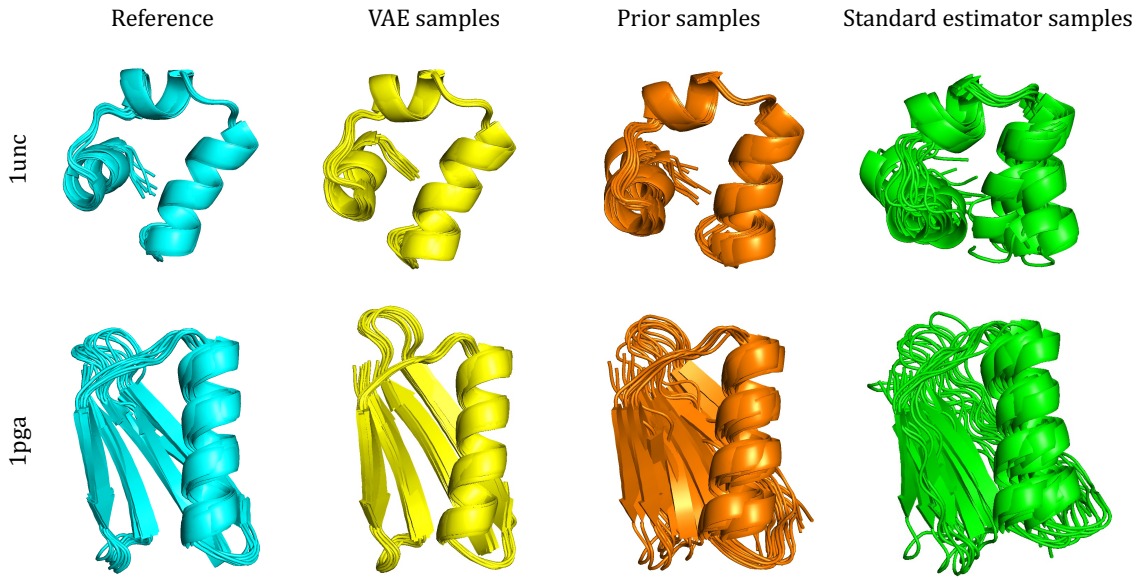


Figure A1. Visualization of ten random, superposed samples from the reference data, the VAE model and the prior and standard estimator baselines, for both 1unc and 1pga.

B.2. Impact of weights.

The prior weight a and the weight on the auxiliary loss on the lagrange multipliers w_{aux} are hyperparameters to the model, and choosing different values can have a lot of impact. As an example, Fig. A2 shows how different choices for 1pga affect how well samples conform to C_m (i.e. how well our approximation holds) and how the resulting fluctuations compare to baselines.

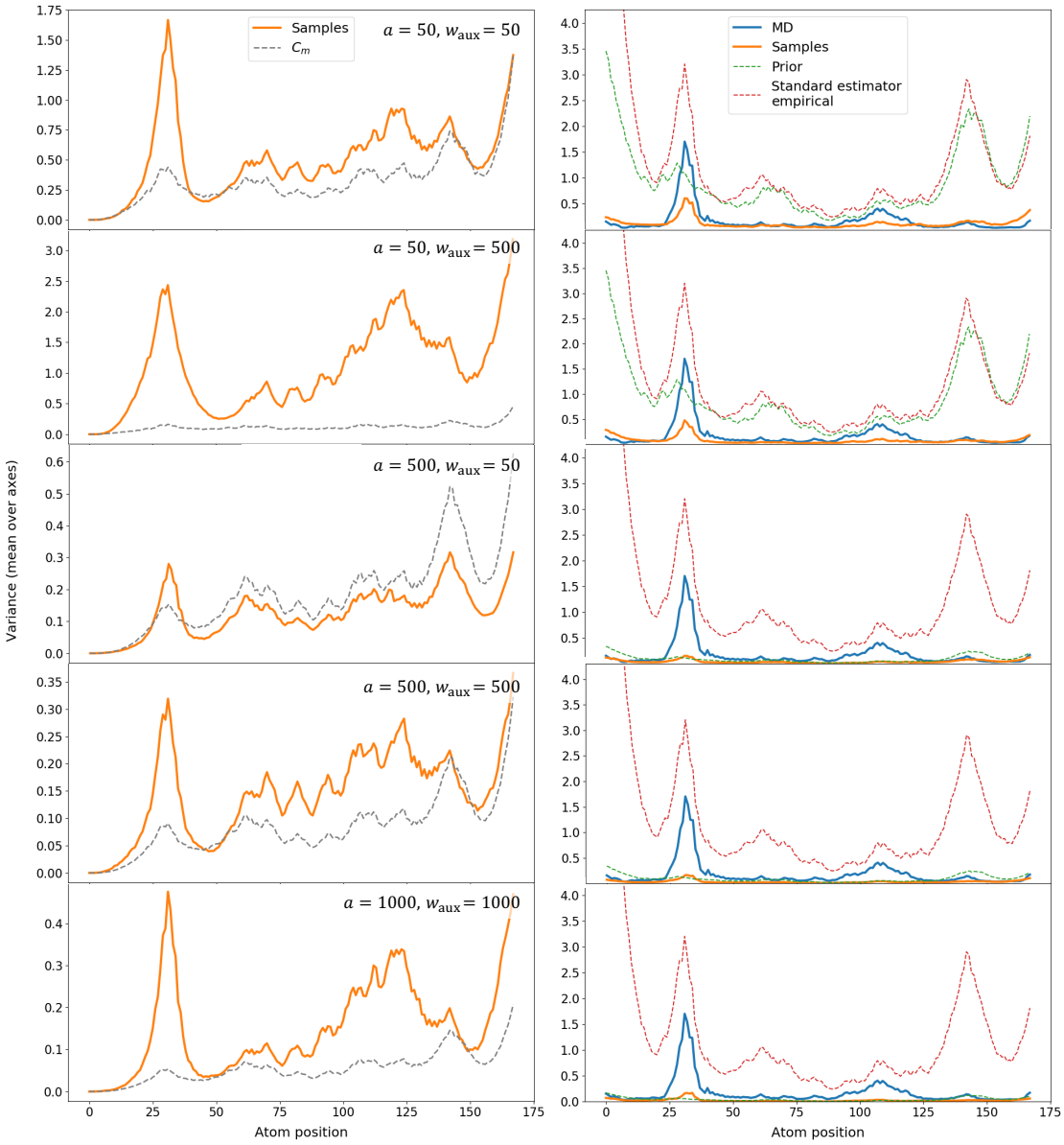


Figure A2. The choice of a and the auxiliary loss weight w_{aux} impact atomic fluctuations. Here we show five different combinations for 1pga. Left: atom fluctuations of model samples compared to imposed constraints, right: atom fluctuations of superimposed structures for model samples and baselines. The variance is calculated as the mean of the variances over the x, y and z axis, in Å^2.

Chapter 4

Two for One: Diffusion Models and Force Fields for Coarse-Grained Molecular Dynamics

Marloes Arts*, Victor Garcia Satorras*, Chin-Wei Huang, Daniel Zuegner, Marco Federici, Cecilia Clementi, Frank Noé, Robert Pinsler, Rianne van den Berg

The work presented in this chapter was done during an internship at Microsoft Research in Amsterdam (AI4Science). The manuscript was submitted to ICML 2023 and is currently under review. A preprint is available on arXiv: `https://arxiv.org/abs/2302.00600`.

*Equal contribution

Two for One: Diffusion Models and Force Fields for Coarse-Grained Molecular Dynamics

Marloes Arts [*1] **Victor Garcia Satorras** [*2] **Chin-Wei Huang** [2] **Daniel Zuegner** [2] **Marco Federici** [1]
Cecilia Clementi [3 2] **Frank Noé** [2] **Robert Pinsler** [2] **Rianne van den Berg** [2]

Abstract

Coarse-grained (CG) molecular dynamics enables the study of biological processes at temporal and spatial scales that would be intractable at an atomistic resolution. However, accurately learning a CG force field remains a challenge. In this work, we leverage connections between score-based generative models, force fields and molecular dynamics to learn a CG force field without requiring any force inputs during training. Specifically, we train a diffusion generative model on protein structures from molecular dynamics simulations, and we show that its score function approximates a force field that can directly be used to simulate CG molecular dynamics. While having a vastly simplified training setup compared to previous work, we demonstrate that our approach leads to improved performance across several small- to medium-sized protein simulations, reproducing the CG equilibrium distribution, and preserving dynamics of all-atom simulations such as protein folding events.

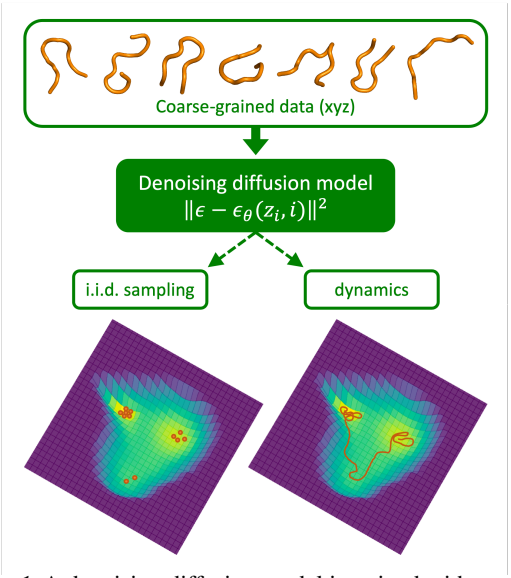


Figure 1. A denoising diffusion model is trained with a standard loss on atomistic (fine-grained) equilibrium samples projected onto the CG space. By leveraging connections between score-based generative modeling, force fields and molecular dynamics, we obtain a single model that can generate i.i.d. equilibrium CG samples *and* whose neural network can be used as a CG force field in CG molecular dynamics simulations.

1. Introduction

Coarse-grained (CG) molecular dynamics (MD) promises to scale simulations to larger spatial and time scales than currently accessible through atomistic MD simulations (Clementi, 2008; Noid, 2013; Saunders & Voth, 2013; Kmiecik et al., 2016). Scaling up MD by orders of magnitude would enable new studies on macromolecular dynamics over longer ranges of time, such as large protein folding events and slow interactions between large molecules.

To obtain a CG simulation model, one first maps the all-atom, or *fine-grained*, representation to a *coarse-grained* representation, e.g. by grouping certain atoms together to form so-called CG beads. Second, a CG force field needs to be designed such that CG molecular dynamics simulations reproduce relevant features of molecular systems.

In top-down approaches, a CG model is often defined to reproduce specific macroscopic observables, as experimentally measured and/or simulated on fine-grained models (Marrink et al., 2007; Davtyan et al., 2012; Matysiak & Clementi, 2006; Chen et al., 2018). In bottom-up approaches, one seeks to obtain a CG model reproducing the microscopic behavior (e.g., thermodynamics, kinetics) of a fine-grained model (Noid et al., 2008; Shell, 2008; Nüske et al., 2019). In the latter case, a common approach is to define a CG force field for the chosen CG representation by enforcing thermodynamic consistency (Noid, 2013). This requires that simulations following the CG model should have the same equilibrium distribution as obtained by projecting

[*]Equal contribution. [1]Work done during an internship at Microsoft Research. [2]AI4Science, Microsoft Research. [3]Freie Universität Berlin, Department of Physics. Correspondence to: Marloes Arts <ma@di.ku.dk>, Victor Garcia Satorras <victorgar@microsoft.com>.

Preprint.

equilibrated all-atom simulations onto the CG resolution.

Traditional bottom-up coarse-graining techniques that rely on the thermodynamic consistency principle have produced significant results in the last decade (Mim et al., 2012; Chu & Voth, 2005; Yu et al., 2021), in particular when used in combination with machine-learning methods (Wang et al., 2019; Husic et al., 2020). Two commonly used approaches are variational force matching and relative entropy minimization.

Variational force matching minimizes the mean squared error between the model's CG forces and the atomistic forces projected onto the CG space, which must be included in the data (Noid et al., 2008). However, due to the stochastic nature of the projected forces, this noisy force-matching estimator has a large variance, leading to data-inefficient training. Alternatively, relative entropy minimization approaches (Shell, 2008) perform density estimation in the CG space without accessing atomistic forces. The majority of this class of methods are equivalent to energy-based models (Song & Kingma, 2021). Since training these models requires iteratively drawing samples from the model to estimate log-likelihood gradients, such methods demand significantly higher computational cost (Hinton, 2002).

Flow-matching (Köhler et al., 2023) is a hybrid approach that does not require atomistic forces for training (like relative entropy minimization) while also retaining good sample-efficiency. The method has two training stages. First, a CG density is modeled with an augmented normalizing flow (Rezende & Mohamed, 2015; Papamakarios et al., 2021; Huang et al., 2020; Chen et al., 2020). A second learning stage with a force-matching-like objective is then required to extract a deterministic CG force field that can be used in CG molecular dynamics simulations. Köhler et al. (2023) demonstrated that flow-matching improves performance on several fast-folding proteins (Lindorff-Larsen et al., 2011). However, the learned CG models are not yet accurate enough for reproducing the thermodynamics of the corresponding fine-grained models, and scaling to larger proteins leads to instabilities.

In this work, we leverage the recently popularized class of denoising diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015), which have already shown promising results for protein structure generation (Wu et al., 2022; Trippe et al., 2022; Watson et al., 2022; Igashov et al., 2022; Qiao et al., 2022), conformer generation (Jing et al., 2022), and docking (Corso et al., 2022). In particular, we train a score-based generative model on CG structures sampled from the CG equilibrium distribution. By highlighting connections between score-based generative models (Song et al., 2020), force fields and molecular dynamics, we demonstrate that learning such a generative model with a standard denoising loss and a conservative score yields a single model that can be used to produce i.i.d. CG samples *and* which can be used directly as a CG force field for CG molecular dynamics simulations. An overview is shown in Figure 1. In addition to having a single-stage training setup, our method leads to improved performance across several small- to medium-sized protein simulations, reproducing the CG equilibrium distribution and preserving the dynamical mechanisms observed in all-atom simulations such as protein folding events. We also provide evidence that our diffusion CG model allows for scaling to a larger protein than previously accessible through flow-matching.

2. Background

Coarse-graining can be described by a dimensionality reduction map $\Xi : \mathbb{R}^{3N} \to \mathbb{R}^{3n}$ that transforms a high-dimensional atomistic representation $\mathbf{x} \in \mathbb{R}^{3N}$ to a lower-dimensional CG representation $\mathbf{z} \in \mathbb{R}^{3n}$, where $n \ll N$. For molecular systems, the CG map is usually linear, $\Xi \in \mathbb{R}^{3n \times 3N}$, and returns the Cartesian coordinates $\mathbf{z}$ of CG "beads" as a linear combination of the Cartesian coordinates $\mathbf{x}$ of a set of representative atoms.

The probability density of the atomistic system at a particular temperature $\mathcal{T}$ is described by the Boltzmann distribution $q(\mathbf{x}) \propto \exp\left(-U(\mathbf{x})/k_B\mathcal{T}\right)$, where $U(\mathbf{x})$ is the system's potential energy and k_B is the Boltzmann constant. By identifying the ensemble of atomistic configurations $\mathbf{x}$ that maps into the same CG configuration $\mathbf{z}$, we can explicitly express the probability density of the CG configurations $\mathbf{z}$ as:

$$q(\mathbf{z}) = \frac{\int \exp(-U(\mathbf{x})/k_B\mathcal{T})\delta(\Xi(\mathbf{x}) - \mathbf{z})\, d\mathbf{x}}{\int \exp(-U(\mathbf{x}')/k_B\mathcal{T}) d\,\mathbf{x}'}, \quad (1)$$

where $\delta(\cdot)$ is the Dirac delta function. Up to an additive constant, this distribution uniquely defines the thermodynamically consistent effective CG potential of mean force $V(\mathbf{z})$ (Noid et al., 2008):

$$\begin{aligned} V(\mathbf{z}) &= -k_B\mathcal{T}\log q(\mathbf{z}) + \text{cst.} = \\ &= -k_B\mathcal{T}\log \int e^{-U(\mathbf{x})/k_B\mathcal{T}}\delta(\Xi(\mathbf{x}) - \mathbf{z})\, d\mathbf{x} + \text{cst.} \end{aligned}$$

Unfortunately, computing the integral is usually intractable. Therefore, methods that approximate thermodynamically consistent effective CG potentials have been proposed. Below we briefly summarize two commonly used approaches.

Variational force matching. Noid et al. (2008) showed that under certain constraints of the coarse-graining mapping Ξ, a more tractable consistency equation between the coarse-grained force field $-\nabla_{\mathbf{z}}V(\mathbf{z})$ and the atomistic force field $-\nabla_{\mathbf{x}}U(\mathbf{x})$ can be obtained. More specifically, if Ξ is a linear map, and if each bead has at least

one atom with a nonzero coefficient only for that specific bead, then the following relation holds: $-\nabla_{\mathbf{z}} V(\mathbf{z}) = \mathbb{E}_{q(\mathbf{x}|\mathbf{z})}[\Xi_f(-\nabla_{\mathbf{x}} U(\mathbf{x}))]$. Here, Ξ_f is a linear map whose coefficients are related to the linear coefficients of the CG map Ξ (Ciccotti et al., 2005). Noid et al. (2008) showed that the above relation can be used to approximate a thermodynamically consistent CG potential $V_\theta(\mathbf{z})$ with parameters θ by minimizing the following variational loss:

$$\mathbb{E}_{q(\mathbf{x},\mathbf{z})}\left[\|\nabla_{\mathbf{z}} V_\theta(\mathbf{z}) - \Xi_f(\nabla_{\mathbf{x}} U(\mathbf{x}))\|_2^2\right]. \tag{2}$$

Relative entropy minimization. Another approach to obtaining the CG forces is via relative entropy minimization, where optimizing the density implicitly leads to optimized mean potential functions. Concretely, we seek to estimate the CG density by minimizing the relative entropy, or Kullback-Leibler divergence, $\mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z})]$, which is equivalent to optimizing the maximum likelihood when a finite number of samples is drawn from $q(\mathbf{z})$. The approximate CG forces can be extracted from the optimized model density $p_\theta(\mathbf{z})$ through $-\nabla_{\mathbf{z}} V_\theta(\mathbf{z}) \propto \nabla_{\mathbf{z}} \log p_\theta(\mathbf{z})$. Unlike variational force matching, relative entropy minimization does not impose any constraints on the CG map, and no atomistic forces are required for training.

Traditionally, an unnormalized version of p_θ is modeled by directly parameterizing the CG potential V_θ, yielding $p_\theta(\mathbf{z}) \propto \exp\left(-V_\theta(\mathbf{z})/k_B\mathcal{T}\right)$. To minimize the relative entropy, one would need to either estimate the free energy (*i.e.* the normalizing constant) of the model (Shell, 2008) or draw i.i.d. samples from the model for gradient estimation (Hinton, 2002; Thaler et al., 2022), which renders this approach impractical for higher-dimensional problems.

An alternative is to use an explicit density model such as a normalizing flow (Dinh et al., 2014; Rezende & Mohamed, 2015; Dinh et al., 2016), allowing for straightforward maximum-likelihood density estimation and force field learning. However, learning expressive invertible functions is challenging, so instead, Köhler et al. (2023) opted for augmented normalizing flows (Huang et al., 2020; Chen et al., 2020). The introduction of auxiliary random variables increases the expressivity of the flow, at the cost of an intractable marginal likelihood, yielding a minimization objective that is a variational upper bound to the relative entropy. Furthermore, one can only extract a stochastic estimate for the CG force from the augmented normalizing flow model. In order to distill a deterministic approximate CG force to simulate the CG dynamics, Köhler et al. (2023) proposed a teacher-student setup akin to variational force-matching. This two-stage approach was dubbed flow-matching.

3. Diffusion Models for Coarse-Grained Molecular Dynamics

Denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020; Sohl-Dickstein et al., 2015) sample from a probability distribution by approximating the inverse of a diffusion process, *i.e.* a denoising process. The diffusion (forward) process is defined as a Markov chain of L steps $q(\mathbf{z}_{1:L} \mid \mathbf{z}_0) = \prod_{i=1}^{L} q(\mathbf{z}_i \mid \mathbf{z}_{i-1})$, where $\mathbf{z}_0$ is a sample from the unknown data distribution $q(\mathbf{z}_0)$. The learned reverse process is defined as a reverse-time Markov chain of L denoising steps $p(\mathbf{z}_{0:L}) := p(\mathbf{z}_L)\prod_{i=1}^{L} p_\theta(\mathbf{z}_{i-1} \mid \mathbf{z}_i)$ that starts from the prior $p(\mathbf{z}_L)$. For real-valued random variables, the choice of distribution for the forward process is typically Gaussian, $q(\mathbf{z}_i \mid \mathbf{z}_{i-1}) = \mathcal{N}(\mathbf{z}_i; \sqrt{1-\beta_i}\mathbf{z}_{i-1}, \beta_i \boldsymbol{I})$, with $\{\beta_i\}$ fixed variance parameters that increase as a function of i such that the Markov chain has a standard Normal stationary distribution. The reverse process distributions are chosen to have the same functional form: $p(\mathbf{z}_L) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $p_\theta(\mathbf{z}_{i-1} \mid \mathbf{z}_i) = \mathcal{N}(\mathbf{z}_{i-1}; \mu_\theta(\mathbf{z}_i, i), \sigma_i^2 \boldsymbol{I})$. Here, $\mu_\theta(\mathbf{z}_i, i)$ is a learnable function with parameters θ, and σ_i^2 is a fixed variance. By making use of closed-form marginalization for Gaussian distributions, and by parameterizing the means as $\mu_\theta(\mathbf{z}_i, i) = \frac{1}{\sqrt{\alpha_i}}\left(\mathbf{z}_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}}\epsilon_\theta(\mathbf{z}_i, i)\right)$, with $\epsilon_\theta(\mathbf{z}_i, i)$ the noise prediction neural network, training proceeds by minimizing the loss (Ho et al., 2020):

$$\sum_{i=1}^{L} K_i \mathbb{E}_{q(\mathbf{z}_0)}\mathbb{E}_{\mathcal{N}(\epsilon;\mathbf{0},\mathbf{I})}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_i}\epsilon, i)\right\|^2\right]. \tag{3}$$

Here, $\alpha_i = 1 - \beta_i$ and $\bar{\alpha}_i = \prod_{s=1}^{i} \alpha_s$. Up to a constant, Eq. 3 is a negative evidence lower bound if $K_i = \frac{\beta_i^2}{2\sigma_i^2 \alpha_i (1-\bar{\alpha}_i)}$. However, Ho et al. (2020) found that a reweighted loss with $K_i = 1$ worked best in practice.

In this manuscript, the data consists of samples from the CG Boltzmann distribution: $q(\mathbf{z}_0) \propto e^{-\frac{V(\mathbf{z})}{k_B \mathcal{T}}}$. Given a trained diffusion model parameterized through a noise prediction network $\epsilon_\theta(\mathbf{z}_i, i)$, we can produce i.i.d. samples of the approximate CG distribution through ancestral sampling from the graphical model $p(\mathbf{z}_L)\prod_{i=1}^{L} p_\theta(\mathbf{z}_{i-1} \mid \mathbf{z}_i)$.

3.1. Extracting Force Fields from Diffusion Models

Song et al. (2020) demonstrated that the DDPM loss in Eq. 3 with $K_i = 1$ is equivalent to the following weighted sum of denoising score matching objectives (Vincent, 2011):

$$\sum_{i=1}^{L} (1-\bar{\alpha}_i)\mathbb{E}_{q(\mathbf{z}_0)}\mathbb{E}_{q(\mathbf{z}_i|\mathbf{z}_0)}\left[\left\|s_\theta(\mathbf{z}_i, i) - \nabla_{\mathbf{z}_i} \log q(\mathbf{z}_i \mid \mathbf{z}_0)\right\|^2\right]. \tag{4}$$

Here, $q(\mathbf{z}_i \mid \mathbf{z}_0) = \mathcal{N}(\mathbf{z}_i; \sqrt{\bar{\alpha}_i}\mathbf{z}_0, (1-\bar{\alpha}_i)\boldsymbol{I})$, and $s_\theta(\mathbf{z}_i, i)$ is the score model. While this was not made explicit by

Song et al. (2020), the equivalence of these two losses is achieved by relating the score model $s_\theta(\mathbf{z}_i, i)$ to the noise prediction network $\epsilon_\theta(\mathbf{z}_i, i)$ through $s_\theta(\mathbf{z}_i, i) = -\frac{\epsilon_\theta(\mathbf{z}_i, i)}{\sqrt{1-\bar{\alpha}_i}}$, see Appendix A.1. Given a sufficiently expressive model and sufficient amounts of data, the optimal score $s_{\theta^*}(\mathbf{z}_i, i)$ will match the score $\nabla_{\mathbf{z}_i} \log q(\mathbf{z}_i)$ (Vincent, 2011), where $q(\mathbf{z}_i) = \int \mathrm{d}\mathbf{z}_0 q(\mathbf{z}_i \mid \mathbf{z}_0) q(\mathbf{z}_0)$ is the marginal distribution at level i of the forward diffusion process. At sufficiently low noise levels, the marginal distribution $q(\mathbf{z}_i)$ will resemble the data distribution $q(\mathbf{z}_0)$, such that $s_{\theta^*}(\mathbf{z}_i, i)$ effectively approximates the score of the unknown data distribution. When the latter is equal to the CG Boltzmann distribution $q(\mathbf{z}_0) \propto e^{-\frac{V(\mathbf{z})}{k_B \mathcal{T}}}$, the optimal score $s_{\theta^*}(\mathbf{z}_i, i)$ at level $i = 1$ will approximately match the CG forces $\nabla_{\mathbf{z}} \log q(\mathbf{z}) = \frac{-\nabla_{\mathbf{z}} V(\mathbf{z})}{k_B \mathcal{T}} = \frac{\mathbf{F}_{\mathbf{z}}}{k_B \mathcal{T}}$. Finally, by using the relation between $s_\theta(\mathbf{z}_i, i)$ and the noise prediction network $\epsilon_\theta(\mathbf{z}_i, i)$, we can extract the approximate CG forces from a denoising diffusion model trained with the loss in Eq. 3:

$$\mathbf{F}_{\mathbf{z}}^{DFF} = -\frac{k_B \mathcal{T}}{\sqrt{1-\bar{\alpha}_i}} \epsilon_{\theta^*}(\mathbf{z}, i). \tag{5}$$

We will refer to such an approximate CG force field as a Denoising Force Field (DFF). While in principle the lowest level ($i = 1$) should provide the best approximation to the CG forces, in practice, we treat i as a hyperparameter and pick the best i by cross-validating the simulated dynamics.

Connections between force fields and denoising diffusion models have been made in previous work. Zaidi et al. (2022) pre-trained a property prediction graph neural network in a denoising diffusion setup by denoising molecular structures that locally maximize the Boltzmann distribution (or minimize the energy). By approximating the data distribution as a mixture of Gaussians centered around these local minima, they demonstrate that the score matching objective is equivalent to learning the force field of this approximate mixture of Gaussians data distribution. Similarly, Xie et al. (2022) connected the learned score in a denoising network for small noise levels to a harmonic force field around energy local minima structures. A key point is that these connections only provide approximate force fields around the local minima structures, making them of limited use in downstream tasks. In this work, we show that training denoising diffusion models on samples from the equilibrium Boltzmann distribution—rather than only the locally maximizing structures—allows us to learn an approximate force field in an unsupervised manner for the entire equilibrium distribution. This is crucial for running stable and reliable CG MD simulations with the extracted CG force field.

3.2. Molecular Dynamics with the Denoising Force Field

With the DFF from Eq. 5, we can perform CG molecular dynamics simulations by propagating the Langevin equation

$$M \frac{\mathrm{d}^2 \mathbf{z}}{\mathrm{d}\, t^2} = -\nabla_{\mathbf{z}} V(\mathbf{z}) - \gamma M \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} + \sqrt{2M\gamma k_B \mathcal{T}} \mathbf{w}(t), \tag{6}$$

where we substitute $-\nabla_{\mathbf{z}} V(\mathbf{z}) = \mathbf{F}_{\mathbf{z}}^{DFF}$. M represents the mass of the CG beads, γ is a friction coefficient, and $\mathbf{w}(t)$ is a delta-correlated stationary Gaussian process $\mathbb{E}_{p(\mathbf{x})}[\mathbf{w}(t) \cdot \mathbf{w}(t')] = \delta(t - t')$ with mean $\mathbb{E}_{p(\mathbf{x})}[\mathbf{w}(t)] = 0$. In our experiments, we set γ and $\mathcal{T}$ to the same values as those used in the original atomistic simulations that produced the data. Therefore, given a trained network ϵ_θ, the only hyper-parameter left to tune is the noise level i.

A well-known limit of the Langevin equation (Eq. 6) is that of a negligible mass and a large friction coefficient (with a finite $\eta = \gamma M$), called Brownian dynamics or overdamped Langevin dynamics. Interestingly, in Appendix A.2 we show that iteratively diffusing and denoising at a low-noise level (e.g. $i = 1$) approximates Brownian dynamics with a simulation timestep Δt implicitly defined through $\Delta t \frac{k_B \mathcal{T}}{M\gamma} = 1 - \bar{\alpha}_1 = \beta_1$.

3.3. Denoising Force Field Architecture

The choice of the neural network ϵ_θ is heavily influenced by the physical symmetries of the system under study. For instance, the CG force field must be conservative, i.e. it must equal the negative gradient of a CG energy potential $V_\theta(\mathbf{z})$. Therefore, we parameterize $\epsilon_\theta(\mathbf{z}_i, i)$ as the gradient of an energy neural network with a scalar output, *i.e.* $\epsilon_\theta(\mathbf{z}_i, i) = \nabla_{\mathbf{z}_i} \mathrm{nn}_\theta(\mathbf{z}_i, i)$, with $\mathrm{nn}_\theta : \mathbb{R}^{3n} \times \{1, ..., L\} \mapsto \mathbb{R}$. Previous studies on image generation by Salimans & Ho (2021) yielded no empirical difference in sample quality when using an unconstrained score network or a score that is parameterized as the gradient of an energy function. However, in Appendix B.1 we demonstrate that using a conservative score in a diffusion model is crucial for stable CG MD simulations with the extracted denoising force field.

Furthermore, the force field must be translation invariant and rotation equivariant. We ensure the model is translation invariant by using the coordinates of the CG beads only through pairwise difference vectors $\mathbf{z}_{(i)} - \mathbf{z}_{(j)}$ as input to the network. While the forces must be equivariant to rotations ,we explicitly do *not* want reflection equivariance, to avoid generating mirrored proteins as reported by Trippe et al. (2022). In other words, we want equivariance with respect to SO(3) instead of O(3). A simple strategy to approximate O(3) equivariance without requiring the more expensive spherical harmonics or angular representations is the use of data augmentation. Previous work by Gruver et al. (2022) showed that learned equivariance with transformers can be competitive with actual equivariant networks. In

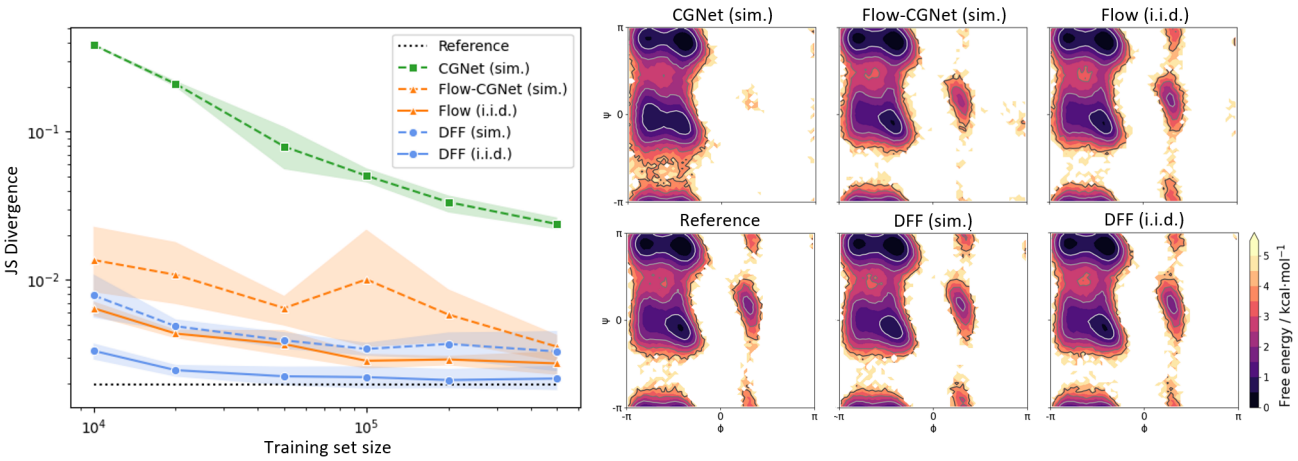


Figure 2. Experimental results for alanine dipeptide. Left: Jensen-Shannon divergence between dihedral distributions produced by several CG methods for different training set sizes and the test partition of the all atom simulation data projected onto the CG resolution. Results are averaged over four runs, and error bars denote a 95% confidence interval. Right: Ramachandran plots showing the dihedral distributions for the different methods trained on 500K samples.

Appendix B.3 we show that our denoising force field learns to be rotation equivariant on a validation set with a relative squared error introduced by rotations of $< 10^{-6}$.

In this work, we model the network nn_θ as a graph transformer adapted with the above symmetry constraints. Further architecture details are given in Appendix C.1. Note that previous works on neural-network-based CG force fields also often add a prior energy term in the scalar energy neural network to enforce better behavior of the CG force field further away from the training dataset (Wang et al., 2019; Husic et al., 2020; Köhler et al., 2023). In contrast, we did not find this to be necessary to obtain stable CG MD simulations with our denoising CG force field.

4. Experiments

By training our diffusion model on samples from a CG equilibrium distribution, we simultaneously obtain an i.i.d. sample generator (denoted *DFF i.i.d.*) as well as a CG force field for running CG MD simulations (*DFF sim.*). In this section, we evaluate the performance and scalability of our model for both use cases on (i) alanine dipeptide, and (ii) several fast-folding proteins (Lindorff-Larsen et al., 2011). In particular, we investigate how well the CG equilibrium distribution and the dynamics can be reproduced.

We compare our model to three baselines: *Flow i.i.d.* and *Flow-CGNet sim.* from Köhler et al. (2023) and *CGNet sim.* (Wang et al., 2019). *CGNet sim.* is a pure force-matching neural network trained on CG forces that were projected from the fine-grained representation onto the CG representation. *Flow i.i.d.* is the force-agnostic augmented normalizing flow model trained as a density estimator in the first stage of the flow-matching setup (Köhler et al., 2023). This flow model can only be used to produce i.i.d. samples. *Flow-CGNet sim.* performs CG simulations using the deterministic CGNet force field distilled from the gradient of the augmented normalizing flow model in the second teacher-student distillation stage of flow-matching. Recall that for our method, we do not require a teacher-student setup since the same network can be used for i.i.d. sampling and for CG simulations. We also provide *reference* data, which is the original MD simulation projected onto the CG resolution. Lastly, note that while we often show results for both i.i.d. and simulation-based methods, the latter have the more challenging task to also model the dynamics in order to obtain correct equilibrium distributions. We therefore expect the proposed i.i.d. methods to perform better when analyzing equilibrium distributions.

4.1. Coarse-Grained Simulation — Alanine Dipeptide

First, we evaluate our method on a CG representation of the well-studied alanine dipeptide system. We use the same CG representation as in Köhler et al. (2023); Wang et al. (2019); Husic et al. (2020), which projects all atoms onto the five central backbone atoms of the molecule (see Figure C1). The simulated data (Köhler et al., 2023) consists of four independent runs of length 500 ns, with 250 000 samples saved per simulation (2 ps intervals). We evaluate the model using four-fold cross-validation, where three of the simulations are used for training and validation, and one is used for testing. We consider different training dataset sizes, ranging from 10K to 500K training samples. For the Langevin dynamics simulation, we follow the same settings as Köhler et al. (2023), *i.e.* we run the simulation at 300 Kelvin for 1M steps with a step size of 2 fs and store the samples every 250 time steps. However, unlike Köhler et al. (2023), we do not use parallel tempering, which is known to improve

Table 1. Experimental results for fast-folders. The table displays the Jensen-Shannon (JS) divergence for TIC distributions and pairwise distance (PWD) distributions, where in the latter case an average is taken over all entries of the upper triangle of the PWD matrix with offset three. The JS divergences compare distributions from the atomistic MD simulations that were projected on CG space, and the distributions produced by the learned CG methods.

		Chignolin		Trp-cage		Bba		Villin		Protein G	
		TIC JS	PWD JS	TIC JS	PWD JS	TIC JS	PWD JS	TIC JS	PWD JS	TIC JS	PWD JS
Reference		.0057	.0002	.0026	.0002	.0040	.0002	.0032	.0004	.0014	.0002
Flow	i.i.d.	.0106	.0022	.0078	.0057	.0229	.0073	.0109	.0142	n/a	n/a
DFF		**.0096**	**.0005**	**.0052**	**.0007**	**.0111**	**.0017**	**.0073**	**.0009**	.0131	.0009
Flow-CGNet	sim.	.1875	.1271	.1009	.0474	.1469	.0594	.2153	.0535	n/a	n/a
DFF		**.0335**	**.0067**	**.0518**	**.0403**	**.1289**	**.0408**	**.0564**	**.0244**	.2260	.0691

the mixing of the dynamics. Our denoising network nn_θ (Section 3.3) consists of two graph transformer layers with 96 features in the hidden layers. Further implementation details are in Appendix C.4.1.

Metrics. Following Köhler et al. (2023); Wang et al. (2019), we evaluate the quality of the generated samples by analyzing statistics over the two dihedral angles (ϕ, ψ) computed along the CG backbone of alanine dipeptide. Each angle describes a four-body interaction, representing the main degrees of freedom of the system. We generate a Ramachandran plot by computing the free energy as a function of these two angles, binning values into a 2D histogram, and taking the negative logarithm of the probability density. To provide a quantitative analysis, we measure the empirical Jensen-Shannon (JS) divergence between the dihedral distributions of samples drawn from the model and the test set. A reference comparing the training and test sets is provided as a lower bound.

Results. As shown in Figure 2 (left), *DFF sim.* significantly outperforms previous CG simulation methods (*Flow-CGNet sim.* and *CGNet sim.*), especially in the low-data regime, and even performs comparably to the i.i.d. sampling method *Flow i.i.d.*. Moreover, *DFF i.i.d.* outperforms its counterpart *Flow i.i.d.* with a significant margin, almost approaching the performance of the lower bound (*Reference*). The right side of Figure 2 shows the Ramachandran plots after training on 500 K samples, further highlighting that our model is able to generate realistic samples.

4.2. Coarse-Grained Simulation — Fast-folding Proteins

Next, we evaluate our model on a more challenging set of fast-folding proteins (Lindorff-Larsen et al., 2011). Such proteins exhibit folding and unfolding events, which makes their simulated trajectories particularly interesting. We pick the same proteins as in Köhler et al. (2023), namely Chignolin, Trp-cage, Bba and Villin. These were coarse-grained by slicing out the C_α atom for every amino acid, yielding one bead per residue (10, 20, 28, and 35 beads, respectively). For these proteins, we produce the *Flow i.i.d.* and *Flow-CGNet sim.* plots through samples that were made publicly available by the authors. Since scaling to larger proteins was found to be challenging for flow-matching (Köhler et al., 2023), we additionally include the larger "Protein G" (56 beads) to analyze the scalability of our method. The all atom simulations vary in length, but for each trajectory the frames are shuffled and split 70-10-20% into a training, validation and test set. More dataset details are in Appendix C.5.1.

4.2.1. Equilibrium Analysis

Metrics. We use several metrics to evaluate the quality of the generated equilibrium distributions. First, we analyze the slowest changes in the protein conformation, which are usually related to (un-)folding events. For this, we calculate the time-lagged independent component analysis (TICA) (Naritomi & Fuchigami, 2011; Pérez-Hernández et al., 2013; Schwantes & Pande, 2013) using the Deeptime library (Hoffmann et al., 2021) and pick the first two TIC coordinates, resulting in a 2D distribution over the slowest processes. Basins in these 2D distributions are associated with meta-stable states. Further, we compute the JS divergence of the obtained TIC distributions between each model and the *reference* MD data (denoted by TIC JS). As a qualitative analysis, we plot the log of the obtained TIC distributions.

To assess the global structure of the proteins, we compare pairwise distance distributions by calculating the JS divergence relative to the test MD distribution for all distances within the upper triangle of the pairwise distance matrix with a diagonal offset larger than three (denoted by PWD JS). The offset is chosen to avoid over-representing local structure. Moreover, we plot the free energy as a function of the root mean squared distance (RMSD) between the generated samples and the native, folded structure for all C_α atoms. Dips in the resulting curve correspond to meta-stable

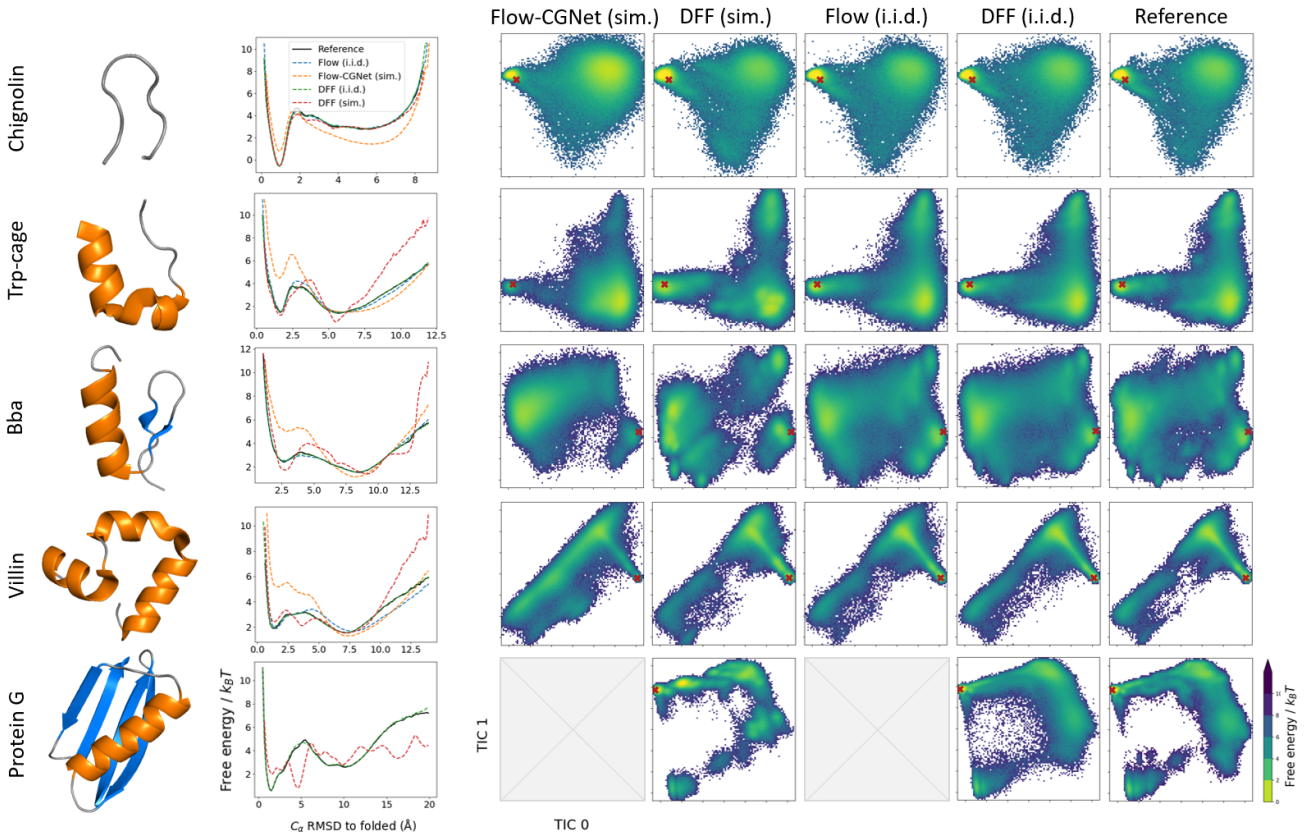


Figure 3. Left: native structure visualization (α-helices in orange, β-sheets in blue) made with PyMOL. Middle: C_α-RMSD free energy w.r.t. the folded native structure. Right: joint density plots for the two slowest TIC coordinates, where the color indicates the free energy value. The red cross indicates the location of the native structure.

ensembles, with higher densities of samples representing a bigger proportion of the distribution. Finally, we analyze the normalized count over contact maps, which results in a 2D histogram that shows the probability of two atoms being in contact, *i.e.* within a threshold of 10 Å of one another.

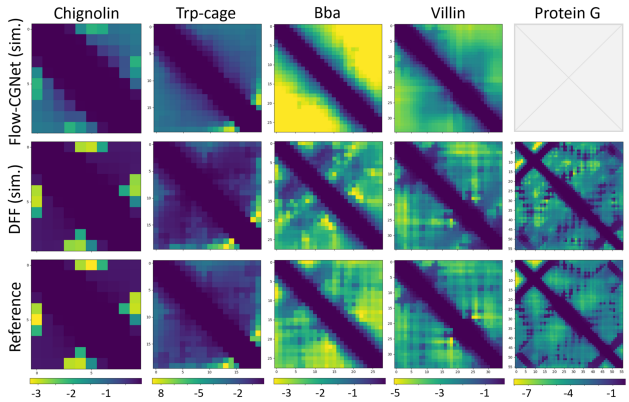


Figure 4. Contact maps for *Flow-CGNet sim.*, *DFF sim.* and the reference MD data for fast-folding proteins. The contact threshold is set to 10 Å. The axes in the plot represent atom indices.

Results. Table 1 shows that *DFF i.i.d.* and *DFF sim.* consistently outperform their respective baselines across the equilibrium metrics (TIC JS and PWD JS). As shown in Figure 3, the TIC 2D free energy landscapes for our model look more similar to the reference MD distribution (as reflected in the TIC JS metrics), especially for Chignolin and Villin. The similarity is weaker for Bba, where local modes are more dominant. Further, the free energy curves as a function of the RMSD are always overlapping with the reference curve for *DFF i.i.d.*, and are close to the reference MD curve in dense regions for *DFF sim.*.

Figure 4 shows further qualitative results in the form of a normalized count over contact maps for *DFF sim.* and *Flow-CGNet sim.*; for i.i.d. models, see Appendix C.5.3. As can be seen, the *DFF* models capture contact probabilities much better than the flow-based models in all proteins, especially in the off-diagonal regions that represent global structure. These results are closely related to the JS divergences between pairwise off-diagonal distances (Table 1). Taken together, these results indicate that diffusion-based models capture global structure better than their flow-based baselines. Moreover, the analysis in Appendix C.5.3 shows that the CG fast folder samples produced by *DFF sim.* do not display chemical integrity violations such as bond dissociations or backbone crossings, therefore, *DFF sim.* does not require an energy prior as used in *Flow-CGNet sim.* to

Table 2. Average and state-probability weighted JS divergence between the reference MD data and model simulations for transition probabilities of the estimated Markov model.

	Chignolin		Trp-cage		Bba		Villin	
	Average	Weighted	Average	Weighted	Average	Weighted	Average	Weighted
Flow-CGNet	$2.5 \cdot 10^{-2}$	$5.7 \cdot 10^{-3}$	$4.8 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$6.9 \cdot 10^{-2}$	$6.8 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$2.9 \cdot 10^{-2}$
DFF	$\mathbf{9.7 \cdot 10^{-4}}$	$\mathbf{5.1 \cdot 10^{-4}}$	$\mathbf{1.3 \cdot 10^{-3}}$	$\mathbf{7.5 \cdot 10^{-4}}$	$\mathbf{4.0 \cdot 10^{-3}}$	$\mathbf{4.2 \cdot 10^{-3}}$	$\mathbf{1.2 \cdot 10^{-4}}$	$\mathbf{2.1 \cdot 10^{-5}}$

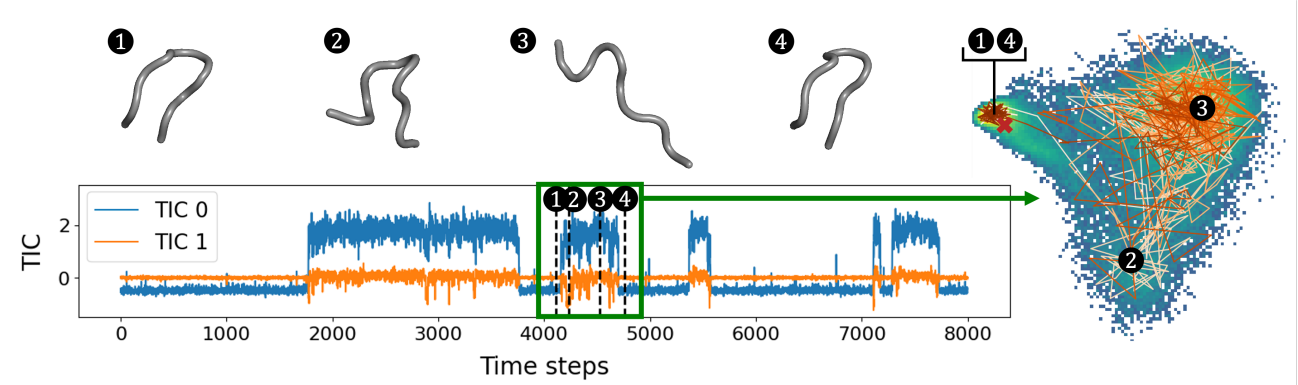


Figure 5. First two TIC coordinates tracked over time for Chignolin, with zoom-in on an unfolding and folding event, showing a 2D trajectory though TIC space and four structures along the path.

run stable simulations. Finally, Protein G is a larger and more complex protein compared to the other fast-folders and out of reach for flow-matching models. Our results show that diffusion-based models are scalable to this larger protein and can capture the global structure.

As a limitation in our method, we found that while the *DFF i.i.d.* model generally improves as we increase the number of features/layers in our neural network, the performance of the simulations obtained by *DFF sim.* is sensitive to the bias/variance trade-off in the network, and it can actually decrease for more flexible networks. See Appendix B.2 for an example in Chignolin.

4.2.2. Dynamics Analysis

Metrics. We qualitatively assess the simulated trajectories by tracking the first two TIC coordinates over time and showing (part of) the corresponding trajectory in 2D TIC space. We visualize (un-)folding events with the corresponding structures along the path. As a quantitative measure, we extract the transition probabilities from one conformational state to the other as follows. First, we use K-means clustering to divide the 2D TIC space into K clusters for the full (unsplit) MD dataset, with K determined by the elbow method. Next, all transitions are counted and normalized to obtain a transition probability matrix corresponding to the estimated Markov model (Prinz et al., 2011), where each row can be compared to MD data using the JS divergence. Even though the relation between fine-grained and coarse-grained time is non-trivial (Jin et al., 2022; Nüske et al., 2019), leading to different time lags, we can still evaluate how well a coarse-grained model reproduces the kinetic model of the fine-grained reference distribution. We show the average JS divergence over all starting states as well as the average weighted by the overall state probability as estimated from the reference data. Note that this metric can only be calculated for simulation samples (*i.e. Flow-CGNet* and *DFF* simulations). We cannot calculate a reference value here that compares transition probabilities between the train and test splits, since the MD dataset was shuffled before it was split and thus all sets contain frames that are distributed across time without retaining temporal information.

Results. Figure 5 depicts the first two TIC coordinates for a DFF sim. trajectory, clearly showing transitions from the folded to unfolded conformations. This is further highlighted by zooming in on part of the trajectory, revealing how the trajectory moves in 2D TIC space, and what the conformations look like in different parts of the landscape. The results of the transition probability analysis are shown in Table 2 for all fast-folders except Protein G, since no *Flow-CGNet sim.* samples were available for this larger protein. The *DFF sim.* model outperforms the *Flow-CGNet sim.* model across all fast-folders, showing a better preservation of dynamics. More results on transition probability matrices and the clustering of 2D TIC space are in Appendix C.5.3.

5. Conclusions

We have presented a new approach to CG molecular dynamics modeling based on denoising diffusion models, motivated by connections between score-based generative models, force fields and molecular dynamics. This results in a simple training setup as well as improved performance and scalability compared to previous work. Future directions to improve our work include scaling to larger proteins and generalizing across different systems. Another interesting direction would be to combine the current force-agnostic training approach with an explicit force-matching objective if such force information is available.

6. Acknowledgements

We would like to express our gratitude to Yaoyi Chen for providing samples and evaluation scripts for the prior work by Köhler et al. (2023). We would also like to thank Max Welling, Leon Klein, Tor Erlend Fjelde and Andreas Krämer for the insightful discussions and suggestions.

References

Chen, J., Chen, J., Pinamonti, G., and Clementi, C. Learning effective molecular models from experimental observables. Journal of Chemical Theory and Computation, 14 (7):3849–3858, 2018.

Chen, J., Lu, C., Chenli, B., Zhu, J., and Tian, T. Vflow: More expressive generative flows with variational data augmentation. In International Conference on Machine Learning, pp. 1660–1669. PMLR, 2020.

Chu, J.-W. and Voth, G. A. Allostery of actin filaments: Molecular dynamics simulations and coarse-grained analysis. Proceedings of the National Academy of Sciences, 102(37):13111–13116, 2005.

Ciccotti, G., Kapral, R., and Vanden-Eijnden, E. Blue moon sampling, vectorial reaction coordinates, and unbiased constrained dynamics. ChemPhysChem, 6:1809–1814, 9 2005.

Clementi, C. Coarse-grained models of protein folding: toy models or predictive tools? Curr. Opin. Struct. Biol., 18 (1):10–15, 2008.

Corso, G., Stärk, H., Jing, B., Barzilay, R., and Jaakkola, T. Diffdock: Diffusion steps, twists, and turns for molecular docking. arXiv preprint arXiv:2210.01776, 2022.

Davtyan, A., Schafer, N. P., Zheng, W., Clementi, C., Wolynes, P. G., and Papoian, G. A. Awsem-md: protein structure prediction using coarse-grained physical potentials and bioinformatically based local structure biasing. The Journal of Physical Chemistry B, 116(29): 8494–8503, 2012.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. arXiv preprint arXiv:1605.08803, 2016.

Gruver, N., Finzi, M., Goldblum, M., and Wilson, A. G. The lie derivative for measuring learned equivariance. arXiv preprint arXiv:2210.02984, 2022.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. Neural computation, 14(8):1771–1800, 2002.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33:6840–6851, 2020.

Hoffmann, M., Scherer, M., Hempel, T., Mardt, A., de Silva, B., Husic, B. E., Klus, S., Wu, H., Kutz, N., Brunton, S. L., et al. Deeptime: a python library for machine learning dynamical models from time series data. Machine Learning: Science and Technology, 3(1):015009, 2021.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In International Conference on Machine Learning, pp. 8867–8887. PMLR, 2022.

Huang, C.-W., Dinh, L., and Courville, A. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. arXiv preprint arXiv:2002.07101, 2020.

Husic, B. E., Charron, N. E., Lemm, D., Wang, J., Pérez, A., Majewski, M., Krämer, A., Chen, Y., Olsson, S., de Fabritiis, G., et al. Coarse graining molecular dynamics with graph neural networks. The Journal of chemical physics, 153(19):194101, 2020.

Igashov, I., Stärk, H., Vignac, C., Satorras, V. G., Frossard, P., Welling, M., Bronstein, M., and Correia, B. Equivariant 3d-conditional diffusion models for molecular linker design. arXiv preprint arXiv:2210.05274, 2022.

Jin, J., Schweizer, K. S., and Voth, G. A. Understanding dynamics in coarse-grained models: I. universal excess entropy scaling relationship. The Journal of Chemical Physics, 2022.

Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. Torsional diffusion for molecular conformer generation. arXiv preprint arXiv:2206.01729, 2022.

Kmiecik, S., Gront, D., Kolinski, M., Wieteska, L., Dawid, A. E., and Kolinski, A. Coarse-grained protein models and their applications. Chemical reviews, 116(14):7898–7936, 2016.

Köhler, J., Chen, Y., Krämer, A., Clementi, C., and Noé, F. Flow-matching: Efficient coarse-graining of molecular dynamics without forces. Journal of Chemical Theory and Computation, 2023.

Lindorff-Larsen, K., Piana, S., Dror, R. O., and Shaw, D. E. How fast-folding proteins fold. Science, 334(6055):517–520, 2011.

Marrink, S. J., Risselada, H. J., Yefimov, S., Tieleman, D. P., and De Vries, A. H. The martini force field: coarse grained model for biomolecular simulations. The journal of physical chemistry B, 111(27):7812–7824, 2007.

Matysiak, S. and Clementi, C. Minimalist protein model as a diagnostic tool for misfolding and aggregation. Journal of Molecular Biology, 363(1):297–308, 2006.

Mim, C., Cui, H., Gawronski-Salerno, J. A., Frost, A., Lyman, E., Voth, G. A., and Unger, V. M. Structural basis of membrane bending by the n-BAR protein endophilin. Cell, 149(1):137–145, 2012.

Naritomi, Y. and Fuchigami, S. Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: the case of domain motions. The Journal of chemical physics, 134(6):02B617, 2011.

Noid, W. G. Perspective: Coarse-grained models for biomolecular systems. The Journal of chemical physics, 139(9):09B201_1, 2013.

Noid, W. G., Chu, J.-W., Ayton, G. S., Krishna, V., Izvekov, S., Voth, G. A., Das, A., and Andersen, H. C. The multiscale coarse-graining method. i. a rigorous bridge between atomistic and coarse-grained models. The Journal of chemical physics, 128(24):244114, 2008.

Nüske, F., Boninsegna, L., and Clementi, C. Coarse-graining molecular systems by spectral matching. The Journal of Chemical Physics, 151(4):044116, 2019.

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

Pérez-Hernández, G., Paul, F., Giorgino, T., De Fabritiis, G., and Noé, F. Identification of slow molecular order parameters for markov model construction. The Journal of chemical physics, 139(1):07B604_1, 2013.

Prinz, J.-H., Wu, H., Sarich, M., Keller, B., Senne, M., Held, M., Chodera, J. D., Schütte, C., and Noé, F. Markov models of molecular kinetics: Generation and validation. The Journal of Chemical Physics, 134(17):174105, 2011.

Qiao, Z., Nie, W., Vahdat, A., Miller III, T. F., and Anandkumar, A. Dynamic-backbone protein-ligand structure prediction with multiscale generative diffusion models. arXiv preprint arXiv:2209.15171, 2022.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In International conference on machine learning, pp. 1530–1538. PMLR, 2015.

Salimans, T. and Ho, J. Should ebms model the energy or the score? In Energy Based Models Workshop-ICLR 2021, 2021.

Saunders, M. G. and Voth, G. A. Coarse-graining methods for computational biology. Annual review of biophysics, 42:73–93, 2013.

Schwantes, C. R. and Pande, V. S. Improvements in markov state model construction reveal many non-native interactions in the folding of ntl9. Journal of chemical theory and computation, 9(4):2000–2009, 2013.

Shell, M. S. The relative entropy is fundamental to multiscale and inverse thermodynamic problems. The Journal of chemical physics, 129(14):144108, 2008.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pp. 2256–2265. PMLR, 2015.

Song, Y. and Kingma, D. P. How to train your energy-based models. arXiv preprint arXiv:2101.03288, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456, 2020.

Thaler, S., Stupp, M., and Zavadlav, J. Deep coarse-grained potentials via relative entropy minimization. arXiv preprint arXiv:2208.10330, 2022.

Trippe, B. L., Yim, J., Tischer, D., Broderick, T., Baker, D., Barzilay, R., and Jaakkola, T. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. arXiv preprint arXiv:2206.04119, 2022.

Vincent, P. A connection between score matching and denoising autoencoders. Neural computation, 23(7):1661–1674, 2011.

Wang, J., Olsson, S., Wehmeyer, C., Pérez, A., Charron, N. E., De Fabritiis, G., Noé, F., and Clementi, C. Machine learning of coarse-grained molecular dynamics force fields. ACS central science, 5(5):755–767, 2019.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. bioRxiv, 2022.

Wu, K. E., Yang, K. K., Berg, R. v. d., Zou, J. Y., Lu, A. X., and Amini, A. P. Protein structure generation via folding diffusion. arXiv preprint arXiv:2209.15611, 2022.

Xie, T., Fu, X., Ganea, O.-E., Barzilay, R., and Jaakkola, T. S. Crystal diffusion variational autoencoder for periodic material generation. In International Conference on Learning Representations, 2022.

Yu, A., Pak, A. J., He, P., Monje-Galvan, V., Casalino, L., Gaieb, Z., Dommer, A. C., Amaro, R. E., and Voth, G. A. A multiscale coarse-grained model of the SARS-CoV-2 virion. Biophysical Journal, 120(6):1097–1104, 2021.

Zaidi, S., Schaarschmidt, M., Martens, J., Kim, H., Teh, Y. W., Sanchez-Gonzalez, A., Battaglia, P., Pascanu, R., and Godwin, J. Pre-training via denoising for molecular property prediction. arXiv preprint arXiv:2206.00133, 2022.

A. Derivations

A.1. Relation between Score Function $s_\theta(\mathbf{z}_i, i)$ and Noise Predicting Network $\epsilon_\theta(\mathbf{z}_i, i)$

In this section, we show that Eq. 3 and 4 are equal up to a reweighting of the summands by setting:

$$s_\theta(\mathbf{z}_i, i) = -\frac{\epsilon(\mathbf{z}_i, i)}{\sqrt{1-\bar{\alpha}_i}}.$$

We start from Eq. 4:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}_i,\mathbf{z}_0}\left[\left\|s_\theta(\mathbf{z}_i,i) - \nabla_{\mathbf{z}_i}\log q(\mathbf{z}_i \mid \mathbf{z}_0)\right\|^2\right] \\
=&\mathbb{E}_{\mathbf{z}_i,\mathbf{z}_0}\left[\left\|s_\theta(\mathbf{z}_i,i) + \frac{\mathbf{z}_i - \sqrt{\bar{\alpha}_i}\mathbf{z}_0}{1-\bar{\alpha}_i}\right\|^2\right] && \text{Substitute } q(\mathbf{z}_i \mid \mathbf{z}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_i}\mathbf{z}_0, (1-\bar{\alpha}_i)\boldsymbol{I}) \\
=&\mathbb{E}_{\epsilon,\mathbf{z}_0}\left[\left\|s_\theta(\sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_i}\epsilon, i) + \frac{\sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_i}\epsilon - \sqrt{\bar{\alpha}_i}\mathbf{z}_0}{1-\bar{\alpha}_i}\right\|^2\right] && \text{Reparameterize } \mathbf{z}_i = \sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_i}\epsilon \\
=&\mathbb{E}_{\epsilon,\mathbf{z}_0}\left[\left\|s_\theta(\sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_i}\epsilon, i) + \frac{\epsilon}{\sqrt{1-\bar{\alpha}_i}}\right\|^2\right] \\
=&\mathbb{E}_{\epsilon,\mathbf{z}_0}\left[\left\|-\frac{\epsilon_\theta(\sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_i}\epsilon, i)}{\sqrt{1-\bar{\alpha}_i}} + \frac{\epsilon}{\sqrt{1-\bar{\alpha}_i}}\right\|^2\right] && \text{Substitute } s_\theta(\cdot, i) = -\frac{\epsilon_\theta(\cdot, i)}{\sqrt{1-\bar{\alpha}_i}} \\
=&\frac{1}{1-\bar{\alpha}_i}\mathbb{E}_{\epsilon,\mathbf{z}_0}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_i}, i)\right\|^2\right]. && \text{Reorganize}
\end{aligned}$$

That is, the noise-prediction loss of Eq. 3 can be seen as a reweighted denoising score matching loss, and vice versa.

A.2. Connecting Denoising Diffusion and the Brownian Dynamics

Recall from Section 3 that a denoising-diffusion model consists of an iterative diffusion process $q(\mathbf{z}_i \mid \mathbf{z}_{i-1}) = \mathcal{N}(\mathbf{z}_i; \sqrt{1-\beta_i}\mathbf{z}_{i-1}, \beta_i\boldsymbol{I})$ and a denoising process $p(\mathbf{z}_{i-1} \mid \mathbf{z}_i) = \mathcal{N}(\mathbf{z}_{i-1}; \mu_\theta(\mathbf{z}_i, i), \sigma_i^2\boldsymbol{I})$ of L steps, where β_i and σ_i^2 are the variance coefficients of the diffusion and denoising steps, respectively, and $i \in [1, \ldots, L]$. In this section, we show that iteratively performing a diffusion step followed by a denoising step at a low-noise level (e.g. $i = 1$) approximates the Brownian dynamics of a molecular system. We demonstrate that by tuning the first diffusion noise level β_1, we effectively modify the step size of a Brownian dynamics simulation, as the two can be related by $\beta_1 = 1 - \bar{\alpha}_1 = \Delta t \frac{k_B \mathcal{T}}{M\gamma}$. We start by re-parameterizing the diffusion step $\mathbf{z}_1 \sim q(\mathbf{z}_1 \mid \mathbf{z}_0)$ as:

$$\mathbf{z}_1 = \sqrt{1-\beta_1}\mathbf{z}_0 + \sqrt{\beta_1}\mathbf{w}_a, \tag{7}$$

where $\mathbf{w}_a \sim \mathcal{N}(0, \mathbf{I})$. Similarly, we re-parameterize $\mathbf{z}_0 \sim p(\mathbf{z}_0 \mid \mathbf{z}_1) = \mathcal{N}(\mathbf{z}_0; \mu_\theta(\mathbf{z}_1, 1), \sigma_1^2\boldsymbol{I})$ and rewrite μ_θ using the noise-predicting network ϵ_θ from Section 3 as:

$$\mathbf{z}_0 = \frac{1}{\sqrt{1-\sigma_1}}\big(\mathbf{z}_1 - \sqrt{\sigma_1}\epsilon_\theta(\mathbf{z}_1, 1)\big) + \sqrt{\sigma_1}\mathbf{w}_b. \tag{8}$$

Following Ho et al. (2020), we set $\sigma_i = \beta_i$. We now introduce a superscript index (t) to denote the sequence of states that unfold the diffusion-denoising process through time. Combining this notation with both Eq. 7 and 8 we obtain the following recursive update:

$$
\begin{aligned}
\mathbf{z}_1^{(t+1)} &= \sqrt{1-\beta_1}\left[\frac{1}{\sqrt{1-\beta_1}}\big(\mathbf{z}_1^{(t)} - \sqrt{\beta_1}\epsilon_\theta(\mathbf{z}_1^{(t)}, 1)\big) + \sqrt{\beta_1}\mathbf{w}_b\right] + \sqrt{\beta_1}\mathbf{w}_a && \text{Combine Eq. 7 and 8} \\
&= \mathbf{z}_1^{(t)} - \sqrt{\beta_1}\epsilon_\theta(\mathbf{z}_1^{(t)}, 1) + \sqrt{1-\beta_1}\sqrt{\beta_1}\mathbf{w}_b + \sqrt{\beta_1}\mathbf{w}_a && \text{Re-arrange terms} \\
&\overset{d}{=} \mathbf{z}_1^{(t)} - \sqrt{\beta_1}\epsilon_\theta(\mathbf{z}_1^{(t)}, 1) + \sqrt{2\beta_1 - \beta_1^2}\mathbf{w}_c && \text{Sum of the squares of the standard devs.} \\
&\approx \mathbf{z}_1^{(t)} - \sqrt{\beta_1}\epsilon_\theta(\mathbf{z}_1^{(t)}, 1) + \sqrt{2\beta_1}\mathbf{w}_c && \text{Taylor of } 2\beta_1 - \beta_1^2 \text{ at } (\beta_1 = 0) \to 2\beta_1 \\
&= \mathbf{z}_1^{(t)} - \beta_1\frac{\epsilon_\theta(\mathbf{z}_1^{(t)}, 1)}{\sqrt{\beta_1}} + \sqrt{2\beta_1}\mathbf{w}_c, && \text{Re-arrange}
\end{aligned}
$$

where $\mathbf{w}_c \sim \mathcal{N}(0, \mathbf{I})$ and $\overset{d}{=}$ denotes equal in distribution. To make the relation to Brownian dynamics more explicit, we can discretize the overdamped limit of the Langevin dynamics from Eq. 6 using the Euler-Maruyama scheme:

$$
\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)} - \Delta t\frac{\nabla_{\mathbf{z}}V(\mathbf{z})}{\gamma M} + \sqrt{\Delta t\frac{2k_B\mathcal{T}}{\gamma M}}\mathbf{w}, \tag{9}
$$

where $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$ and Δt is the simulation time step. By comparing the two equations, we see that the denoising-diffusion process corresponds to simulating a Brownian dynamics of a force field $\nabla_{\mathbf{z}}V(\mathbf{z}) = \frac{k_B\mathcal{T}}{\sqrt{\beta_1}}\epsilon_\theta(\mathbf{z}, i) = \frac{k_B\mathcal{T}}{\sqrt{1-\bar{\alpha}_1}}\epsilon_\theta(\mathbf{z}, i)$ and an implicit step size $\Delta t = \frac{M\gamma}{k_B\mathcal{T}}\beta_1$. Note that the force field is the same as the denoising force field (Eq. 5). Furthermore, the variance of the denoising-diffusion process $\beta_1 = 1 - \bar{\alpha}_1$ is proportional to the simulation time step Δt.

B. Additional Experiments

B.1. Ablation on Conservative Forces

Designing our neural network to approximate a conservative force is necessary to satisfy the following property in physics: *"A conservative force is a force with the property that the total work done in moving a particle between two points is independent of the path taken."*. This property can be satisfied by computing the force as the gradient of a scalar with respect to the input coordinates. In this section, we empirically show the benefits of using a conservative network when simulating dynamics (for *DFF sim.*). In contrast, the benefits for i.i.d. generation (*DFF i.i.d.*) are smaller.

We build a non-conservative network by replacing the last linear layer and average pooling operation of the conservative network, such that the network $\hat{\mathrm{nn}}_\theta : \mathbb{R}^{n\times 3} \to \mathbb{R}^{n\times 3}$ outputs a three-dimensional vector for each node (instead of a scalar). The non-conservative noise prediction network is then defined as $\epsilon_\theta = \hat{\mathrm{nn}}_\theta$.

Next, we compare the performance of both variants on the proteins Chignolin and Trp-cage using the same experimental settings as in Appendix C.5.

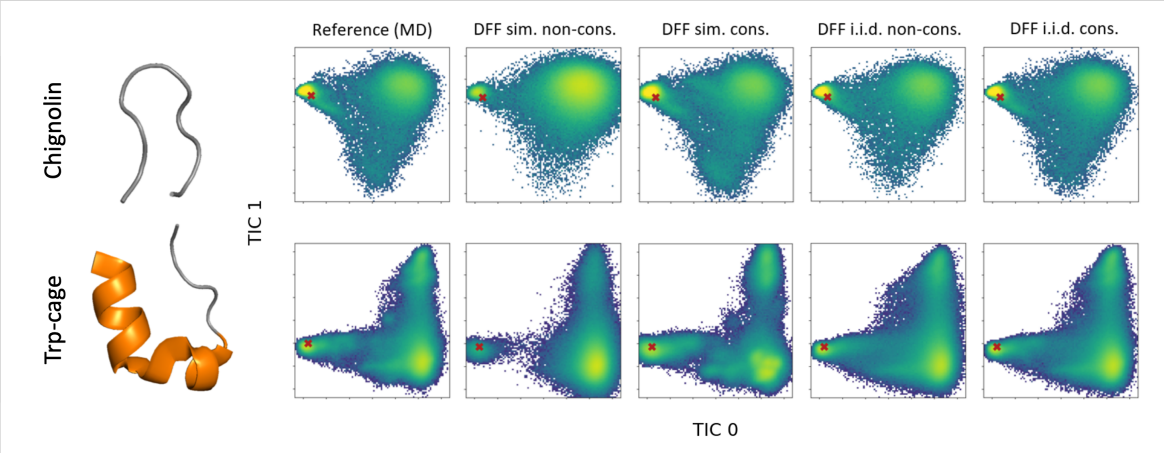


Figure B1. Ablation on TICA plots for chignolin and trp-cage fast folders comparing conservative vs non-conservative neural networks for simulated and i.i.d. data with our DFF model.

Table B1. Ablation study on conservative versus non-conservative variants in Chignolin and Trp-cage. We report the JS divergence between TICA distributions and pairwise distance distributions.

		Chignolin		Trp-cage	
		TIC JS	PWD JS	TIC JS	PWD JS
Reference		.0057	.0002	.0026	.0002
DFF (non-conservative)	i.i.d.	.0104	.0008	.0120	.0030
DFF	i.i.d.	.0096	.0005	.0052	.0007
DFF (non-conservative)	sim.	.3216	.2147	.0879	.0399
DFF	sim.	.0335	.0067	.0518	.0403

As illustrated in Figure B1, and numerically shown in Table B1, the conservative variant outperforms the non-conservative case in most metrics for Chignolin and Trp-cage. Notice this difference is very significant in Chignolin simulated data where the TIC JS performance differs by an order of magnitude and even more for PWD JS. In Chignolin, we also tried using a more powerful non-conservative network, and while the non-conservative dynamics would improve for deeper networks, the performance would still be quite far from the conservative DFF sim. metric reported in the current experiment. On the other hand, the performance gap between conservative and non-conservative force fields in i.i.d. generation was reduced when we increased the size of the network.

B.2. Ablation on Number of Hidden Features

In Appendix C.5, we mentioned that the expressivity of the neural network (e.g. number of hidden features) has a notable influence on the quality of the simulated dynamics. More specifically, while i.i.d. generation tends to perform better as we increase the expressivity of our network, dynamics simulations may become worse. We found that properly choosing the number of features in our network plays an important role in the quality of the generated samples. Here, we show that the sweet spot in the bias/variance trade-off is significantly more sensitive when using the force field to simulate dynamics than when doing i.i.d. generation with the same network. We report the performance of DFF (i.i.d. and sim.) for different number of hidden features $\{32, 64, 128, 256\}$. All other training settings are the same as described for Chignolin in the fast-folding protein experiment in Appendix C.5.2.

Figure B2 and Table B2 show that while the performance of i.i.d. generation improves as we increase the number of hidden features, the performance of simulated dynamics peaks at 64 features and deteriorates as we increase the network size. This implies that dynamics simulation is more sensitive to the number of features compared to i.i.d. sampling. The number of training iterations was set to 1M in all models and the validation curves on noise-prediction loss did not show symptoms of over-fitting.

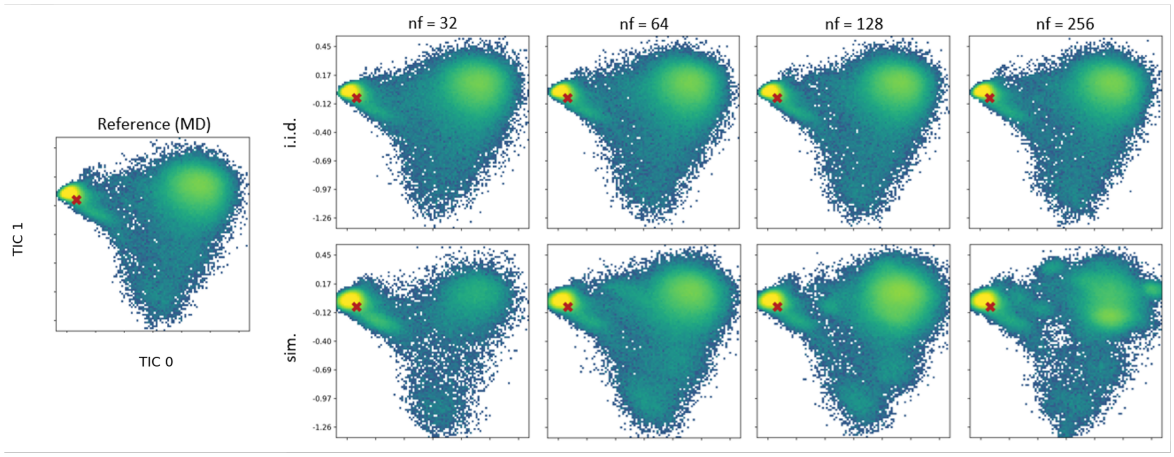


Figure B2. TICA plots for the ablation on the number of features per layer for Chignolin.

Table B2. TIC JS and PWD JS w.r.t the test partition with varying numbers of features. The model used to compute all metrics is our proposed DFF in both i.i.d. sampling and dynamics simulation.

	Number of features	Chignolin	
		TIC JS	PWD JS
Reference		.0057	.0002
i.i.d.	32	.0105	.0010
	64	.0096	.0005
	128	**.0091**	**.0004**
	256	**.0091**	**.0004**
sim.	32	.0692	.0318
	64	**.0335**	**.0067**
	128	.0434	.0179
	256	.0503	.0263

The proposed method (DFF sim.) has shown excellent performance in systems such as Alanine, Chignolin (nf=64) or Villin compared to previous ML coarse grained methods. Despite its good performance, we believe improving the robustness of the model via introducing new inductive biases or regularization techniques can be a promising direction in order to leverage higher-capacity neural networks without compromising the performance in simulation.

B.3. Rotation Equivariance through Data Augmentation

In this section, we experimentally confirm that using data augmentation for training the neural network ϵ_θ to be approximately equivariant to rotations is sufficient in practice. Recall that a function f is equivariant to rotations if rotating its input results in an equivalent rotation of its output, i.e. $\mathbf{R}f(\mathbf{z}) = f(\mathbf{R}\mathbf{z})$, where $\mathbf{R}$ is a rotation matrix. To analyze the degree of equivariance of our network ϵ_θ, we compute the relative squared error between $\epsilon_\theta(\mathbf{z})$ and $\mathbf{R}^{-1}\epsilon_\theta(\mathbf{R}\mathbf{z})$ for samples $\mathbf{z} \sim p_{\text{val}}(\mathbf{z})$ drawn from the validation partition. In Table B3, we show the errors of the networks $\epsilon_\theta(\cdot, i)$ trained on different proteins. Specifically, we evaluate the non-equivariance at the noise level i at which we extract the force field to simulate Langevin dynamics. These noise levels are $\{20, 15, 5, 5, 5\}$ for Chignolin, Trp-cage, Bba, Villin and Protein G, respectively. Since the relative squared error is in the order of $< 10^{-6}$ for all proteins, we conclude that applying data augmentation is sufficient for achieving rotation equivariance for most practical use cases.

Table B3. Relative squared error between $\epsilon_\theta(\mathbf{z})$ and $\mathbf{R}^{-1}\epsilon_\theta(\mathbf{R}\mathbf{z})$ across different fast-folding proteins.

	Chignolin	Trp-cage	Bba	Villin	Protein G
Relative squared error (mean)	$7.8 \cdot 10^{-7}$	$4.9 \cdot 10^{-7}$	$2.8 \cdot 10^{-7}$	$1.9 \cdot 10^{-7}$	$1.9 \cdot 10^{-7}$
Relative squared error (stdev)	$8.7 \cdot 10^{-7}$	$6.0 \cdot 10^{-7}$	$1.6 \cdot 10^{-7}$	$6.5 \cdot 10^{-8}$	$7.0 \cdot 10^{-8}$

C. Experimental Details

C.1. Architecture Details

In this section, we describe the neural network used to parameterize ϵ_θ. For this, we adapted Graph Transformer (GT) from https://github.com/lucidrains/graph-transformer-pytorch. We modified the architecture to satisfy the symmetry constraints from Section 3.3. We start by naming a function that calls the Graph Transformer from the mentioned link as

$$\text{nodes}_{\text{out}} = \text{GT}[\text{nodes}_{\text{in}}, \text{edges}_{\text{in}}],$$

where $\text{nodes}_{\text{in}} \in \mathbb{R}^{n \times d_n}$, $\text{edges}_{\text{in}} \in \mathbb{R}^{n \times n \times d_e}$. n being the number of nodes, d_n the number of dimensions per node and d_e the number of dimensions per edge. GT will receive as input the node embeddings $\mathbf{h} \in \mathbb{R}^{n \times (\cdot)}$ and the noise level index i as node features. Pairs of vector differences $\mathbf{z}_j - \mathbf{z}_k$ as edge features such that

$$\begin{aligned} \text{nodes}_{\text{in}}[j, :] &= \text{concat}\left[\mathbf{h}[j, :], i\right] \\ \text{edges}_{\text{in}}[j, k, :] &= \mathbf{z}[j, :] - \mathbf{z}[k, :] \end{aligned}$$

Now we define a network $\text{nn}_{\theta'} : [\mathbf{h}, \mathbf{z}, i] \to \mathbb{R}^1$ that takes the Graph Transformer and outputs a scalar value:

$$\text{nn}_{\theta'} : \{\mathbf{h}, \mathbf{z}, i\} \to \{\text{nodes}_{\text{in}}, \text{edges}_{\text{in}}\} \to \text{GT} \to \{\text{nodes}_{\text{out}}\} \to \text{nn.Linear}(d_n, 1) \to \text{sum}(\cdot) \to \{\text{output}\}$$

Finally, to define ϵ_θ, we incorporate $\mathbf{h}$ as its learnable parameters $\theta = \{\theta', \mathbf{h}\}$ and compute the gradient of $\text{nn}_{\theta'}$ w.r.t. $\mathbf{z}$

$$\epsilon_\theta(\mathbf{z}, i) = \nabla_{\mathbf{z}} \text{nn}_{\theta'}(\mathbf{h}, \mathbf{z}, i)$$

C.2. Coarse-graining Operator

In Figure C1, we give an overview of all systems we use in our experiments along with their CG representations. Coarse-graining is done by slicing out the relevant atoms, *i.e.* the backbone atoms for alanine-dipeptide, and C_α atoms for all fast-folding proteins.

C.3. The optimization objective

We defined the optimization objective of a diffusion model at a given noise level i in Section 3 as $\mathbb{E}_{q(\mathbf{z}_0)} \mathbb{E}_{\mathcal{N}(\epsilon; \mathbf{0}, \mathbf{I})} \left[\left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_i}\mathbf{z}_0 + \sqrt{1 - \bar{\alpha}_i}\epsilon, i) \right\|^2 \right]$. In (Ho et al., 2020), the noise level i is sampled from a discrete uniform distribution $i \sim \mathcal{U}\{1, \ldots, L\}$. In this work, in order to encourage the training at lower noise levels, we sample i from $\mathcal{U}\{1, \ldots, L/10\}$ and $\mathcal{U}\{L/10 + 1, \ldots, L\}$ with 50% probability each. This results in more training samples at low noise levels which led to a better performance in the extracted force field.

C.4. Alanine Dipeptide

The alanine dipeptide dataset simulated by Köhler et al. (2023) consists of four different simulations of length 500ns with 250K samples each. We partition the data using three simulations for train-validation (750K samples) and one for testing (250K samples). The three training-validation simulations are shuffled and 250K are used for validation and 500K for training.

C.4.1. Implementation details

To evaluate the influence of the dataset size on model performance, we train the model on the following amounts of training data: 10K, 20K, 50K, 100K, 200K and 500K.

We use a neural network with two graph Transformer layers and 96 hidden features per layer. The network is optimized using Adam with a learning rate of $3 \cdot 10^{-4}$ and cosine learning rate decay dropping to $1 \cdot 10^{-5}$. All experiments are run with batch size 1024 and for 1M iterations, with early stopping on the training set sizes 10K and 20K.

For the Langevin dynamics simulation, we use the same simulation settings as in Köhler et al. (2023), except that we do not use parallel tempering. We simulate for 1M iterations, with a time step resolution of 2 femtoseconds, samples are saved

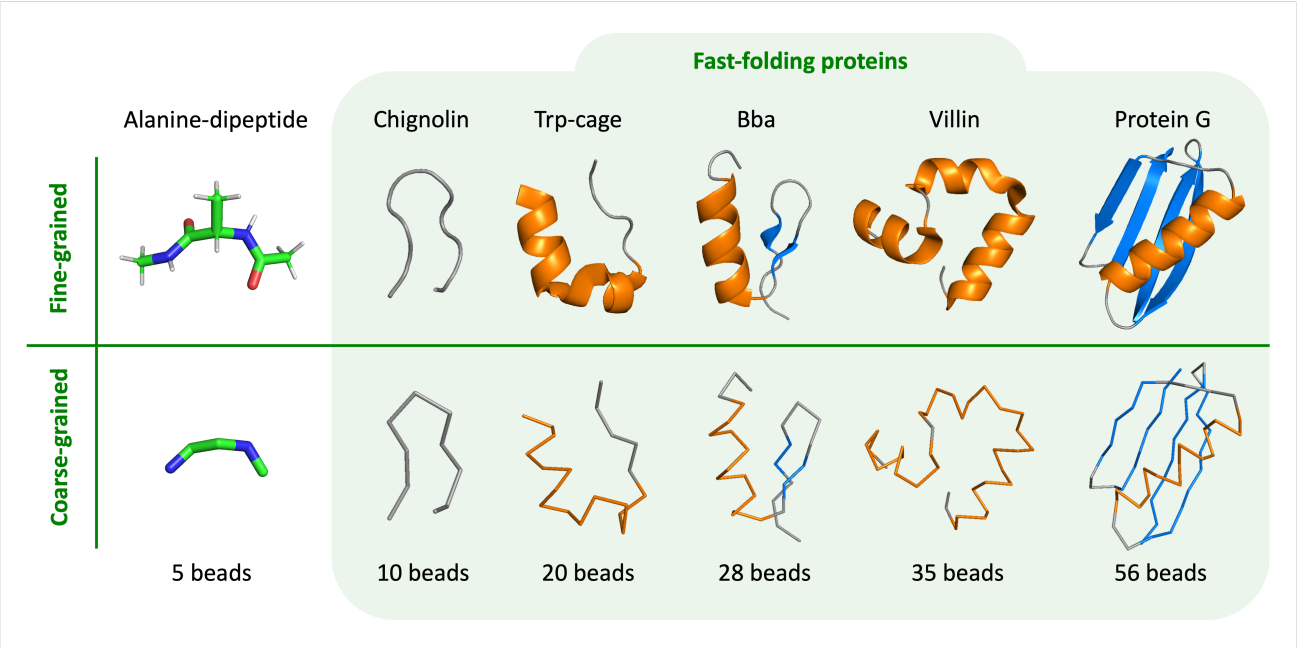


Figure C1. Overview of all systems used in our experiments, consisting of alanine dipeptide and five different fast-folding proteins. Top: fine-grained representation. Bottom: coarse-grained representation, i.e. backbone for alanine-dipeptide and C_α atoms for the fast-folding proteins. α-helices are shown in orange, β-sheets are shown in blue.

every 250 steps resulting in 4K samples per simulation. 100 simulations are run in parallel resulting in a total of 400K samples. The mass of each CG node is set to 12.8 g/mol, which is the weighted average of the mass of carbon and oxygen atoms. The temperature and the friction coefficient are the same as in the reference data (300 Kelvin and 1ps^{-1}).

The noise level i of the denoising diffusion process is obtained by cross-validation. As mentioned in Section 3.1, ideally, with infinite data and network capacity, we would choose i to be the smallest value ($i = 1$). But in practice we cross-validate it, and we find that the smaller the training dataset size, the larger the value of i. The i values obtained for each amount of training samples are the following (indexing from 0 as in the code instead of 1 as in the code):

Table C1. Cross-validated noise levels i across different training set sizes.

Training set size	10K	20K	50K	100K	200K	500K
Noise level i	26	25	20	19	17	8

C.5. Fast-folding Proteins

C.5.1. Dataset details

All fast-folding protein data was obtained from Lindorff-Larsen et al. (2011). The trajectories were randomly split 70-10-20% into a training, validation and test set, using the same seed as Köhler et al. (2023). All trajectories we used contain C_α atoms only, and the interval between frames is 200 picoseconds. Further details can be found in the table below.

C.5.2. Implementation details

In diffusion-denoising process, we used the cosine scheduler with 1000 different noise levels i which is the standard setting in (Ho et al., 2020; Hoogeboom et al., 2022). We used the Adam optimizer. Remaining training hyperparameters are reported in Table C3. Given the training settings from Table C3, the training iterations per second were 17 it/s for chignolin on two V100 GPUs and 13 it/s, 9.8 it/s, 6.5 it/s, 5.9 it/s, for trp-cage, bba, villin and protein G respectively on four V100 GPUs each.

Table C2. Additional dataset details for fast-folding proteins.

	Chignolin	Trp-cage	Bba	Villin	Protein G
ID	CLN025	2JOF	1FME	2F4K	NuG2/1MIO
Temperature (K)	340	290	325	360	350
Amino acids	10	20	28	35	56
Simulation length (μs)	106	208	223	125	369
Data points	534 743	1 044 000	1 114 545	629 907	1 849 251

Table C3. Hyperparameters used in different experiments.

	Chignolin	Trp-cage	Bba	Villin	Protein G
Batch size	512	512	512	512	256
Learning rate	$4 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$4 \cdot 10^{-4}$
Training iterations	1M	1M	2M	2M	3M
Number of layers	3	3	3	3	3
Number of features	64	128	96	128	128
Exponential moving average	.995	.995	.995	.995	.995

For the Langevin dynamics, we ran 6 million steps simulations and saved samples every 500 steps, running 100 simulations in parallel resulted in 1.2M samples. The mass of the particles is set to 12 g/mol which is the mass of each slices Carbon atom, the simulated temperature is the same as in the ground truth data simulation which is reported in Table C2). The noise levels i used for each fast-folder protein indexing from 0 are $\{20, 15, 5, 5, 5\}$ for Chignolin, Trp-cage, Bba, Villin and Protein G respectively.

C.5.3. Additional results

Contact maps As presented in the main text, one way to analyze the global structure of a protein is by evaluating contact maps. Contacts are places within a protein where two atoms are close together, which will occur either when atoms are close in sequence space or when the protein's global fold demands the atoms to be within short distance range. These contacts can be shown as binarized pairwise distance matrices, where only those atoms pairs that are within a certain distance from one another are given value 1 while all other pairs are 0. Contact maps provide rich information about the global conformation of a protein, and they evolve over time during dynamics. Here, we choose to evaluate the distribution over contact maps at equilibrium. We get contact maps for all samples, where a pairwise distance between two atoms smaller than 10Å is considered a contact, and show a normalized count over these contact maps. Note that C_α atoms are typically about 3.8Å apart due to local constraints, which means that the diagonal of the pairwise distance matrix as well as diagonals up until an offset between two and four are always contacts. From Figure C2, it is clear that the DFF i.i.d. method is better at capturing the off-diagonal contacts, especially for larger structures. Even for our largest test case of Protein G, where we have no Flow-CGNet samples to compare, DFF i.i.d. extremely close to the reference. DFF sim. captures the off-diagonal distributions less well compared to the i.i.d. generator, but still improves significantly upon Flow-CGNet.

Chemical integrity In order to show the chemical integrity of our samples, we show statistics over sequence-subsequent and sequence-distant C_α atoms. In the coarse-grained molecules, C_α atoms form the backbone of the protein structure, where we expect the bond length distribution between subsequent atoms to be consistent with the *Reference* data through the simulation. In Figure C3, we compare the ground truth MD distribution to the samples from our *DFF sim.* model. Our model shows the same mean, but a broader variance distribution due to the intrinsic variance introduced by the diffusion noise in $q_i(\mathbf{z})$. This is because *DFF sim.* is actually modelling $q_i(\mathbf{z})$ to approximate the Reference data. To demonstrate that the variance increase in the bonded nodes is caused by the variance in $q_i(\mathbf{z})$ and not by a bond dissociation in the dynamics, we also plot the distribution of the ground truth $q_i(\mathbf{z})$, which we can sample by diffusing the reference MD data $\mathbf{z} \sim q(\mathbf{z}_i \mid \mathbf{z}_0) q(\mathbf{z}_0)$. In Figure C3, we see the bonds distribution approximated by *DFF (sim.)* are the same as the ground truth diffused distribution $q_i(\mathbf{z})$, this means, the dynamics do not diverge into unknown regions that would lead to bond dissociation.

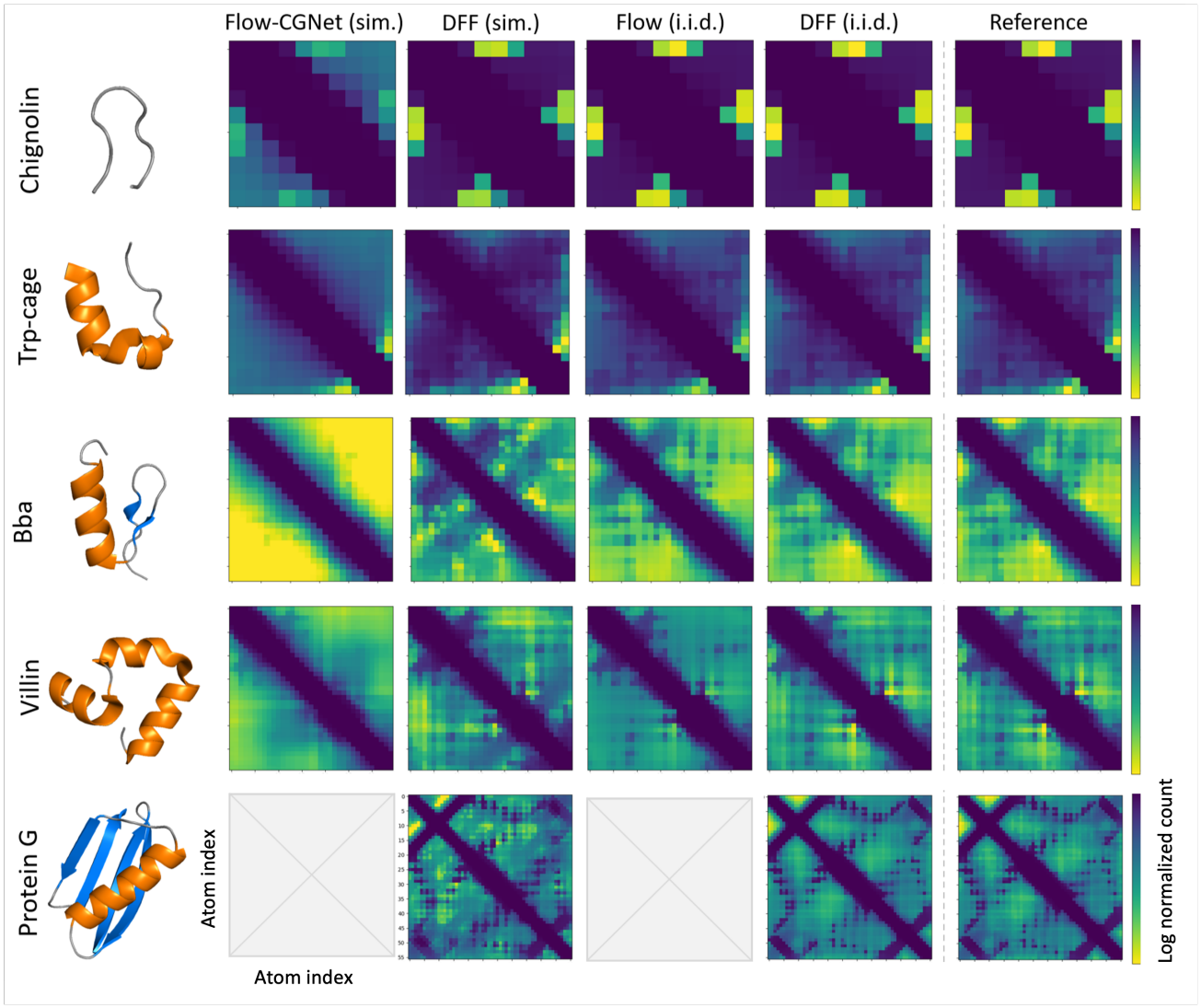


Figure C2. Contact analysis at equilibrium distribution for fast-folding proteins. Each row of results corresponds to one protein. Left: native structure visualization (α-helices in orange, β-sheets in blue), made with PyMOL. Right: log of normalized contact counts over all samples. Red squares indicate region of interest that contains longer-range contacts, crucial toe global structure.

Secondly, in the same Figure C3, we evaluate the small-distance tail of sequence-distant atoms that are three ore more residues apart, corresponding to off-diagonal contact points (see previous paragraph: ”Contact maps”). When these small-distance tails contain values that approach zero, it means van der Waals forces are violated and in the worst case there can be crossings of the backbone with itself. Figure C3 shows that our samples don’t violate chemical integrity.

Transition probabilities using K-means clustering In this paragraph we present more detailed results of transition probability analysis using K-means clustering. Figure C4 depicts the results of k-means clustering in 2D TIC space as well as transition pribability matrices for the reference distribution, the Flow-CGNet model and our own DFF model.

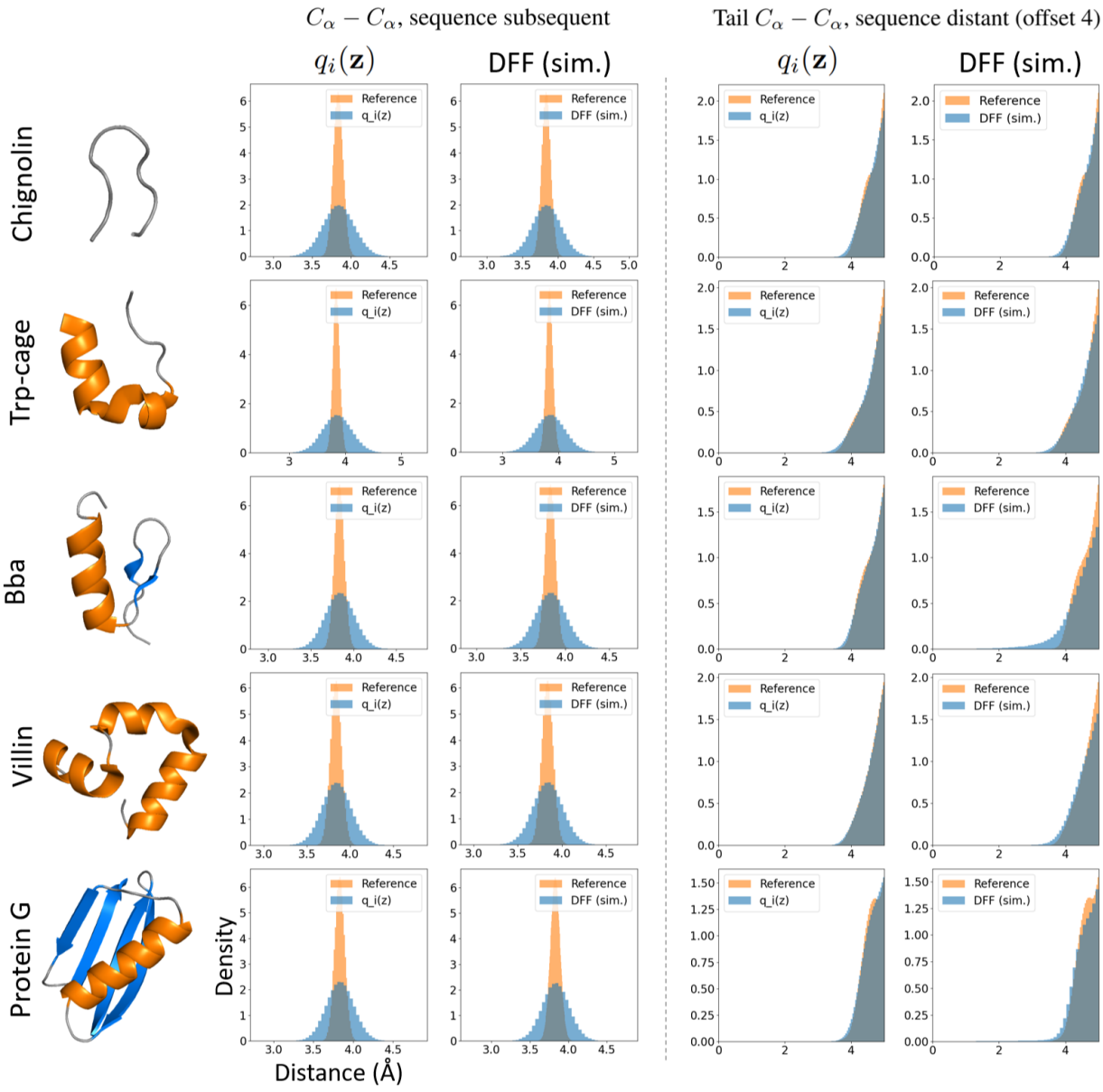


Figure C3. Chemical integrity analysis. Plots in the first two columns display the bond distribution of $q_i(\mathbf{z}_i)$ and *DFF (sim)* w.r.t. to *Reference* Molecular Dynamics. This demonstrates that our model respects the bond distribution of $q_i(\mathbf{z})$ without the dynamics degenerating into unkown regions of space that would lead to bond dissociations. Plots in the right two columns display the small-distance tail of the distribution for sequence-distant atoms, these plots show there are no crossings of the backbone with itself (the distribution doesn't get close to zero).

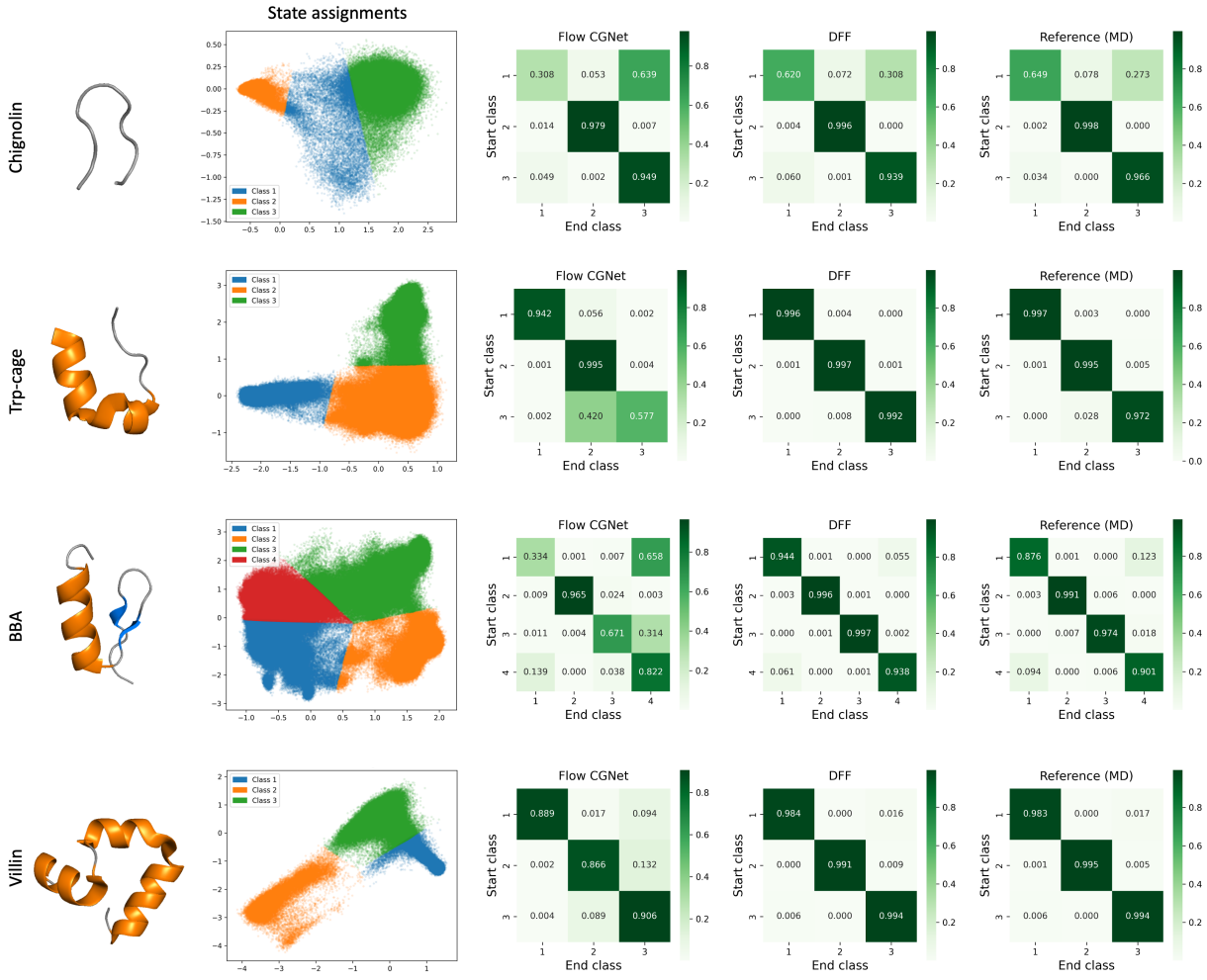


Figure C4. K-means clustering in 2D TIC space for fast-folders (K determined by elbow method) and resulting transition probability matrices for Flow-CGNet, DFF and the reference distribution. Color intensity indicates probability.

Chapter 5

Sampling quality in deep generative models of protein sequences

Marloes Arts*, **Federico Bergamin***, **Wouter Boomsma**, **Jes Frellsen**

The work presented in this chapter was submitted to UAI 2023 and is currently under review.

*Equal contribution

Sampling quality in deep generative models of protein sequences

Marloes Arts[1,*] **Federico Bergamin**[2,*] **Wouter Boomsma**[1] **Jes Frellsen**[2]

[1]Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
[2]Department of Applied Mathematics and Computer Science, Technical University of Denmark, Copenhagen, Denmark
[*]Equal contribution

Abstract

The variational autoencoder has become a common model for analyzing multiple sequence alignments of proteins. Trained on a single protein family, such models can capture the higher-order dependencies between sites in a multiple sequence alignment and have proven useful both for quantifying variant effects and for proposing novel protein sequences in the context of protein engineering. In this paper, we ask how we can improve the sampling quality in these models. One potential weakness of these models is that they are not well-specified, in the sense that the aggregated posterior distribution is dramatically different from the prior. Sampling from the model will therefore produce latent values that are not representative of any training data points and thus depend critically on the extrapolation properties of the decoder. We demonstrate that this problem can be alleviated in a fairly straightforward manner using a hierarchical stochastic structure, giving rise to samples which better capture the covariance between sites. Interestingly, the use of a Bayesian decoder, which has previously been shown to improve accuracy for variant effect prediction, is shown to have a severely detrimental effect on sampling quality, possibly due to limitations of the mean-field approximation.

1 INTRODUCTION

Protein families are collections of protein sequences that are similar in sequence, structure and function. These amino acid sequences are usually presented as a multiple sequence alignment (MSA), where protein sequences are aligned to get same-length sequences with as much overlap as possible, introducing gaps where necessary. Protein families can be regarded as a snapshot of evolution, potentially holding a vast amount of information on protein functionality and the effects of mutations, insertions and deletions [Tatusov et al., 1997]. The rise of modern biological sequencing techniques is accompanied by an increased interest in biological sequence analysis from the field of machine learning [Riesselman et al., 2018, Bepler and Berger, 2019, Alley et al., 2019, Rao et al., 2019, Rives et al., 2021, Shin et al., 2021, Heinzinger et al., 2019, Madani et al., 2020, Elnaggar et al., 2020, Lu et al., 2020, Frazer et al., 2021, Ji et al., 2021, Repecka et al., 2021, Detlefsen et al., 2022].

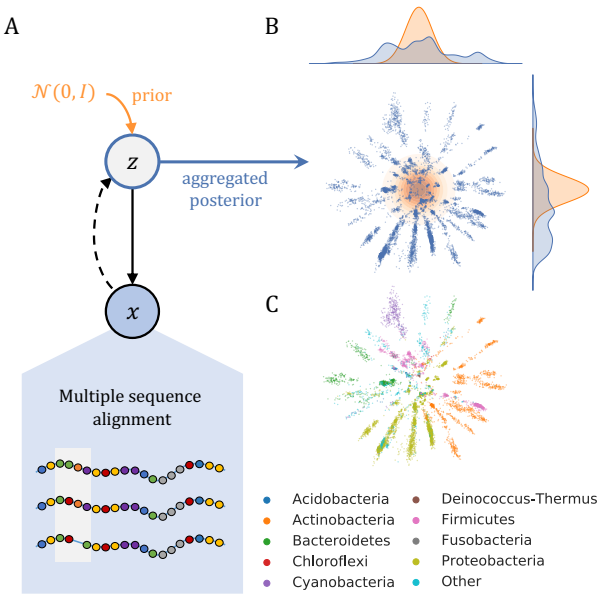


Figure 1: The protein family VAE is not well-specified. A) A VAE with a normal prior can be used for density estimation of MSAs. B) The aggregated posterior distribution differs dramatically from the prior distribution. C) Colors indicate species, which cluster together in latent space.

Deep latent variable models, and in particular variational autoencoders (VAEs) [Kingma and Welling, 2014, Rezende et al., 2014], are commonly used to learn unsupervised rep-

Preprint. Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

resentations from data, and have become a standard choice for analyzing sequences within protein families [Sinai et al., 2017, Riesselman et al., 2018]. A protein family VAE takes multiple sequence alignments (MSAs) as its input and learns a generative model by approximating the posterior distribution over the latent representations, as shown in Figure 1A. Even though these models perform well in terms of variant generation and mutation effect prediction, Figure 1B exemplifies a common problem: the aggregated posterior distribution differs substantially from the prior, i.e. the model is not well specified. We would expect this to cause improper sampling from the prior, which can give practical problems for generating new, stable protein sequences for e.g. protein design. On the other hand, an advantage of this "star-shaped" structure is that different species cluster together in latent space, making the aggregated posterior distribution biologically interpretable (Figure 1C). This begs the question whether it is possible to have a well-specified model that increases performance but at the same time generates a latent space where we can still retrieve meaningful clusters.

To this end, we investigate the effect of using a hierarchical VAE (HVAE) [Rezende et al., 2014, Burda et al., 2016], which has a learnable prior with a hierarchical stochastic latent structure. In a HVAE, the latent representation that is furthest from the output in the generative process can conform to a prespecified prior, commonly a standard Gaussian prior, while the distribution of the lower layers is learned to increase performance. We adopt a ladder VAE (LVAE) architecture [Sønderby et al., 2016], where the inference model consists of a deterministic upward pass and a stochastic downward pass with connections between the two passes, and downward pass parameters are shared with the generative model except for the final decoder step. The resulting inference scheme and generative scheme of the LVAE is depicted in Figure 2 along the usual bottom-up hierarchical VAE. We examine if the added flexibility of a learnable prior aids in obtaining a better specified model with higher sample quality, which would be a valuable trait in protein engineering applications.

In this paper, we investigate strategies for improving the sample quality of deep generative protein sequence models, exploring several variations of VAEs and LVAEs. Our main contributions are:

- We demonstrate that while the use of a Bayesian decoder is beneficial for obtaining high performance in fitness prediction, these models fail to generate meaningful samples. A likely culprit is the independence assumption underlying mean field variational inference. We investigate whether tempering the posterior mitigates this issue, but find that cooling of the posterior does in not lead to an increase in sampling quality – suggesting that the variational approximation finds a different minimum in the landscape than the maximum likelihood solution.
- We show that LVAEs alleviate the mismatch between the prior and the aggregated posterior, giving rise to improved sampling quality in the non-Bayesian decoder setting when the number of layers is increased, while preserving the interpretable "star-shape" in 2D latent space. However, this increase in sampling quality is not observed in the Bayesian setting.
- We find that fitness prediction, measured in terms of Spearman correlation between predicted and experimentally determined scores, is to a large extent influenced by how the marginal log-likelihood is computed. Surprisingly, the common approach of using importance sampling to obtain more accurate estimates of the marginal likelihood leads to correlation scores worse than simple Monte Carlo sampling (which merely reduces the variance of the lower bound on the marginal likelihood).

2 RELATED WORK

Multiple sequence alignments have been an omnipresent ingredient in bioinformatics methods over the last decades. Initially, they were primarily used to extract site-specific statistics based on the frequency of amino acids occurring at each position. A decade ago, it became clear that pairwise correlations could be exploited to infer structural properties from such alignments, using energy models like the Potts model [Marks et al., 2011, Morcos et al., 2011, Balakrishnan et al., 2011]. While going beyond pairwise correlations is not tractable through direct enumeration, the DeepSequence protein family variational autoencoder demonstrated that its latent structure allowed for further gains by exploiting higher order effects [Riesselman et al., 2018]. This model was shown to provide both state-of-the art prediction of variant effects in Deep Mutational Scan experiments, and was later also shown to be useful for disease variant prediction (the EVE model [Frazer et al., 2021]). Even with the recent developments in language modelling for protein sequences, variational autoencoder remain the method of choice for variant effect prediction when sufficiently many sequences are available [Meier et al., 2021, Hesslow et al., 2022]. Also for the generation of new protein sequences the variational autoencoder has been successful, producing experimentally verified functional proteins [Hawkins-Hooker et al., 2021]. Finally, although care should be taken when interpreting the latent spaces of generative models, the original protein family VAE demonstrated a tree-like structure in latent space, which reflected the phylogeny of the sequences in the alignment [Riesselman et al., 2018]. This result was later expanded upon in [Ding et al., 2019], and recently analyzed in detail, demonstrating how latent spaces can give rise to meaningful distances and interpolation [Detlefsen et al., 2022].

In the image analysis community, the variational autoen-

coder is an attractive model primarily for its generative purposes. In this setting it was originally found to generate more blurry images than methods like Generative Adverserial Networks (GANs). However, in recent years a number of methodological advances have made the variational autoencoder fully competitive both in terms of generation quality and log-likelihood on natural image benchmarks. Most of the advances involved either creating more flexible and expressive variational distributions to better approximate the true posterior or using more expressive priors rather then the usual diagonal Gaussian. These can be interpreted as different strategies to tackle the mismatch between the learned aggregated posterior and the prior in VAEs [Rosca et al., 2018, Hoffman and Johnson, 2016], i.e. latent space regions where the prior is assigning high density but which have low density under the aggregated posterior. Tomczak and Welling [2018] suggested approximating the aggregated posterior as a prior by fitting a mixture model with fixed components on a posterior computed over learned pseudo-inputs. Other approaches, instead, involve reweighting the standard Gaussian prior, by either using rejection sampling with a learned acceptance function [Bauer and Mnih, 2019] or using an energy-based model prior [Aneja et al., 2021]. Flexibility can also be gained by parameterizing the prior using normalizing-flows [Chen et al., 2017] or autoregressive models [Gulrajani et al., 2017]. However, most of the recent success of VAEs is due to architecture design choices and tricks that allow to train deep hierarchical VAE [Burda et al., 2016, Sønderby et al., 2016] with a large number of stochastic layers [Maaløe et al., 2019, Vahdat and Kautz, 2020, Child, 2021]. By defining a conditional hierarchy of latent variables, deep hierarchical VAEs have a more expressive prior and a more flexible posterior approximation at the same time, and are able to tackle the mismatch problem between the prior and the aggregated posterior by tuning both.

Recently, denoising diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020] have gained popularity for their generative capabilities in the image domain. However, they are less established in the context of biological sequence modelling, and need additional tweaking to extract the likelihood [Song et al., 2021]. Since we require evaluations of likelihoods, we opted for pursuing the VAE approach here.

3 PROTEIN FAMILY LADDER VARIATIONAL AUTOENCODERS

We start by reviewing VAEs, their extension to hierarchical architectures and the specific design of the ladder VAE to make robust training of these models feasible. Finally, we consider Bayesian VAEs, where we learn a distribution over the decoder parameters.

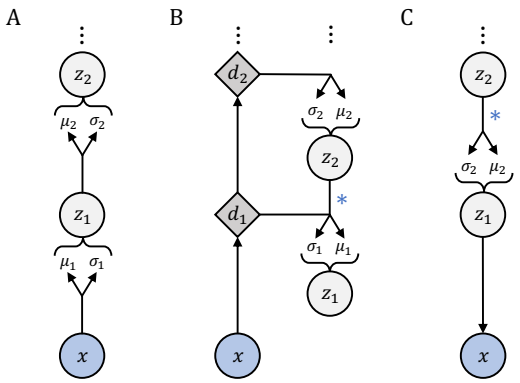


Figure 2: A) HVAE bottom-up inference model, where x is the input, arrows indicate neural networks parameterizing a mean and a variance, and z are sampled latent variables. B) LVAE top-down inference model, where deterministic variables d are combined with stochastic variables z in order to get new mean and variance values to sample the next latent variable from. C) The generative model is the same both for HVAE and LVAE, apart from the shared weights (indicated by blue stars).

3.1 VARIATIONAL AUTOENCODER

The variational autoencoder is a framework to train deep latent variable models $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ where the assumption is that an observed variable $\mathbf{x}$ is generated by a transformation of an unobserved variable $\mathbf{z}$. The prior distribution $p(\mathbf{z})$ over the latent variables can be learned, as highlighted in the Related Works section, but common choice is to consider a standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ is a parametric family of distributions over the input space parametrized by a neural network with parameters $\boldsymbol{\theta}$. The true posterior $p(\mathbf{z}|\mathbf{x})$ is intractable, making it impossible to optimize the model parameters by maximizing the log-likelihood directly. We rely on a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$, also parameterized by a neural network, to approximate the true posterior. The model parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are jointly learned by maximizing the evidence lower bound, $\mathcal{L}(\boldsymbol{\theta}, \phi)$, on the log-likelihood (ELBO) defined as:

$$\log p_\theta(\mathbf{x}) \geq \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{=:\mathcal{L}(\boldsymbol{\theta},\phi)},$$

where KL is the Kullback-Leibler divergence.

3.2 HIERARCHICAL VAE AND LADDER VAE

The regular VAE is defined with a single (n-dimensional) latent variable, which limits the VAE's ability to model complex input distributions. Hierarchical VAEs extend

the VAE framework by considering a hierarchical prior, i.e. a hierarchy of L latent variables instead of a single one. This structure has the benefit of increasing the expressivity of the prior and the approximate posterior. The new prior $p(\mathbf{z})$ is defined as $p(\mathbf{z}) = \prod_{i=1}^{L-1} p(\mathbf{z}_i|\mathbf{z}_{i+1})$, and therefore the top-down generative process becomes $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z}_1)p(\mathbf{z}_1|\mathbf{z}_2)\cdots p(\mathbf{z}_{L-1}|\mathbf{z}_L)$. For the inference model we can either follow a bottom-up approach Burda et al. [2016] (HVAE), where the inference is defined as

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = q_\phi(\mathbf{z}_1 \mid \mathbf{x}) \prod_{i=2}^{L} q_\phi(\mathbf{z}_i \mid \mathbf{z}_{i-1}), \tag{1}$$

or a top-down Sønderby et al. [2016] (LVAE), where the inference is defined as

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = q_\phi(\mathbf{z}_L \mid \mathbf{x}) \prod_{i=L-1}^{1} q_\phi(\mathbf{z}_i \mid \mathbf{z}_{i+1}). \tag{2}$$

In the top-down inference of the LVAE, weights can be shared between the encoder and decoder (Figure 2). Optimization of model parameters for both cases is done by maximizing the evidence lower bound.

Training deep HVAEs is known to become unstable as we increase the number of layers, due to the introduction of more and more conditional stochasticity [Vahdat and Kautz, 2020]. In addition, due to their bottom-up inference model, latent variables in the upper stochastic layers of HVAEs tend to become inactive, meaning that they are not contributing in learning a meaningful representation [Burda et al., 2016]. The top-down inference was introduced as a solution for exactly this problem [Sønderby et al., 2016]. Another possible approach to avoid the problem of upper latent variables being inactive is the use of skip connections [Maaløe et al., 2019, Vahdat and Kautz, 2020]. In this work we consider only hierarchical VAEs with top-down inference as defined in Sønderby et al. [2016].

3.3 BAYESIAN VAE

Both DeepSequence [Riesselman et al., 2018] and EVE [Frazer et al., 2021] use a Bayesian decoder for their VAE model. This is motivated by the improvements in performance when predicting stability of protein mutations for different families. In the usual VAE notation, which we also adopted, $p_\theta(\mathbf{x}|\mathbf{z})$ means $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta} = \boldsymbol{\theta}_{\text{MLE}})$, where $\boldsymbol{\theta}_{\text{MLE}}$ is just a single point estimate learned during training. In a Bayesian VAE, one assumes a distribution over the decoder parameters $\boldsymbol{\theta}$ and the overall goal is to estimate the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. A possible approach to approximate the posterior distribution is to optimize a variational approximation of the posterior by minimizing the KL between the variational distribution $q(\boldsymbol{\theta}|\lambda)$ and the prior distribution $p(\boldsymbol{\theta})$ during training using gradient descent. In the literature this is also known as "Bayes by Backprop" [Blundell et al., 2015], which was also used by [Riesselman et al., 2018, Frazer et al., 2021] to train their Bayesian VAE. When using this approach, the model is trained by maximizing:

$$\log p_\theta(\mathbf{x}) \geq N \cdot \mathbb{E}_{p(x)}[\mathbb{E}_{q(\theta|\lambda)q_\phi(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] - \mathrm{KL}(q(\boldsymbol{\theta}|\lambda)||p(\boldsymbol{\theta})),$$

where the prior distribution is usually assumed to be a standard Gaussian, i.e. $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the variational approximation is assumed to be a diagonal Gaussian $q(\boldsymbol{\theta}|\lambda) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2 I})$, with $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\sigma^2}\}$ being the parameters optimized during training and N being the number of training points.

The choice of employing a Bayesian decoder in a VAE has also seen successful applications in the domain of out-of-distribution detection [Daxberger and Hernández-Lobato, 2019, Glazunov and Zarras, 2022]. In this context, it was shown that potential benefits can be gained by a more comprehensive Bayesian treatment of the weights, using stochastic gradient Hamiltonian Monte Carlo [Chen et al., 2014]. We will only consider only Bayes by Backprop in this study, to allow for direct comparisons to the earlier DeepSequence and Eve models, but will return to the discussion on potential weaknesses of the Bayes by Backprop approach in the Discussion section.

4 EXPERIMENTS

We will investigate the following:

1. Given the success of hierachical VAEs for image generation, can we improve sampling quality in protein sequence VAEs using an architecture of multiple stochastic layers?
2. Does the use of a hierarchical stochastic structure in these VAEs alleviate the mismatch between the prior and the aggregated posterior we see in Fig. 1B?
3. What is the effect of the Bayesian decoder on sample quality and the performance in quantifying variant stability?
4. Is the final Spearman correlation between predicted and experimentally determined variant effects influenced by the way we compute the marginal log-likelihood?

To answer these questions, we start by considering the EVE model [Frazer et al., 2021], which is the most commonly used VAE for modelling amino acid sequences for protein families. This is a one-stochastic layer VAE with a Bayesian decoder, which also considers a 1D convolution to encourage correlation between amino acids and temperature scaling of the output logits. An important difference between the EVE model we used in this paper and the original EVE implementation is that we decide to model sequence gaps in the alignments, while in the original model these are masked

during training. This means that unlike DeepSequence and EVE, our models are able to generate sequences with gaps, which is of critical importance for reliable sequence generation.

We consider possible modifications that can improve EVE architecture by making it more flexible. We investigate hierarchical VAEs with two and three stochastic layers and for all models we consider two equivalent architectures with and without a Bayesian decoder. A full detailed description of these models and the training setup is available in the Supplementary Material (section A)[1].

Protein family data We test our hypotheses on the the beta-lactamase protein family multiple sequence alignment dataset made available by Riesselman et al. and labeled by species, as well as the corresponding deep mutational scan dataset [Stiffler et al., 2015] as a test set for calculating mutant effect prediction (Spearman correlation). As shown in [Riesselman et al., 2018], we use the log-ratio of the lower bound on the marginal likelihood as a proxy for sequence variant fitness. During training, sequences are reweighed to reduce sampling bias, using a 80% identity cutoff [Riesselman et al., 2018, Ekeberg et al., 2013]. For the hold out set, we randomly pick 5% of the training set and remove all sequences from the training set that have more than 80% sequence similarity with the hold out set. After processing, there are 4412 sequences in the training set and 388 in the hold out set.

4.1 PROTEIN FAMILY VAE SAMPLE QUALITY

To demonstrate the effect of the mismatch between the prior and the aggregated posterior for protein family VAE models, we assess the quality of generated samples. This was done by drawing samples from an EVE model, i.e. a 1-layer VAE with a Bayesian decoder, to obtain a "pseudo-alignment" with the same number of sequences as the input MSA that the model was trained on. We use EVcouplings [Hopf et al., 2019] to predict a precision-ranked list of contacts, i.e. sites where atoms are predicted to be in close spatial proximity, as shown in Figure 3A. Subsequently, we customarily take the average precision over the top L, top L/2 and top L/4 contacts to obtain a sample quality score, where L is the length of the protein (which is 263 amino acids for beta-lactamase). Figure 3B reports the sample quality for the input MSA, a site-independent model and different variants of EVE. Top L, top L/2 and top L/4 all follow a similar pattern, where top L/n quality scores are generally higher for larger values of n since the top-to-bottom decrease in precision-ranked contacts is not linear. The MSA serves as a reference value, providing a loose upper sample quality limit that can be obtained based on the input data. The site independent model is a simple model that assumes each residue position to

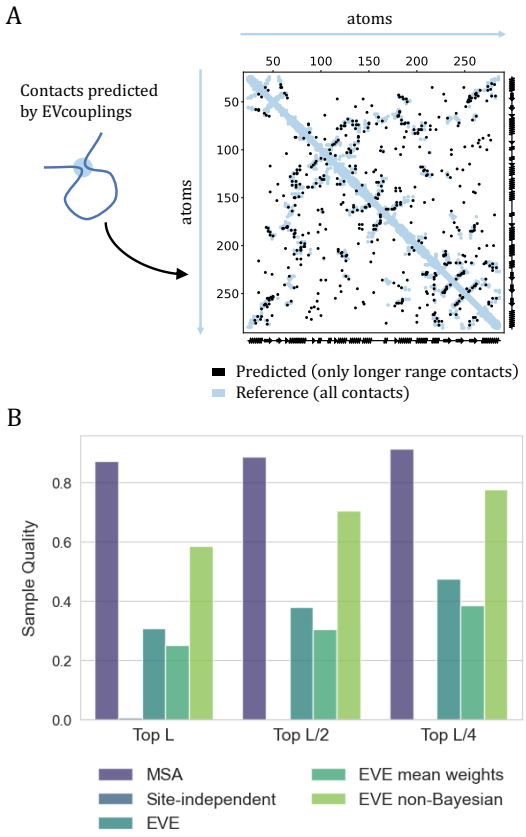


Figure 3: A) Example of a contact map that can be predicted using evolutionary couplings, compared to a reference map. Long-range predicted contacts in black, reference in blue. Secondary structure elements are shown in black on the right and bottom of the contact map. B) Sample quality for the input MSA, a site-independent model, and different variants of EVE (a 1-layer VAE with a Bayesian decoder).

be independent, where we use the input MSA to get all positional amino acid distributions. Since our experiment focuses on covariation, and ignores conservation of amino acids, the sample-quality of the site-independent model is close to zero. For EVE, we consider three different variants. The first one is the original EVE model, where at sampling time we sample both the decoder weights $\boldsymbol{\theta}^{(i)} \sim q(\boldsymbol{\theta}|\lambda)$ and a latent representation $\boldsymbol{z}^{(i)} \sim p(\boldsymbol{z})$ from the prior before decoding the latent $\boldsymbol{z}^{(i)}$. Even though this model has been shown to have good performance in the context of disease variant prediction [Frazer et al., 2021], the sample quality of generated protein sequence samples is still relatively low. Freezing the weights of the Bayesian decoder to be equal to the mean (MAP estimate) yields similar, but slightly lower, sample quality. In contrast, replacing the Bayesian decoder by a non-Bayesian variant gives a boost in sample quality. This indicates that even though the Bayesian decoder is useful for improving stability predictions, this modelling choice can be detrimental for generated samples.

[1]Code will be made available on Github upon publication.

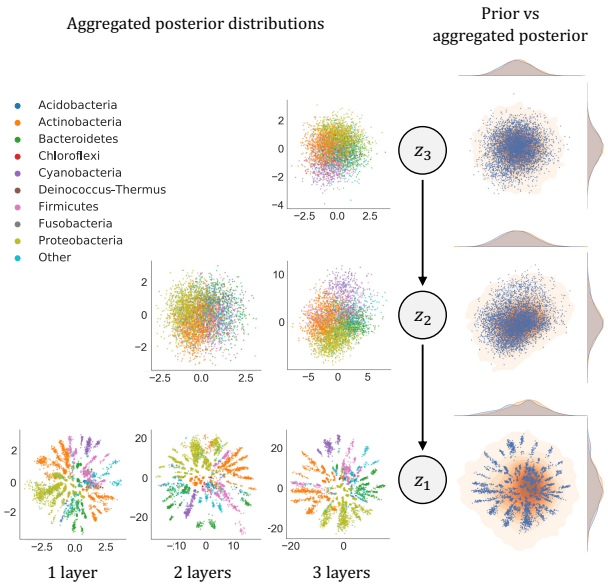


Figure 4: 2D latent representations. Left: different levels of aggregated posterior distributions for a model with 1 to 3 hierarchical layers, color coded by species. Right: comparison between the prior (kernel density estimate, orange) and the aggregated posterior (scatter, dark blue) for the 3 layer model, with kernel density estimations of marginal distributions on the marginal axes.

4.2 THE HIERARCHICAL VAE AS A BETTER SPECIFIED MODEL?

We hypothesize that one of the main factors explaining low sample quality in protein family VAEs is the inconsistency between the prior and the learned aggregated posterior, which is depicted in Figure 1B. These inconsistencies are reflected in differences in structure and overall scale of the two distributions. Here, we visually investigate if hierarchical VAEs can overcome this mismatch. We explore the learned 2D latent space for 1-, 2-, and 3-layer VAEs trained with two latent variables without a Bayesian decoder, since the non-Bayesian decoder scored the highest in terms of sample quality in Figure 3B. The left-hand side of Figure 4 shows the latent space for these three models, color coded by species. For the 2- and 3-layer hierarchical models, the top layers furthest away from the output ($\boldsymbol{z}_2$ and $\boldsymbol{z}_3$, respectively) are much closer to the Gaussian prior, while preserving the interpretable star-shaped structure close to the output ($\boldsymbol{z}_1$). On the right-hand side of Figure 4 demonstrates how well the aggregated posterior overlaps with the prior for the 3-layer model. At each layer, the prior matches the aggregated posterior quite well in terms of scale and overall shape. Note that we show one sample for each data point in the aggregated posterior, which does not reflect the fluctuations each point has. We show the effect of using more samples in the Supplementary Material (section B).

However, even for many samples, there are still regions of latent space where the prior assigns some probability mass even though these regions are empty for the aggregated posterior. Despite this, hierarchical VAEs seem to be better, although not perfectly, specified models for generating protein sequences.

4.3 THE EFFECT OF WEIGHT DISTRIBUTION TEMPERATURE DURING SAMPLING

Since VAEs with a non-Bayesian decoder seem to perform relatively well in terms of sampling quality, we investigate whether or not sample quality improves for VAE models with Bayesian decoders when the temperature of the distribution over decoder weights is turned down at sampling time. Here, a temperature of 1 corresponds to regular weight sampling from the Bayesian decoder before sampling sequences, and zero temperature corresponds to sampling using the mean weights of the decoder (MAP estimate). We evaluate sample quality as well as Spearman correlation with mutation effects at six different temperatures: 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0, as reported in Figure 5. Alongside 1-, 2-, and 3-layer Bayesian decoder VAEs at this range of temperatures, we also report the temperature-independent performance of the non-Bayesian variants and, in case of sample quality, the input MSA. Figure 5A shows that decreasing the temperature for decoder weight sampling does not have a substantial effect on sampling quality across hierarchical models. Moreover, the sampling quality of the Bayesian decoder models is still far below their non-Bayesian counterparts, which show some slight improvement upon adding more layers to the hierarchical model. For the Spearman correlation for mutation effect prediction, Figure 5B shows an opposite trend where the Bayesian variants have a much higher correlation compared to the non-Bayesian variants. Adding more layers to the non-Bayesian decoder VAE even seems detrimental to this specific downstream task, despite having much higher sampling quality. Finally, even though the Spearman correlation hardly seems to be influenced by decoder weight sampling temperature for the 1- and 3-layer models, the 2-layer model clearly shows a downward trend when lowering the temperature.

4.4 SENSITIVITY OF SPEARMAN CORRELATION TO THE NUMBER OF SAMPLES

We use the probability log-ratio as a proxy for the fitness of a specific mutation. This is defined as $\log \frac{p(\mathbf{x}^{(\text{mutant})}|\boldsymbol{\theta})}{p(\mathbf{x}^{(\text{wild-type})}|\boldsymbol{\theta})} = \log p(\mathbf{x}^{(\text{mutant})}|\boldsymbol{\theta}) - \log p(\mathbf{x}^{(\text{wild-type})}|\boldsymbol{\theta})$. In the context of a VAE, a common way to approximate the marginal log-likelihood is by importance sampling, as initially proposed by Rezende et al. [2014], which provides a tighter bound on the marginal log-likelihood. In contrast, directly using the ELBO gives a looser approximation of the marginal log-

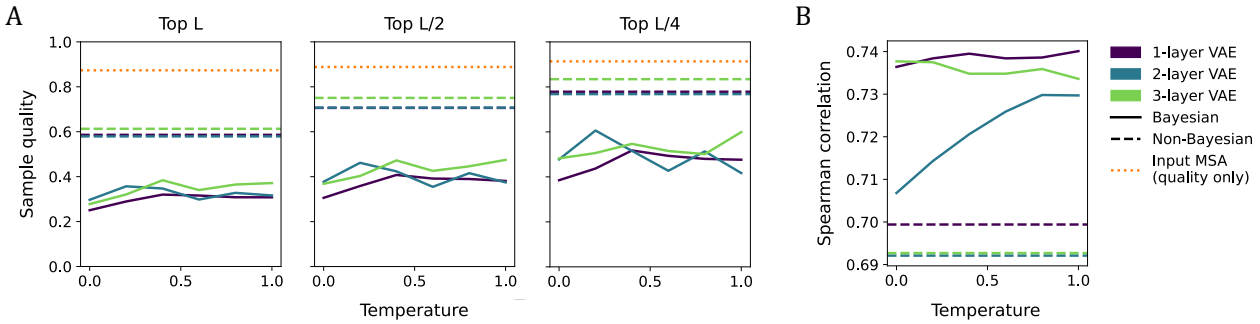


Figure 5: Sample quality (A) and Spearman correlation for mutation effect prediction (B) for hierarchical models at different decoder weight sampling temperatures. Dashed lines show non-Bayesian temperature-independent model variants, dotted orange line shows input MSA sample quality. Precise numerical values can be found in the Supplementary Material (section D), along with results for hierarchical models with a smaller architecture (section C).

likelihood. Indeed, the former is an actual approximation of the marginal likelihood, while the latter is just a lower bound. We can write the approximation of the marginal log-likelihood as Burda et al. [2016]:

$$\log p(\mathbf{x}) \geq \mathcal{L}(\mathbf{x}; \boldsymbol{\theta}, \phi) \approx \frac{1}{M} \sum_{m=1}^{M} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}|\mathbf{z}^{m,k}) p(\mathbf{z}^{m,k})}{q_\phi(\mathbf{z}^{m,k}|\mathbf{x})} \right]$$

where $\mathbf{z}^{1,1}, \ldots, \mathbf{z}^{M,K} \sim q_\phi(\mathbf{z}|\mathbf{x})$. If in the above equation we consider the number of importance samples $K = 1$, then this corresponds to the usual VAE ELBO, which can be estimated using M Monte Carlo (MC) samples. If, instead, $K > 1$, then we get the importance weighted autoencoder (IWAE) bound from Burda et al. [2016], where we usually set $M = 1$. The first approach results in a low variance estimate as M goes to infinity, while the second results in a lower-variance as well as a tighter bound as K increases.

We investigate how the choice of both the type and number of samples influence the final Spearman correlation. We consider the EVE model as described in Frazer et al. [2021] and compare its performance to a corresponding non-Bayesian variant in terms of Spearman correlation. We consider two approaches to estimate the log-likelihood; in the first approach, we keep $K = 1$ and vary the number of MC samples, while in the second approach we do the opposite. From Figure 6, we can see that increasing the number of MC samples leads to a larger gain in performance compared to increasing the number of IWAE samples. This is true both in the Bayesian and non-Bayesian setting, although the fitness prediction gap between the two approaches is smaller for the model without the Bayesian decoder.

5 DISCUSSION

Over the last years, VAEs have become popular tools for characterizing mutations within protein families. In particular, they display unmatched performance in prediction of variant effects and their tree-like latent spaces have been observed to contain biologically relevant information. However, in the context of the VAE as a generative model, the structured latent space has been a source of concern, since it is at odds with the prior distribution. This means that sampling from the prior can result in z-values that are are not representative for any of the training data. Moreover, if the prior does not cover the same space as the aggregated posterior, this might influence the diversity of the samples. One might therefore expect that sampling quality could be improved considerably if this problem was addressed. In this paper, we set out to investigate how substantial this problem is in practice, what the effects are on protein fitness prediction and how to mitigate the mismatch between the prior and the aggregated posterior to improve sample quality. We demonstrate that hierarchical VAEs increase the flexibility of the prior distribution to match that of the structured latent space, which leads to an improvement in terms of sample quality in the non-Bayesian decoder setting.

Although we see some gains in sampling quality in the hierarchical non-Bayesian setting, the improvements are less dramatic than we had expected, especially in the Bayesian context. We believe the explanation to be related to the extrapolation properties of neural networks. In areas of the latent space which are unsupported by data (off-manifold), the decoder will map to similar sequences as the nearest latent point on the boundary of the manifold, and do so with high certainty [Detlefsen et al., 2022]. Thus, if we take an off-manifold latent point, decode it, and subsequently encode it back to the latent space, we effectively project the point onto the manifold. Although this is a problematic property for uncertainty quantification, it appears to be beneficial for the generative aspects of the model, since off-manifold latent points will still produce reasonable samples. One would, however, expect that this introduces a bias in the sampled distribution, implicitly upweighting the boundary region. This aspect of sampling quality was not tested in our experiments, and it is not obvious how one would quantify this effect in the high dimensional space of protein sequences.

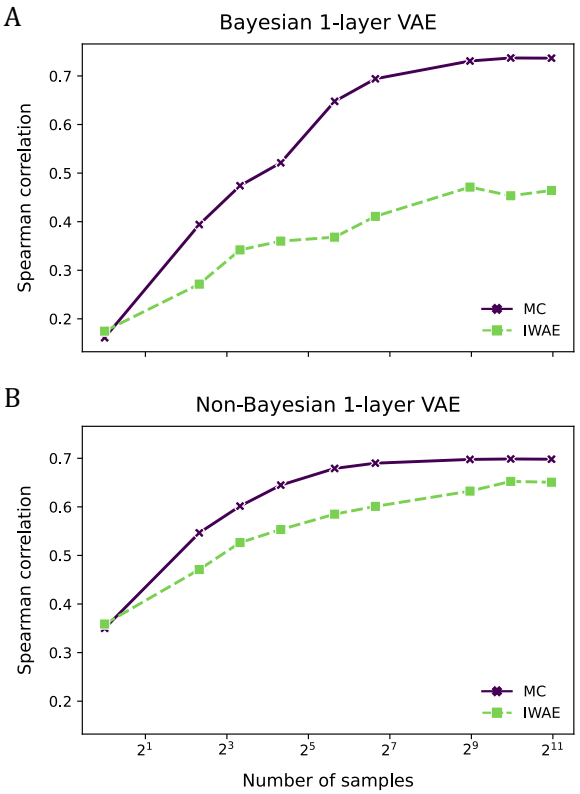


Figure 6: Evolution of the Spearman correlation with respect to the type and the number of samples used to approximate the log-likelihood. A) Spearman for the original EVE model as described in Frazer et al. [2021] and B) non-Bayesian version of the EVE. Using MC samples is always giving better results, but the difference between the MC and IWAE approach is smaller in the non-Bayesian case.

One surprising result of our study was the fact that we saw no improvement in fitness prediction (Spearman correlation to experiment) when we consider hierarchical VAEs, despite the fact that the model is better specified and produces higher quality samples. Likewise, we saw no improvement when considering improving our estimate of the marginal log-likelihood. While unexpected, these observations is in tune with a recent study considering the relationship between density estimation and fitness estimation, in which it was concluded that we cannot generally assume these two objectives to align [Weinstein et al., 2022]. In fact, in some cases, misspecification of a density model was shown to be beneficial for fitness estimation. This observation is complicated further by the fact that in general, depending on the assay, the experimentally measured quantity might also not be identical to the function optimized by evolution (although this alignment should be strong for the beta-lactamase assay studied here). For the purpose of variant effect prediction, we should therefore consider optimizing hyperparameters of the models on (hold-out sets of) the experimental data directly.

The hierarchical VAEs presented here do not fully solve the prior-aggregated posterior mismatch, in the sense that they still assign some probability mass in areas of latent space which do not correspond to data observed during training. Further extensions of the expressivity of the prior could resolve this issue, e.g. by using a normalizing flows or diffusion-based priors [Chen et al., 2017]. However, these changes would typically involve an increase in the number of parameters, which is potentially problematic in the setting where these models are to be estimated on single protein families consisting of only hundreds to thousands of sequences. Likewise, training issues are known to arise in these settings due to the competition between the conditional log-likelihood and the KL term in the VAE ELBO. However, the spectacular progress observed for hierarchical VAEs in image generation in recent years suggests that a hierarchical approach might provide benefits not attainable by such priors. Another approach is to use even more informative priors. One example is the recent use of an Ornstein-Uhlenbeck process to provide a tree-structure the latent space [Moreta et al., 2022] of a variational autoencoder. Currently, this latter approach is however limited to multiple sequence alignments with a low number of sequences, and requires a precomputed phylogenetic tree, making it less of a general purpose solution.

Another open problem we highlight in this paper is that Bayesian VAEs generate lower-quality samples compared to an equivalent non-Bayesian model. In addition to that, the temperature scaling experiments reveal that the minimum found by maximum likelihood and VI yield to very different results in terms of quality of the generated sequences. We believe that mean-field approximation can be the cause of this behaviour, and modelling the correlation between weights either by doing full (or block) covariance Laplace or Hamiltonian Monte Carlo would be a promising approach to investigate in future works. However, modelling the full covariance over decoder weights is challenging due to constrictions in memory and compute power. As an example, if we consider the beta-lactamase protein family from the experiments section, we have to model a 263-long sequence with an alphabet of 21. Assuming a last-layer with 100 hidden units, this would results in a $\approx 500000 \times 500000$ covariance matrix. Therefore, although the recent advances in Laplace approximation [Daxberger et al., 2021], especially last-layer Laplace approximation, it will be difficult to estimate the posterior distribution over the decoder weights in a post-hoc way. An alternative used by earlier studies in out-of-distribution detection is to use Hamiltonian Monte Carlo for estimating the weight distributions. This replaces the need for storing a full covariance matrix with a simple registration of weight samples, but at the price of a computationally more involved training procedure. We leave a full exploration of these tradeoff as future work.

6 ACKNOWLEDGEMENTS

The work conducted within the Center for Basic Machine Learning Research in Life Science (MLLS, grant nr NNF20OC0062606). MA was supported by the Novo Nordisk Foundation (grant nr NNF18OC0052719) and FB by the Innovation Fund Denmark (grant nr 0175-00014B).

References

E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322, 2019.

J. Aneja, A. Schwing, J. Kautz, and A. Vahdat. A contrastive learning approach for training variational autoencoder priors. *Advances in Neural Information Processing Systems*, 34, 2021.

S. Balakrishnan, H. Kamisetty, J. G. Carbonell, S.-I. Lee, and C. J. Langmead. Learning generative models for protein fold families. *Proteins: Structure, Function, and Bioinformatics*, 79(4):1061–1078, 2011.

M. Bauer and A. Mnih. Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 66–75. PMLR, 2019.

T. Bepler and B. Berger. Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*, 2019.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. In *5th International Conference on Learning Representations, ICLR*, 2017.

R. Child. Very deep vaes generalize autoregressive models and can outperform them on images. In *9th International Conference on Learning Representations, ICLR*, 2021.

E. Daxberger and J. M. Hernández-Lobato. Bayesian variational autoencoders for unsupervised out-of-distribution detection. *arXiv preprint arXiv:1912.05651*, 2019.

E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

N. S. Detlefsen, S. Hauberg, and W. Boomsma. Learning meaningful representations of protein sequences. *Nature communications*, 13(1):1–12, 2022.

X. Ding, Z. Zou, and C. L. Brooks III. Deciphering protein evolution and fitness landscapes with latent space models. *Nature communications*, 10(1):1–13, 2019.

M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, and E. Aurell. Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. *Physical Review E*, 87 (1):012707, 2013.

A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, et al. Prottrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing. *arXiv preprint arXiv:2007.06225*, 2020.

J. Frazer, P. Notin, M. Dias, A. Gomez, J. K. Min, K. Brock, Y. Gal, and D. S. Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 599 (7883):91–95, 2021.

M. Glazunov and A. Zarras. Do bayesian variational autoencoders know what they don't know? In *Uncertainty in Artificial Intelligence*, pages 718–727. PMLR, 2022.

I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taïga, F. Visin, D. Vázquez, and A. C. Courville. Pixelvae: A latent variable model for natural images. In *5th International Conference on Learning Representations, ICLR*, 2017.

A. Hawkins-Hooker, F. Depardieu, S. Baur, G. Couairon, A. Chen, and D. Bikard. Generating functional protein variants with variational autoencoders. *PLoS computational biology*, 17(2):e1008736, 2021.

M. Heinzinger, A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes, and B. Rost. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC bioinformatics*, 20(1):1–17, 2019.

D. Hesslow, N. Zanichelli, P. Notin, I. Poli, and D. Marks. Rita: a study on scaling up generative protein sequence models. *arXiv preprint arXiv:2205.05789*, 2022.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

M. D. Hoffman and M. J. Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, 2016.

T. A. Hopf, A. G. Green, B. Schubert, S. Mersmann, C. P. Schärfe, J. B. Ingraham, A. Toth-Petroczy, K. Brock, A. J. Riesselman, P. Palmedo, et al. The evcouplings python framework for coevolutionary sequence analysis. *Bioinformatics*, 35(9):1582–1584, 2019.

Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

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

A. X. Lu, H. Zhang, M. Ghassemi, and A. Moses. Self-supervised contrastive learning of protein representations by mutual information maximization. *BioRxiv*, 2020.

L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. Biva: A very deep hierarchy of latent variables for generative modeling. *Advances in neural information processing systems*, 32, 2019.

A. Madani, B. McCann, N. Naik, N. S. Keskar, N. Anand, R. R. Eguchi, P.-S. Huang, and R. Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

D. S. Marks, L. J. Colwell, R. Sheridan, T. A. Hopf, A. Pagnani, R. Zecchina, and C. Sander. Protein 3d structure computed from evolutionary sequence variation. *PloS one*, 6(12):e28766, 2011.

J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in Neural Information Processing Systems*, 34, 2021.

F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108 (49):E1293–E1301, 2011.

L. S. Moreta, O. Rønning, A. S. Al-Sibahi, J. Hein, D. Theobald, and T. Hamelryck. Ancestral protein sequence reconstruction using a tree-structured ornstein-uhlenbeck variational autoencoder. In *International Conference on Learning Representations*, 2022.

R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 2019.

D. Repecka, V. Jauniskis, L. Karpus, E. Rembeza, I. Rokaitis, J. Zrimec, S. Poviloniene, A. Laurynenas, S. Viknander, W. Abuajwa, et al. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*, 3(4):324–333, 2021.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

A. J. Riesselman, J. B. Ingraham, and D. S. Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

M. Rosca, B. Lakshminarayanan, and S. Mohamed. Distribution matching in variational inference. *arXiv preprint arXiv:1802.06847*, 2018.

J.-E. Shin, A. J. Riesselman, A. W. Kollasch, C. McMahon, E. Simon, C. Sander, A. Manglik, A. C. Kruse, and D. S. Marks. Protein design and variant prediction using autoregressive generative models. *Nature communications*, 12(1):1–11, 2021.

S. Sinai, E. Kelsic, G. M. Church, and M. A. Nowak. Variational auto-encoding of protein sequences. *arXiv preprint arXiv:1712.03346*, 2017.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29, 2016.

Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34: 1415–1428, 2021.

M. A. Stiffler, D. R. Hekstra, and R. Ranganathan. Evolvability as a function of purifying selection in tem-1 β-lactamase. *Cell*, 160(5):882–892, 2015.

R. L. Tatusov, E. V. Koonin, and D. J. Lipman. A genomic perspective on protein families. *Science*, 278(5338):631–637, 1997.

J. Tomczak and M. Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

E. N. Weinstein, A. N. Amin, J. Frazer, and D. S. Marks. Non-identifiability and the blessings of misspecification in models of molecular fitness and phylogeny. *bioRxiv*, 2022.

Sampling quality in deep generative models of protein sequences (Supplementary Material)

Marloes Arts[1,*] **Federico Bergamin**[2,*] **Wouter Boomsma**[1] **Jes Frellsen**[2]

[1]Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
[2]Department of Applied Mathematics and Computer Science, Technical University of Denmark, Copenhagen, Denmark
[*]Equal contribution

A MODEL ARCHITECTURE DETAILS

We present in detail the model architectures and the training setup we used to generate the results and plots of the main paper. An overview of model settings can be found in Table A1 and Table A2. Code will be made available upon publication.

EVE For the EVE implementation Frazer et al. [2021] we closely follow the available implementation in their GitHub repository [1]. EVE is one-stochastic layer VAE with a Bayesian decoder. The encoder is a 3-layers neural network with $[2000, 1000, 300]$ hidden units which map the amino acid sequence to a 50 dimensional latent variable. The decoder is also a 3-layers neural network with $[300, 1000, 2000]$ hidden units, which map the latent observation to the logits of a Categorical distribution, modelling the input sequence. In the decoder they also consider having a final 1D convolution layer to encourage amino acid correlation and a temperature scaling parameter for the final logits, i.e. the final logits are computed as $\texttt{logits} = \log(1.0 + \exp(\texttt{temperature})) * \texttt{logits}$. Having a Bayesian decoder means that we are not treating the decoder parameters deterministically, i.e. learning a single set of weights, but instead we learn a distribution over that. This also include the logits temperature parameter. The prior over the decoder weights is assumed to be a standard Gaussian and the weights are optimize by Bayes by Backprop Blundell et al. [2015], which makes convergence to be pretty slow. Indeed, following their setting we trained the model for 45000 epochs and choose the model with the best validation error. In addition to that, they use dropout Srivastava et al. [2014] with $p = 0.1$ and they initialize both the encoder and the Bayesian decoder in a specific way. We use the same initialization when we consider a Bayesian Decoder in our models.

For the non-Bayesian version of EVE, we just consider the same encoder and decoder architecture but we did not use the 1D convolution, the logits temperature parameter, and the use of dropout during training. We also did not use the same initialization because it was mostly suited for training model with a Bayesian decoder. We instead consider batch normalization Ioffe and Szegedy [2015].

2-stochastic layer VAE For the 2-stochastic layer VAE we consider a slightly different architecture. The bottom-up structure is defined by two neural network with the following hidden units $[[1000, 500, 200], [500, 300, 200]]$ and the top-down part of the inference has a single neural network with $[200, 300, 500]$ hidden units. The final decoder, instead, is defined as a three layer network with the following structure: $[300, 500, 1000]$. The latent on the top of the hierarchy has 30 dimension, while the other one is 50 dimensional. In case we are using a Bayesian decoder we initialized all the bottom-up structure as the EVE encoder and we also initialize the Bayesian decoder in the same way as EVE. Apart from the number of hidden units, the Bayesian decoder has the same structure in same of type of layers as the original EVE.

For the non-Bayesian version, as in the EVE case, we use the same architectures without the "tricks" suggested by the DeepSequence model Riesselman et al. [2018] and included batch normalization.

3-stochastic layer VAE We follow mostly the same implementation details as before and just considered a different architecture. In this case the bottom up structure is composed by three different networks with the following shapes $[[1000, 500, 200], [500, 300, 200], [300, 200, 100]]$, where the first element of the list is the encoder. The top-down part,

[1]EVE codebase can be found here: `https://github.com/OATML-Markslab/EVE`

Preprint. Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

Table A1: Model architectures for hierarchical models.

	BOTTOM-UP	TOP-DOWN	FINAL DECODER	LATENT DIMENSION
1 LAYER	$[[2000, 1000, 300]]$	–	$[300, 1000, 2000]$	$[50]$
2 LAYER	$[[1000, 500, 200], [500, 300, 200]]$	$[[200, 300, 500]]$	$[300, 500, 1000]$	$[50, 30]$
3 LAYER	$[[1000, 500, 200], [500, 300, 200], [300, 200, 100]]$	$[[100, 200, 300], [200, 300, 500]]$	$[300, 500, 1000]$	$[50, 30, 10]$
1 LAYER 2D	$[[2000, 2000]]$	–	$[100, 2000]$	$[2]$
2 LAYER 2D	$[[2000, 2000], [100, 100]]$	$[[100, 100]]$	$[100, 2000]$	$[2, 2]$
3 LAYER 2D	$[[2000, 2000], [500, 500], [100, 100]]$	$[[50, 50], [100, 100]]$	$[100, 2000]$	$[2, 2, 2]$

Table A2: Additional model configurations, Bayesian versus non-Bayesian.

	BAYESIAN DECODER	1D CONVOLUTION	BATCHNORM	TEMPERATURE SCALING LOGITS	DROPOUT DECODER	EVE INITIALIZATION
BAYESIAN	✓	✓	–	✓	0.1	✓
NON-BAYESIAN	–	–	✓	–	0.0	–

instead, has two networks with shapes $[[100, 200, 300], [200, 300, 500]]$. The final decoder is the same as in the 2-stochastic layers case, meaning that it has $[300, 500, 1000]$ hidden units. Starting from the top latent, we use the following latent dimensions: $[50, 30, 10]$.

For the non-Bayesian version we do exactly as in the previous two models.

2D latent space models For the 2D latent space models, we use hierarchical VAE models without a Bayesian decoder, as decribed in the previous paragraphs. Since the latent space has a reduced size of only two latent variables, we use a slightly simpler setup: the neural networks between layers have only two layers instead of three, where both layers are of the same size. See Table A1 for more details.

Training setup For all the considered model we use Adam optimizer Kingma and Ba [2015] with $1e{-4}$ learning rate and a batch size of 256. All models were run for 45000 epochs, after which the best model was chosen based on the best validation loss. The only exception are the 2D latent space models, which were run for 80 epochs. As we also mention in the paper introduction, the optimization process of VAE with multiple stochastic layers is difficult. Combining them with a Bayesian decoder makes the optimization procedure more brittle.

B MULTIPLE SAMPLES FOR THE AGGREGATED POSTERIOR

In the main paper, we visualize the aggregated prior using one sample per data point. However, one could also consider drawing multiple samples per input, to get and idea of the fluctuations that can occur for a data point. Figure A1 shows the comparison between the prior and the aggregated posterior for 1 and 100 samples per data point. For 100 samples, the distribution is more spread out and certain "islands" in latent space at z_1-level start overlapping.

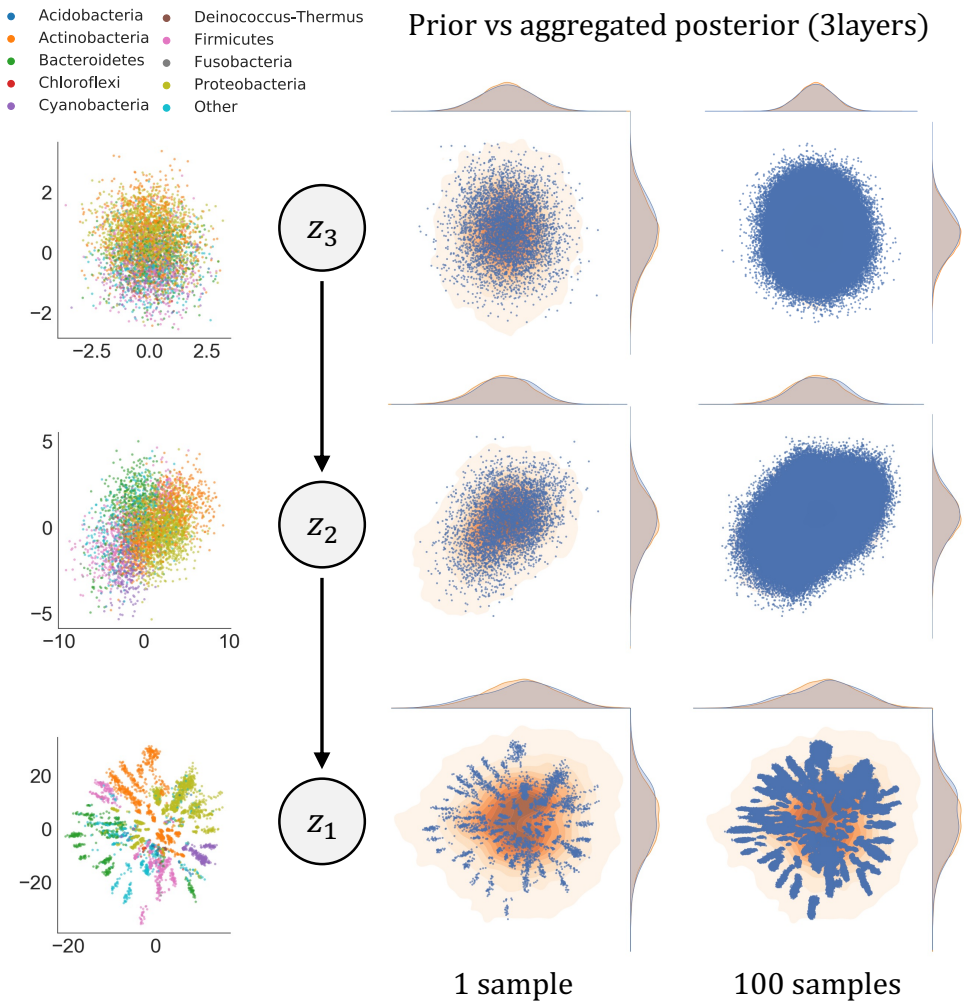


Figure A1: Left: aggregated posterior for a 3-layer non-Bayesian VAE, color coded by species. Right: comparison between prior and aggregated posterior for the same model, with 1 and 100 samples, respectively, for the aggregated posterior.

C SMALLER HIERARCHICAL VAE NEURAL NETWORK ARCHITECTURES

Since the MSA dataset that we use is relatively small and the original EVE model as well as the 2- and 3-layer variants reported in this paper contain neural networks with layer sizes that are relatively large, we also explore a smaller architecture. We reduce the number of layers in every neural network between latent variables and we set a lower latent space dimensionality (details in Table A3). Similar to the results in the main paper, we evaluate all models both in the Bayesian and the non-Bayesian setting. Interestingly, Figure A2 show that the non-Bayesian performance is very close to their larger counterparts in the main paper, both in terms of sample quality and Spearman correlation. Additionally, sample quality for the small Bayesian models behaves similarly to the large Bayesian models. However, the Spearman correlation drops substantially for 2- and 3- layer hierarchical models with a Bayesian decoder. This shows that the Bayesian variants of the model are very sensitive to the chosen architecture, and might need more flexibility to perform well in downward tasks.

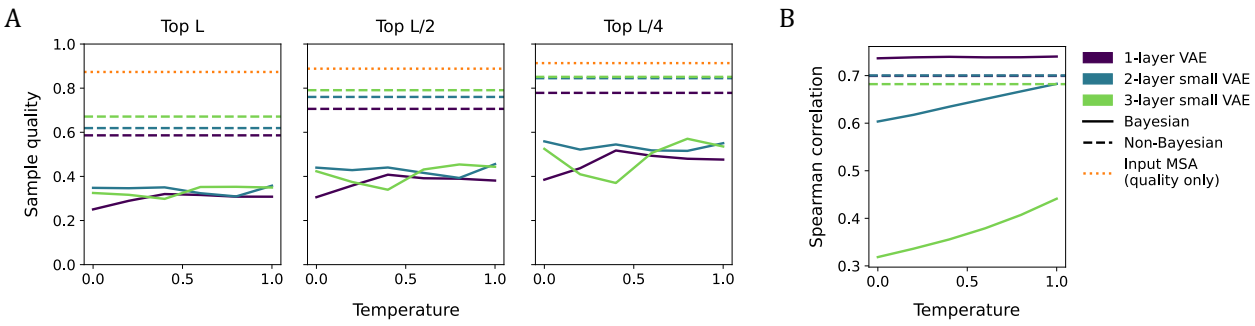


Figure A2: Sample quality (A) and Spearman correlation for mutation effect prediction (B) for hierarchical models with a smaller architecture compared to the regular sized 1-layer models at different decoder weight sampling temperatures. Dashed lines show non-Bayesian temperature-independent model variants, dotted orange line shows input MSA sample quality.

Table A3: Model architectures for small hierarchical models.

	BOTTOM-UP	TOP-DOWN	FINAL DECODER	LATENT DIMENSION
2 LAYER	$[[1000, 500], [500, 300]]$	$[[300, 500]]$	$[500, 1000]$	$[30, 10]$
3 LAYER	$[[1000, 500], [500, 300], [300, 200]]$	$[[200, 300], [300, 500]]$	$[500, 1000]$	$[30, 20, 10]$

D RESULTS

We report also the results we showcase in the main paper in a table form, which make it easier for future performance comparison. In Table A4, we report the results related to the plot in Fig.5 in the paper.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K Min, Kelly Brock, Yarin Gal, and Debora S Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 599(7883):91–95, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Table A4: Numerical results visualized in Fig.5 of the paper. We compare hierarchical VAEs both in terms of fitness prediction and sample quality. For the model with a Bayesian decoder we considered temperature scaling both for computing the Spearman correlation and for sampling.

1-STOCHASTIC LAYER (EVE)					
		FITNESS	SAMPLE QUALITY		
BAYESIAN DECODER	TEMPERATURE	SPEARMAN	TOP L	TOP L/2	TOP L/4
✓	0.0	0.7364	0.2505	0.3058	0.3851
✓	0.2	0.7384	0.2894	0.3584	0.4371
✓	0.4	0.7395	0.3200	0.4077	0.5174
✓	0.6	0.7384	0.3159	0.3916	0.4935
✓	0.8	0.7386	0.3082	0.3896	0.4800
✓	1.0	0.7401	0.3080	0.3811	0.4762
✗	–	0.6994	0.5861	0.7061	0.7786

2-STOCHASTIC LAYERS					
		FITNESS	SAMPLE QUALITY		
BAYESIAN DECODER	TEMPERATURE	SPEARMAN	TOP L	TOP L/2	TOP L/4
✓	0.0	0.7068	0.2968	0.3775	0.4776
✓	0.2	0.7143	0.3569	0.4618	0.6061
✓	0.4	0.7206	0.3472	0.4237	0.5156
✓	0.6	0.7259	0.2981	0.3547	0.4270
✓	0.8	0.7298	0.3278	0.4157	0.5133
✓	1.0	0.7297	0.3162	0.3746	0.4169
✗	–	0.6921	0.5792	0.7074	0.7680

3-STOCHASTIC LAYERS					
		FITNESS	SAMPLE QUALITY		
BAYESIAN DECODER	TEMPERATURE	SPEARMAN	TOP L	TOP L/2	TOP L/4
✓	0.0	0.7377	0.2779	0.3683	0.4822
✓	0.2	0.7375	0.3206	0.4030	0.5059
✓	0.4	0.7348	0.3839	0.4725	0.5468
✓	0.6	0.7348	0.3402	0.4258	0.5151
✓	0.8	0.7359	0.3649	0.4462	0.5011
✓	1.0	0.7336	0.3714	0.4747	0.5992
✗	–	0.6927	0.6130	0.7507	0.8344

Chapter 6

Concluding Remarks

The field intersecting machine learning and structural biology is moving at an incredible pace and the arrival of AlphaFold2 [1] only added more fuel to the fire, setting off a cascade of generative models for static structure prediction. However, the dynamic nature of proteins is still lacking from this static picture. Despite progress in ensemble density modelling and force field computation [41, 49, 117], the field is still far away from a generative method over ensembles that generalizes across systems, and even further away from a cheap way to obtain force fields for unseen systems. While these objectives might be currently out of reach due to lack of training data, the majority of the work done during my Ph.D. was aimed at taking intermediate steps towards these goals.

Initially, we focused on the parameterization choice that needs to be made when modeling protein structure variance. For most models, the selected representation is a consequence of the model application: if local structure is prioritized over global structure, *e.g.* in local models or for small proteins, one can be inclined to

choose an internal coordinate parameterization, while Cartesian coordinates might be the better option when the emphasis is on global structure. In Chapter 3, we reconciled these two representations to get the best of both worlds. The proof-of-concept VAE model trained on protein structure ensembles is able to generate high-quality samples, both in terms of local and global structure.

However, we are also aware of the limitations of this method. First of all, our first-order approximation only holds for small atom perturbations in 3D space. This issue could be alleviated by combining multiple small perturbations in a hierarchical VAE or, ultimately, a diffusion model. For example, the recently released diffusion-based Chroma model relies on correlated diffusion rather than the commonly used uncorrelated variant [35, Appendix C], and we believe our method could provide a richer covariance structure compared to the ones mentioned in the paper. The model we chose in our paper to show the method's potential was a relatively simple VAE in combination with a U-Net [118], where the latter was a remnant of our own earlier experiments predicting the fluctuations of pairwise distances directly in a non-generative setting, see Appendix A. In the current setting, the U-Net directly predicts the Lagrange multipliers $\boldsymbol{\lambda}$, which affect the degree of 3D fluctuation per atom and are kept low through a regularizing loss term. Since the relationship between $\boldsymbol{\lambda}$ and the true atom fluctuation constraints $\boldsymbol{C}$ is known (see Equation 10 in the paper), we are presently experimenting with an auxiliary maximum likelihood loss on $\boldsymbol{C}$ directly. This is ongoing work and so far this model objective has proven hard to optimize in the current model setting. We hope this research will present a small step towards accurately modelling protein structure ensembles in terms of local

as well as global structure.

In Köhler et al. [49], the authors chose to parameterize protein structures using internal coordinates, and train a normalizing flow-based model to get a density estimator over $C_\alpha$coarse-grained protein dynamics structure data. Additionally, they train a second model in a teacher-student setup to do explicit force matching with respect to the flow model and thereby obtain a force field to use for simulations. While the results were promising, the method does not scale well to larger systems due to its limited capacity to model global structure and the restrictions imposed by the bijective transformations within the normalizing flow. Moreover, ensuring local chemical integrity necessitated the use of auxiliary loss terms to prevent steric clashes and bond dissociation. Finally, performing dynamics with the obtained force field was quite challenging and required parallel tempering (*i.e.* increasing and decreasing the temperature) to increase the change of going from one low-energy basin to another. In Chapter 4, we set out to solve some of these problems using a diffusion model. The inputs are set in Cartesian space with rotational data augmentation and all molecules are centered to zero to make our output invariant to rototranslational transformations. This method, which has a much simpler training setup without auxiliary losses, outperforms the flow-based method in equilibrium setting. In addition, we can extract the implicitly learned score function from the model and directly translate it into a force field. The simulations obtained utilizing these extracted force fields (without parallel tempering) follow the original molecular dynamics simulations better than the flow-based model. We also demonstrate that our method scales better to larger systems, showing results on protein G which was outside the modeling range of the flow-based method.

However, there is still room for improvement. Since the proposed method still has a slight tendency to get stuck in low-energy basins, we are now experimenting with flexible levels of noise, similar to parallel tempering, to improve simulations. Another thing that might help the model "generalize within the protein" is to include amino acid labels in the coarse-grained bead properties, such that the individual chemical properties can be learned. Even though this method still needs to be trained per system, our hope is that a similar model could be trained to generalize across proteins, given enough training data.

Generative models, and VAEs in particular, have been applied to protein family sequence (MSA) data [11, 12]. Interestingly, for one-layer VAEs, different species separate into a "star-shape" in the latent representation. This means there is a considerable mismatch between the standard Gaussian prior and the aggregated posterior, potentially leading to suboptimal sample quality. In Chapter 5 we address this issue and alleviate it slightly by introducing multiple layers of latent space in an LVAE model. Even though the improvement was less dramatic than expected in terms of sample quality and the model still assigns probability mass to regions without observed data points, the reduced mismatch between the prior and the aggregated posterior is a promising result.

A counter-intuitive result that we encountered throughout all our experiments is that there appears to be an inverse relationship between sample quality and protein fitness prediction. Even though it has been shown before that these objectives do not necessarily align, an effect that has even been called the "blessing of misspecification" [119], we are still looking into explaining this observation. To this end, we plan to include experimental data in the train-

ing set to investigate whether assay data and evolutionary data agree on the objective to be optimized. We also found that using a Bayesian decoder is detrimental to sample quality, where the effect of the decoder weight sampling temperature is negligible, *i.e.* the cold posterior effect is not present or not strong enough to make a difference. This might be due to the lack of covariance structure between the weight sampling distributions in the decoder, and we are looking into ways to address this problem while avoiding memory problems, such as last-layer Laplace approximation [80].

Overall, the work presented in this Ph.D. thesis focused on generative models for protein structure and sequence variation in the context of protein structure ensembles, protein dynamics, and the evolution of amino acid sequences. It has been rewarding to be able to contribute with steps towards a better understanding of this challenging, hybrid field.

Appendix A

Pairwise distance distributions

Poster: **Protein Structure Variance Prediction using Deep Learning and Molecular Dynamics Simulations**

The poster on the next page outlines a U-Net based model to predict pairwise distance variance directly from the amino acid sequence and native structure for a protein. The poster was presented at the Geometric Deep Learning Summer School 2021, where it received the Best Poster Award.

Protein Structure Variance Prediction using Deep Learning and Molecular Dynamics Simulations

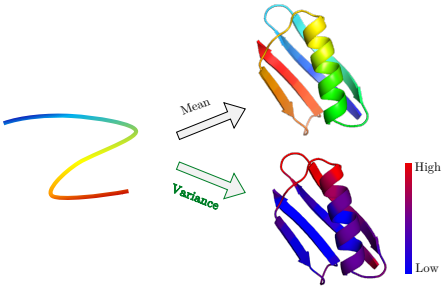
Marloes Arts ✉ ma@di.ku.dk , Wouter Boomsma ✉ wb@di.ku.dk

UNIVERSITY OF COPENHAGEN

Protein Structure Variance

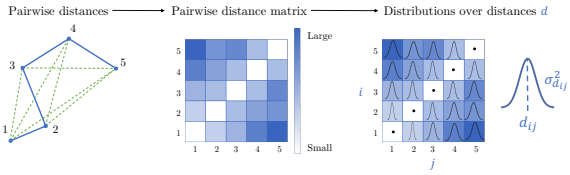
Proteins are involved in virtually all processes within cells. They consist of a string of amino acids, folded up into a **specific 3D structure** which is directly linked to the function of the protein.

Recently, there has been huge progress in predicting the static structure of proteins, but this is not the whole story. **Proteins are dynamic molecules**, and some parts have more variance than others.



Goal

Given the amino acid sequence and static structure of a protein, **predict the distribution over all pairwise distances within a protein**.



Motivation:

- Insight in protein **function**
- **Sampling** structures (e.g. data augmentation)
- *Future direction:* **weigh distances according to their variance** when predicting the mean structure from amino acid sequence (i.e. distances with less variance are probably more important for the main structure)

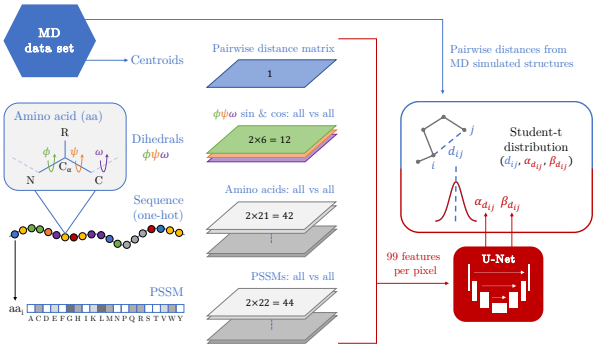
Data Set: Molecular Dynamics

We train and validate our network on a (soon to be published) data set that was constructed using **Molecular Dynamics (MD)**[1]. For each protein, we have 399 simulated structures with an interval of 50 ps. For this poster, we use all proteins with al length of ≤200 amino acids (474 train, 19 validation, 62 test).

Methods

- **Model**

  Our model is based on a **U-Net**, a type of architecture that is well-known in imaging tasks such as segmentation. The input features are processed to form an "image" with 99 features per pixel.



- **Reparameterization of $\sigma_{d_{ij}}$**

  For more robust training, we choose the variance to be distributed according to the **conjugate prior** of Gaussian data: the inverse-Gamma distribution. This results in a **student-t distribution**[2].

$y_{d_{ij}}$: target distance

$\mu_{d_{ij}}$: mean (centroid) distance

$\sigma^2_{d_{ij}} \sim \text{Inv-Gamma}(\alpha_{d_{ij}}, \beta_{d_{ij}})$

$$\begin{aligned} \text{NLL}_{d_{ij}} &= -\log p_\theta\left(y_{d_{ij}}\right) \\ &= -\log \int \mathcal{N}\left(y_{d_{ij}} \mid \mu_{d_{ij}}, \sigma^2_{d_{ij}}\right) d\sigma^2_{d_{ij}} \\ &= -\log t_{\mu_{d_{ij}}, \alpha_{d_{ij}}, \beta_{d_{ij}}}\left(y_{d_{ij}}\right) \end{aligned}$$

Gaussian

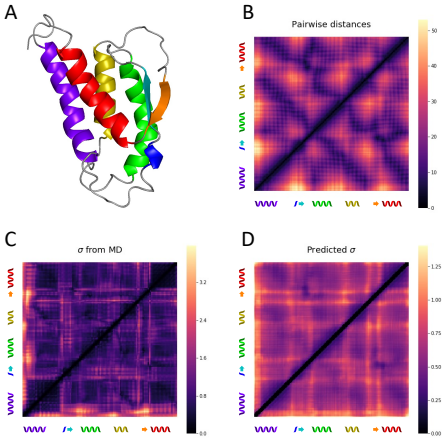
Integrate out $\sigma^2_{d_{ij}}$

(Preliminary) Results

Our network captures the pairwise distance **variance pattern** quite well. Importantly, the predicted σ is a result of minimizing the NLL, **without the use of a ground truth σ**.

Example →

A. 3D structure ('1irl'), elements are colored and indicated in B-D
B. Pairwise distances
C. "Ground truth" σ from Molecular Dynamics
D. Predicted σ



Conclusion:

We propose a U-Net based network that can capture protein structure variance patterns given their amino acid sequence and static structure, solely by minimizing the negative log likelihood on structures simulated through Molecular Dynamics.

1. Data set built by Tone Bengtsen, previously a postdoc in our group.
2. Nicki S Detlefsen, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. arXiv preprint arXiv:1906.03260, 2019.

Bibliography

[1] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[2] Carl Ivar Branden and John Tooze. *Introduction to protein structure*. Garland Science, 2012.

[3] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular biology of the cell (6th edition)*. Garland Science, 2014.

[4] Frederick Sanger and Hans Tuppy. The amino-acid sequence in the phenylalanyl chain of insulin. 1. the identification of lower peptides from partial hydrolysates. *Biochemical journal*, 49(4):463, 1951.

[5] Jerry Donohue. Hydrogen bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 39(6): 470–478, 1953.

[6] Christian B Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[7] Margaret O Dayhoff. The origin and evolution of protein superfamilies. In *Federation proceedings*, volume 35, pages 2132–2138, 1976.

[8] Victor Kunin, Ildefonso Cases, Anton J Enright, Victor de Lorenzo, and Christos A Ouzounis. Myriads of protein families, and still counting. *Genome biology*, 4:1–2, 2003.

[9] CA Floudas, HK Fung, SR McAllister, M Mönnigmann, and R Rajgaria. Advances in protein structure prediction and de novo protein design: A review. *Chemical Engineering Science*, 61(3):966–988, 2006.

[10] Mohammad Yaseen Sofi, Afshana Shafi, and Khalid Z Masoodi. *Bioinformatics for everyone*, chapter 6. Academic Press, 2021.

[11] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

[12] Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K Min, Kelly P. Brock, Yarin Gal, and Debora S. Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 2021.

[13] E Fischer. Die hydrolyse der proteinstoffe. *Zeitschrift für Untersuchung der Nahrungs-und Genußmittel, sowie der Gebrauchsgegenstände*, 5: 1207–1208, 1902.

[14] Fr Hofmeister. Über bau und gruppierung der eiweisskörper. *Ergebnisse der Physiologie*, 1(1):759–802, 1902.

[15] Kaj Ulrik Linderstrøm-Lang. *Proteins and enzymes*, volume 6. Stanford university press, 1952.

[16] Linus Pauling, Robert B Corey, and Herman R Branson. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 37(4):205–211, 1951.

[17] William Thomas Astbury and Henry J Woods. X-ray studies of the structure of hair, wool, and related fibres. ii.-the molecular structure and elastic properties of hair keratin. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 232(707-720):333–394, 1933.

[18] William Henry Bragg and William Lawrence Bragg. The reflection of x-rays by crystals. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 88 (605):428–438, 1913.

[19] John C Kendrew, G Bodo, Howard M Dintzis, RG Parrish, Harold Wyckoff, and David C Phillips. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, 181(4610):662–666, 1958.

[20] Jeffrey A Purslow, Balabhadra Khatiwada, Marvin J Bayro, and Vincenzo Venditti. Nmr methods for structural characterization of protein-protein complexes. *Frontiers in molecular biosciences*, 7:9, 2020.

[21] Kurt Wüthrich. The way to nmr structures of proteins. *Nature structural biology*, 8(11):923–925, 2001.

[22] G Marius Clore. Adventures in biomolecular nmr. *eMagRes*, pages 1–7, 2007.

[23] Richard Henderson, Joyce M Baldwin, Thomas A Ceska, Friedrich Zemlin, Erich Beckmann, and Kenneth H Downing. Model for the structure of bacteriorhodopsin based on high-resolution electron cryomicroscopy. *Journal of molecular biology*, 213(4):899–929, 1990.

[24] Ka Man Yip, Niels Fischer, Elham Paknia, Ashwin Chari, and Holger Stark. Atomic-resolution protein structure determination by cryo-em. *Nature*, 587(7832):157–161, 2020.

[25] Christian B Anfinsen, Robert R Redfield, Warren L Choate, Juanita Page, and William R Carroll. Studies on the gross structure, cross-linkages, and terminal sequences in ribonuclease. *Journal of Biological Chemistry*, 207(1):201–210, 1954.

[26] Mario Compiani and Emidio Capriotti. Computational and theoretical methods for protein folding. *Biochemistry*, 52(48):8601–8624, 2013.

[27] Benjamin Webb and Andrej Sali. Comparative protein structure modeling using modeller. *Current protocols in bioinformatics*, 54(1):5–6, 2016.

[28] Ambrish Roy, Alper Kucukural, and Yang Zhang. I-tasser: a unified platform for automated protein structure and function prediction. *Nature protocols*, 5(4):725–738, 2010.

[29] David E Kim, Dylan Chivian, and David Baker. Protein structure prediction and analysis using the robetta server. *Nucleic acids research*, 32(suppl_2):W526–W531, 2004.

[30] Julia Koehler Leman, Brian D Weitzner, Steven M Lewis, Jared Adolf-Bryfogle, Nawsad Alam, Rebecca F Alford, Melanie Aprahamian, David Baker, Kyle A Barlow, Patrick Barth, et al. Macromolecular modeling and design in rosetta: recent methods and frameworks. *Nature methods*, 17(7):665–680, 2020.

[31] Mohammed AlQuraishi. End-to-end differentiable learning of protein structure. *Cell systems*, 8(4):292–301, 2019.

[32] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557): 871–876, 2021.

[33] Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, Bonnie Berger, et al. High-resolution de novo structure prediction from primary sequence. *BioRxiv*, pages 2022–07, 2022.

[34] Kevin E Wu, Kevin K Yang, Rianne van den Berg, James Y Zou, Alex X Lu, and Ava P Amini. Protein structure generation via folding diffusion. *arXiv preprint arXiv:2209.15611*, 2022.

[35] John Ingraham, Max Baranov, Zak Costello, Vincent Frappier, Ahmed Ismail, Shan Tie, Wujie Wang, Vincent Xue, Fritz Obermeyer, Andrew Beam, et al. Illuminating protein space with a programmable generative model. *bioRxiv*, pages 2022–12, 2022.

[36] Jon Baker, Alain Kessi, and Bernard Delley. The generation and use of delocalized internal coordinates in geometry optimization. *The Journal of chemical physics*, 105(1):192–212, 1996.

[37] Marissa G Saunders and Gregory A Voth. Coarse-graining methods for computational biology. *Annual review of biophysics*, 42:73–93, 2013.

[38] Helgi I Ingólfsson, Cesar A Lopez, Jaakko J Uusitalo, Djurre H de Jong, Srinivasa M Gopal, Xavier Periole, and Siewert J Marrink. The power of coarse graining in biomolecular simulations. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 4(3):225–248, 2014.

[39] Yao Xu and Martina Havenith. Perspective: Watching low-frequency vibrations of water in biomolecular recognition by thz spectroscopy. *The Journal of chemical physics*, 143(17):170901, 2015.

[40] Scott A Hollingsworth and Ron O Dror. Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143, 2018.

[41] Paul Robustelli, Stefano Piana, and David E Shaw. Developing a molecular dynamics force field for both folded and disordered protein states. *Proceedings of the National Academy of Sciences*, 115(21): E4758–E4766, 2018.

[42] Jing Huang, Sarah Rauscher, Grzegorz Nawrocki, Ting Ran, Michael Feig, Bert L De Groot, Helmut Grubmüller, and Alexander D MacKerell Jr. Charmm36m: an improved force field for folded and intrinsically disordered proteins. *Nature methods*, 14(1):71–73, 2017.

[43] Edward Harder, Wolfgang Damm, Jon Maple, Chuanjie Wu, Mark Reboul, Jin Yu Xiang, Lingle Wang, Dmitry Lupyan, Markus K Dahlgren, Jennifer L Knight, et al. Opls3: a force field providing broad coverage of drug-like small molecules and proteins. *Journal of chemical theory and computation*, 12(1):281–296, 2016.

[44] Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, and David E Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

[45] T Schneider and E Stoll. Molecular-dynamics study of a three-dimensional one-component model for distortive phase transitions. *Physical Review B*, 17(3):1302, 1978.

[46] Sabry G Moustafa, Andrew J Schultz, and David A Kofke. Effects of thermostatting in molecular dynamics on anharmonic properties of crystals: Application to fcc al at high pressure and temperature. *The Journal of chemical physics*, 149(12):124109, 2018.

[47] William George Noid, Jhih-Wei Chu, Gary S Ayton, Vinod Krishna, Sergei Izvekov, Gregory A Voth, Avisek Das, and Hans C Andersen. The multiscale coarse-graining method. i. a rigorous bridge between atomistic and coarse-grained models. *The Journal of chemical physics*, 128(24):244114, 2008.

[48] M Scott Shell. The relative entropy is fundamental to multiscale and inverse thermodynamic problems. *The Journal of chemical physics*, 129 (14):144108, 2008.

[49] Jonas Köhler, Yaoyi Chen, Andreas Krämer, Cecilia Clementi, and Frank Noé. Flow-matching: Efficient coarse-graining of molecular dynamics without forces. *Journal of Chemical Theory and Computation*, 19(3):942–952, 2023.

[50] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.

[51] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, chapter 14. MIT press, 2016.

[53] Raphael R Eguchi, Christian A Choe, and Po-Ssu Huang. Ig-vae: Generative modeling of protein structure by direct 3d coordinate generation. *PLoS computational biology*, 18(6):e1010271, 2022.

[54] Alex Hawkins-Hooker, Florence Depardieu, Sebastien Baur, Guillaume Couairon, Arthur Chen, and David Bikard. Generating functional protein variants with variational autoencoders. *PLoS computational biology*, 17(2):e1008736, 2021.

[55] Joe G Greener, Lewis Moffat, and David T Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific reports*, 8(1):16189, 2018.

[56] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4): 307–392, 2019.

[57] Ben Lambert. A student's guide to bayesian statistics. *A Student's Guide to Bayesian Statistics*, pages 121–140, 2018.

[58] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[59] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[60] Diederik Pieter Kingma. Variational inference & deep learning: A new synthesis. *Thesis, Universiteit van Amsterdam*, 2017.

[61] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[62] Teun Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978.

[63] Chin-Wei Huang, Kris Sankaran, Eeshan Dhekane, Alexandre Lacoste, and Aaron Courville. Hierarchical importance weighted autoencoders. In *International Conference on Machine Learning*, pages 2869–2878. PMLR, 2019.

[64] Chris Cremer, Quaid Morris, and David Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

[65] Geoffrey Roeder, Yuhuai Wu, and David K Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *Advances in Neural Information Processing Systems*, 30, 2017.

[66] Lorenz Richter, Ayman Boustati, Nikolas Nüsken, Francisco Ruiz, and Omer Deniz Akyildiz. Vargrad: a low-variance gradient estimator for variational inference. *Advances in Neural Information Processing Systems*, 33:13481–13492, 2020.

[67] Tom Rainforth, Adam Kosiorek, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning*, pages 4277–4285. PMLR, 2018.

[68] Justin Domke and Daniel R Sheldon. Importance weighting and variational inference. *Advances in neural information processing systems*, 31, 2018.

[69] Jack Klys, Jesse Bettencourt, and David Duvenaud. Joint importance sampling for variational inference. 2018.

[70] George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. Doubly reparameterized gradient estimators for monte carlo objectives. *arXiv preprint arXiv:1810.04152*, 2018.

[71] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29, 2016.

[72] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. Biva: A very deep hierarchy of latent variables for generative modeling. *Advances in neural information processing systems*, 32, 2019.

[73] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33: 19667–19679, 2020.

[74] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*, 3 (2), 2016.

[75] David JC MacKay. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. *Inference Group, Cavendish Laboratory, Cambridge, UK*, 2001.

[76] Erik Daxberger and José Miguel Hernández-Lobato. Bayesian variational autoencoders for unsupervised out-of-distribution detection. *arXiv preprint arXiv:1912.05651*, 2019.

[77] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[78] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[79] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[80] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

[81] Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

[82] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[83] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[84] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[85] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[86] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[87] Jonas Oppenlaender. The creativity of text-to-image generation. In *Proceedings of the 25th International Academic Mindtrek Conference*, pages 192–202, 2022.

[88] StabilityAI. Stable diffusion, . URL `https://stablediffusionweb.com/`.

[89] StabilityAI. Midjourney, . URL `https://www.midjourney.com/`.

[90] OpenAI. Dall-e 2. URL `https://openai.com/product/dall-e-2`.

[91] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *bioRxiv*, pages 2022–12, 2022.

[92] Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. *arXiv preprint arXiv:2205.15019*, 2022.

[93] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *arXiv preprint arXiv:2206.01729*, 2022.

[94] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[95] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[96] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[97] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

[98] Lucas Theis and Matthias Bethge. Generative image modeling using spatial lstms. *Advances in neural information processing systems*, 28, 2015.

[99] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[100] Marloes Arts, Ihor Smal, Maarten W Paul, Claire Wyman, and Erik Meijering. Particle mobility analysis using deep learning and the moment scaling spectrum. *Scientific reports*, 9(1):17160, 2019.

[101] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[102] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[103] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

[104] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[105] Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. *Advances in Neural Information Processing Systems*, 32, 2019.

[106] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[107] James Martens, Ilya Sutskever, and Kevin Swersky. Estimating the hessian by back-propagating curvature. *arXiv preprint arXiv:1206.6464*, 2012.

[108] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[109] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[110] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[111] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[112] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

[113] Tim Salimans and Jonathan Ho. Should ebms model the energy or the score? In *Energy Based Models Workshop-ICLR 2021*, 2021.

[114] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*, 2021.

[115] Giorgio Favrin, Anders Irbäck, and Fredrik Sjunnesson. Monte carlo update for chain molecules: biased gaussian steps in torsional space. *The Journal of Chemical Physics*, 114(18):8154–8158, 2001.

[116] Sandro Bottaro, Wouter Boomsma, Kristoffer E. Johansson, Christian Andreetta, Thomas Hamelryck, and Jesper Ferkinghoff-Borg. Subtle monte carlo updates in dense molecular systems. *Journal of chemical theory and computation*, 8(2):695–702, 2012.

[117] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

[118] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[119] Eli Weinstein, Alan Amin, Jonathan Frazer, and Debora Marks. Non-identifiability and the blessings of misspecification in models of molecular fitness. *Advances in Neural Information Processing Systems*, 35: 5484–5497, 2022.

Acknowledgements

This Ph.D. was by no means done in isolation, and I would like to take this opportunity to thank all the people who have helped me in various ways.

Above all, Wouter Boomsma, for being the most supportive and enthusiastic supervisor a Ph.D. student could wish for. Thanks for giving me the opportunity to be the first Ph.D. student in your group, for all the great ideas and brainstorming sessions, and for always staying infectiously positive throughout the many bumps in the road.

The entire BioML group, both past and present, for always being up for whiteboard discussions and coffee breaks. Thanks for the cozy group meetings, for the interesting journal clubs and for collectively proof-reading this thesis and giving feedback, which was a truly great and helpful experience.

All the people at DIKU, for a welcoming environment, a "hygge" time at lunch, engaging scientific conversations and really fun parties. Thanks to everyone for making me look forward to coming to work.

The Microsoft Research group in Amsterdam, in particular Rianne van den Berg and Victor García Satorras, for giving me the privilege of doing an internship in a group of incredibly talented people. I have immensely enjoyed the scientific collaboration, the visit to Cambridge, the amazing food and the ping-pong sessions.

Jes Frellsen and Federico Bergamin, as well as the Probabilistic Modeling group at DTU, where I was fortunate enough to do my change of environment. Thanks to Federico for following through

on a cursed paper despite his negative view, and thanks to Jes for being the yin to Federico's yang.

My family back in the Netherlands, for their loving support and for being unconditionally proud of me. Thanks to my mother, my father, my brother and his girlfriend, and my grandparents, of whom my grandma sadly passed away during the last year of my Ph.D.

All the friends I made here in Denmark as well as the friends back in the Netherlands. Thanks for the fun trips, video game nights, dungeons and dragons sessions, sports, dinners, karaoke nights, and all your genuinely amazing friendship.

Last, but certainly not least, my wonderful boyfriend Arno, who has always been there for me throughout the last four years. Thanks for reminding me that putting in more hours does not necessarily mean being more productive, for being patient with me close to deadlines, for playing board games with me every evening and for making me feel loved and supported every single day.

Funding

My Ph.D. was financially supported by the Novo Nordisk Foundation, which also funded the Center for Basic Machine Learning in Life Science (MLLS) that has provided me with opportunities for networking and collaboration. The grant numbers are NNF18OC0052719 for the Ph.D. , and NNF20OC0062606 for the MLLS Center.