On deep generative modelling methods for protein-protein interaction

Adam Leach

A Thesis presented for the degree of
Doctor of Philosophy

Department of Computer Science
Durham University
United Kingdom
March 2023

Abstract

Proteins form the basis for almost all biological processes, identifying the interactions that proteins have with themselves, the environment, and each other are critical to understanding their biological function in an organism, and thus the impact of drugs designed to affect them. Consequently a significant body of research and development focuses on methods to analyse and predict protein structure and interactions. Due to the breadth of possible interactions and the complexity of structures, *in sillico* methods are used to propose models of both interaction and structure that can then be verified experimentally. However the computational complexity of protein interaction means that full physical simulation of these processes requires exceptional computational resources and is often infeasible. Recent advances in deep generative modelling have shown promise in correctly capturing complex conditional distributions. These models derive their basic principles from statistical mechanics and thermodynamic modelling. While the learned functions of these methods are not guaranteed to be physically accurate, they result in a similar sampling process to that suggested by the thermodynamic principles of protein folding and interaction. However, limited research has been applied to extending these models to work over the space of 3D rotation, limiting their applicability to protein models. In this thesis we develop an accelerated sampling strategy for faster sampling of potential docking locations, we then address the rotational diffusion limitation by extending diffusion models to the space of $SO(3)$ and finally present a framework for the use of this rotational diffusion model to rigid docking of proteins.

Declaration

The work in this thesis is based on research carried out at the Department of Computer Science, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

Conducting this PhD has been a challenging experience which would not have been possible for me to finish without the support and dedication of numerous people.

Firstly, I would hope to express my gratitude to my supervisor Dr Chris G. Willcocks, without whose guidance and encouragement I would not have imagined succeeding. Chris' enthusiasm, friendliness and knowledge have given me the support and motivation I needed to publish the work present in this thesis. I could not have asked for a better mentor for my PhD study.

I am also very grateful for my fellow Durham PhD students, whose company and warmth got me through many deadlines and doubts about my ability to complete this work.

Finally I would like to say thank you to my Mum and Dad, for believing in me and supporting me throughout my entire life. Without their guidance and support I could never imagine getting to where I am now.

Contents

List of Figures

List of Tables

Nomenclature

ALDH2 Aldehyde Dehydrogenase 2

CD Contrastive Divergence

CDF Cumulative Distribution Function

CLT central Limit Theorem

CNN Convolutional Neural Network

CoM Center of Mass

Cryo-EM Cryogenic Electron Microscopy

DDPM Denoising Diffusion Probabilistic Model

DIPS Database of Interacting Protein Structures

DNA DeoxyriboNucleic Acid

E(n)-GNN E(n) equivariant Graph Neural Network

EBM Energy Based Model

EGCL Equivariant Graph Convolutional Layer

FAVOR+ Fast Attention Via positive Orthogonal Random features

FFT Fast Fourier Transform

FPA Flower Pollination Algorithm

GAN Generative Adversarial Network

GCN Graph Convolutional Network

GECN Group Equivariant Convolutional Network

GFN Graph Field Network

GIN Graph Isomorphism Network

GMN Graph Matching Networks

GNN Graph Neural Network

GSO Glowworm Swarm Optimization

GTN Graph Transformer Network

IDP Intrinsically Disordered Protein

IG Isotropic Gaussian distribution on $SO(3)$

KL Kullback–Leibler

LSTM Long Short Term Memory

MCMC Markov Chain Monte Carlo

MLE Maximum Likelihood Estimation

MPNN Message Passing Neural Network

mRNA messenger RNA

MSE Mean Squared Error

NLL Negative Log-Likelihood

NLP Natural Language Processing

ORN Oriented Response Network

PaiNN Polarisable atom interaction Neural Network

PDE Partial Differential Equation

PDF Probability Density Function

PLM Protein Language Model

PSO Particle Swarm Optimisation

RBF Radial Basis Function

RNN Recurrent Neural Networks

SGD Stochastic Gradient Descent

SVD Singular Value Decomposition

SVM Support Vector Machine

TFN Tensor Field Network

UAT Universal Approximation Theorem

VAE Variational Auto-Encoder

WN Wrapped Normal

CHAPTER 1

Introduction

Proteins play a key part in almost all biological interactions. Structural proteins such as keratin [1] and collagen [2] are essential in forming materials such as nail, hair, cartilage, tendons and skin; Other proteins such as those found in blood, haemoglobin [3] and fibrin [4] are fundamentally required for a functioning circulatory system, carrying oxygen around the body and preventing blood loss through clotting respectively. Energy and nutrient extraction require proteins to function, digestive enzymes [5], and photosynthetic reaction centre proteins [6] are essential for all forms of cellular life to exist. Fundamentally, virtually all processes inside of living organisms are executed and mediated by proteins.

Experimental results on how proteins fold and interact suffer from the complexity, cost and time required to run experiments. Computational modelling allows for rapid advancements in the structural knowledge of proteins [7], leading to the ability to reason about likely protein interactions, to propose mechanisms of action, and to design drugs that interact with proteins before experimental results can confirm the computational suggestions. Therefore, accurately predicting protein properties can significantly speed up many pharmaceutical developments. Accurate predictions of protein behaviour are typically done using computationally intensive techniques

such as molecular dynamics simulations. However, as these proteins have developed through evolution, we can assume that small changes in sequence will result in small changes in structure and function. If this assumption did not hold, small mutations would result in drastic changes to these properties, making gradual evolution and therefore functional optimisation over generations impossible. Thus, we can apply the core assumption of deep learning methods; that the mapping between input and output can be well approximated and generalised by smooth functions. Therefore, these simulations may be accurately approximated by strategies such as deep generative modelling.

This thesis aims to contribute to this area of research by exploring methods to enhance existing protein modelling workflows and to apply state-of-the-art generative models to protein interaction. The first part of this thesis focuses on extending traditional rigid protein docking methods by providing better "seed" positions (Chapter 3). The rest of the thesis focuses on extending and applying diffusion models to rigid protein docking by modifying them to work on 3D rotations (Chapters 4 and 5).

1.1 Motivation

Recent advances in deep generative modelling have led to many new applications in modelling complex datasets across a variety of tasks. Domains such as natural images, audio and text have seen rapid advancements in the quality of generated samples. In particular, diffusion models [8] have shown high quality results in a variety of fields. While previous forms of high quality generative models such as Generative Adversarial Networks (GANs) [9] and Variational Auto-Encoders (VAEs) [10] are able to learn distributions, issues such as mode collapse during training and blurry samples limit their practical applications. Diffusion models use a stochastic process to sample from a learned data distribution in an iterative fashion. Iterative sampling allows for diffusion models to address the shortcomings of VAEs without resorting to the adversarial techniques used in GANs that introduce biases such as mode collapse.

However, several shortcomings exist when attempting to apply diffusion models to protein data. Foremost is the choice of representation of protein structure. While audio and image data (respectively) are typically represented as dense $\mathbb{R}^{l \times c}$ and $\mathbb{R}^{h \times w \times c}$ arrays for 1D and 2D convolutions, structural protein data exists in 3D space. A volumetric representation - i.e. $\mathbb{R}^{h \times w \times d \times c}$, scales with the cube of the resolution, and quickly takes up large amounts of storage (Table 1.1).

Type	Dense Representation	Memory Required (32-bit floating point)
Audio	1024×16	64kB
Image	$1024^2 \times 16$	64MB
Volumetric	$1024^3 \times 16$	**64GB**

Table 1.1: Memory usage of typical dense data representations. An early layer in a neural network can easily consist of 16 feature channels. Memory usage quickly becomes untenable.

Typical diffusion models act upon data in $\mathbb{R}^n$ space, $\mathbb{R}^{n \times n \times 3}$ for images and $\mathbb{R}^{n \times 1}$ for audio. However, protein structures are heavily constrained by physical properties such as molecular bond length. While a naive diffusion process can learn these physical properties, the learned data distribution is only an approximation of the true distribution. Enforcing these physical constraints through informed network design allows for a closer approximation to the true data distribution. This requires defining a form of diffusion model that can act upon a compound space of $SO(2)$, $SO(3)$ and R^3.

The inherent structure of protein data poses an interesting challenge. Whereas dense representations of image and audio data can be reasonably represented given current computational constraints, three dimensional data such as proteins cannot. Application of alternative data representations such as pointcloud, rigid gas and graph based networks are increasingly common and leverage results from a wide range of deep learning areas.

Stochastic generative models such as denoising diffusion [8] and score matching networks [11] derive their basic principles from statistical mechanics and thermodynamic modelling [12]. While the learned functions of these methods are not guaranteed to be physically accurate, they result in a similar sampling process to that suggested by the thermodynamic principles of protein folding. Therefore inves-

tigating this class of models is motivated by strong parallels with the actual physical processes of protein folding. While outside the scope of this thesis, diffusion models also have the potential to solve a wide range of problems regarding the calculation of transitional states between known protein conformations [13], leading to a better understanding of many biological processes.

1.2 Thesis Contributions

The main contributions of this thesis are as follows:

- A broad literature review of research and concepts pertinent to diffusion based protein models, covering classical protein docking computational methods, existing machine learning driven enhancements, and the building blocks of state-of-the-art models (Chapter 2).
- A extension to classical rigid-body protein docking methods to quickly generate candidate docking poses for further evaluation (Chapter 3).
- A novel form of diffusion process, enabling their use on the space of 3D rotations (Chapter 4).
- A framework for the application of $SO(3)$ diffusion processes to rigid protein docking, including details on combining rotational and translational diffusion, multiple target prediction methods, and discussion on issues involved with modelling diffusion across a wide variety of protein scales, including true docking position leakage, and correct scaling of the noising process.

1.3 Publications

Work contained in this thesis has been previously published in the following peer-reviewed publications by the author, and is used in the chapters as indicated below.

- **Shape tracing: An extension of sphere tracing for 3D non-convex collision in protein docking**, Adam Leach, Lucas S.P. Rudden, Sam Bond-

Taylor, John C. Brigham, Matteo T. Degiacomi, Chris G. Willcocks, In Proceedings of the 20th International Conference on Bioinformatics and Bioengineering (BIBE), IEEE 2020 (Contributing to Chapter 3).

- **Denoising diffusion probabilistic models on SO(3) for rotational alignment**, A. Leach, S.M. Schmon, M.T. Degiacomi, C.G. Willcocks, In Workshop on Geometrical and Topological Representation Learning, ICLR 2022 (Contributing to Chapter 4).

Furthermore, the following peer-reviewed publications have been co-authored but have not been included as part of the narrative of this thesis:

- **Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models** Sam Bond-Taylor, A. Leach, Y. Long, C.G. Willcocks, In PAMI 2022.
- **AnoDDPM: Anomaly Detection with Denoising Diffusion Probabilistic Models using Simplex Noise**, Julian Wyatt, A. Leach, S.M. Schmon, C.G. Willcocks, In NTIRE workshop, CVPR 2022.

1.4 Thesis Scope and Structure

This thesis presents a number of topics spanning protein kinematics, generative modelling, probabilistic modelling and statistics on manifolds. In particular, generative modelling applied to protein interactions can be used to reframe the problem of predicting the correct interaction into generating potential interactions to further evaluate and score. Chapter 2 thoroughly reviews a broad array of topics crucial to the application of generative modelling to the protein domain. In this chapter, this broad array of topics is divided into proteins, transformers, geometric deep learning and energy based models.

Chapter 3 explores the use of an enhancement to traditional computational protein docking seed points by fast generation of potential docking poses. Employing GPU-accelerated parallelised raycasting across the entire protein surface, large speedups in initial docking position generation are achieved. These docking positions

can then be fed into existing docking optimisers for faster evaluation of potential docking candidates.

Chapter 4 explores the usage of denoising diffusion probabilistic models for modelling the distribution of candidate solutions for 3D rotation alignment problems. Systemic modifications to the vanilla DDPM model to include information relevant to the underlying $SO(3)$ manifold are applied. This includes replacing the gaussian distribution in the diffusion process with a carefully chosen gaussian alternative distribution on $SO(3)$, called the Isotropic Gaussian on $SO(3)$. With this distribution, along with appropriate replacements of euclidean operations like addition with the geodesic flow, this chapter successfully redefines the Denoising Diffusion Probabilistic Model (DDPM) equations to learn distributions defined on $SO(3)$. Finally, the chapter benchmarks this novel $SO(3)$ DDPM with regular euclidean DDPM (applied on quaternion rotations) to learn the distribution of candidate solutions in rotation alignment.

Chapter 5 explores the application of deep generative modelling and the $SO(3)$ diffusion model in Chapter 4 to rigid protein docking. Several issues such as the correct choice of model scaling, method of $SE(3)$ gradient prediction, and network architecture are discussed.

Chapter 6, the final chapter, draws together the key findings presented within the previous chapters, provides a discussion on the strengths and weaknesses of diffusion models for protein docking, and presents a brief overview of derivative works in which techniques developed in Chapter 4 have been applied to protein interaction and other areas.

CHAPTER 2

Literature Review

This chapter forms a review of relevant literature to this thesis. Firstly, an overview of proteins, their synthesis, sequence and structure is provided, along with current approaches to protein docking and how machine learning algorithms have been adapted and applied to protein docking. The second section is an overview of Transformer models, and how they are applied to protein sequence data. Transformer-derived attention mechanisms are used in many of the methods discussed in Section 2.3. These methods can be applied to 3D structures such as proteins. In the final section, Energy Based Models, an overview of these models and geometric learning approaches used on similar tasks are discussed.

2.1 Proteins

Proteins are a vital part of living organisms, performing a vast array of functions, including catalysing metabolic reactions, providing structure for cells and connective tissue, replicating DNA, responses to external stimuli (light, sound, pressure, smell and taste), cell-to-cell communication via hormones, and transportation of molecules. Virtually all reactions inside an organism are the result of the interac-

tions between proteins, either through short-term interactions such as enzymatic catalysis and signal transduction, or long term construction of protein complexes that function as macromolecular machines. Analysis of proteins and how they interact is critical in understanding how the human body works, and how infectious diseases can spread.

2.1.1 Protein Biosynthesis

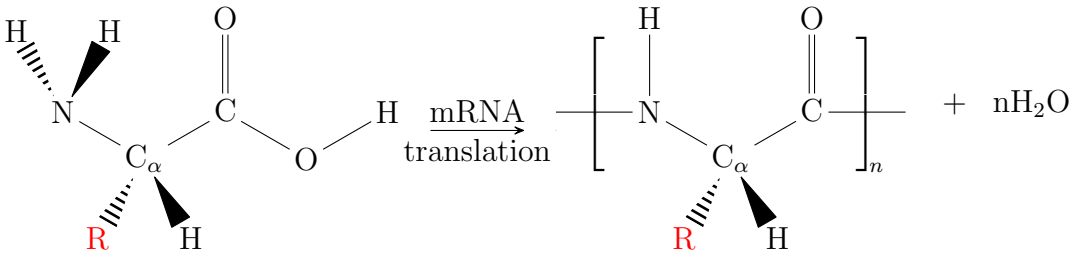


Figure 2.1: Generic amino acid structure, the "R" atom corresponds to the sidechain that differs between each type of amino acid. The two carbon atoms are typically distinguished by labelling the one attached to the sidechain as "carbon alpha" Translation of messenger RNA (mRNA) results in a protein backbone consisting of many residues linked together.

Initially a section of DNA encoding a protein (a gene), is transcribed into mRNA. Premature form mRNA is produced through the action of RNA polymerase enzymes on a DNA strand, "unzipping" the strand and "copying" it onto a section of mRNA. This then goes through a post-transcription modification phase, where caps are added to each end, and sections of the sequence that don't need to be encoded into the protein (introns) are removed. After the transcription phase, the mature mRNA is then translated into a sequence of amino acids (residues) chemically bonded by covalent bonds known as "peptide bonds" (Figure 2.1). Amino acids are the organic compounds that form the building blocks of proteins. Each 3-base sequence of mRNA encodes a particular amino acid into the protein's backbone. Each amino acid has a unique side chain, the different side chains result in different behaviours, some have electrostatic charges, while others are hydrophobic. The interaction between side chains, each other and the water around them determines the shape of the folded protein. This is known as the "Sequence Hypothesis" [14] and can be summarised as "sequence determines structure".

2.1.2 Protein Structure

Protein interaction is determined by many factors, including electrostatic forces, hydrogen bonding and hydrophobic effects. The atomic composition and overall structure of a protein also affects how and what a protein interacts with. Direct imaging of protein structure is difficult, and while methods such as cryo-EM and X-ray crystallography exist and can produce images of the protein's overall shape, determining the corresponding residues present is difficult and requires computational and statistical approaches in order to predict the complete structure of a protein.

Protein structure is typically defined into four hierarchical categories: primary, the linear sequence of amino acids; secondary, the three dimensional form of local segments of a chain; tertiary, the three dimensional shape of a single chain; and quaternary, the three dimensional shape formed by multiple protein chains.

Primary Structure

The primary structure of a protein (the residue sequence) can be determined directly through protein sequencing, or indirectly by DNA sequencing of the corresponding gene. By convention, the primary protein structure is listed starting from the unbonded nitrogen atom (N-terminal group).

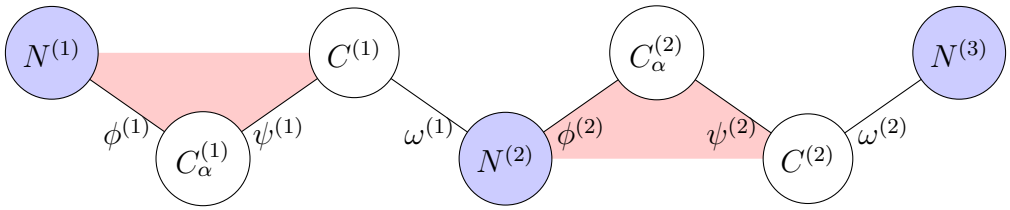


Figure 2.2: Protein backbone shape, each triangle corresponds to a residue.

Secondary Structure

Secondary protein structures are formed out of local residue sequences (Figure 2.2). The formation of secondary structure is the first step in the folding process that a protein takes. Dihedral (torsion) angles, specified by the ϕ, ψ and ω in Figure 2.2 are used to describe the twisting of the protein backbone. Dihedral angles are

defined as the angle between the two planes formed by adjacent atoms (Figure 2.3a). The peptide bond $(C-N)$ linking two consecutive residues corresponds to the ω dihedral angle and almost always exists in the *trans* orientation [15], corresponding to an angle of 180°. In contrast, the ϕ and ψ angles can take on a wide range of values. The distribution of ϕ and ψ angles can be evaluated through the use of a Ramachandran plot [16]. Plotting ϕ and ψ angles taken from a variety of proteins shows that these angle pairs fall into regions of high and low density (Figure 2.3b). The distribution of angle-pairs is strongly influenced by the local residue sequence - i.e. the orientations that are energetically favourable are dependent on the electrostatic influence of nearby sidechains.

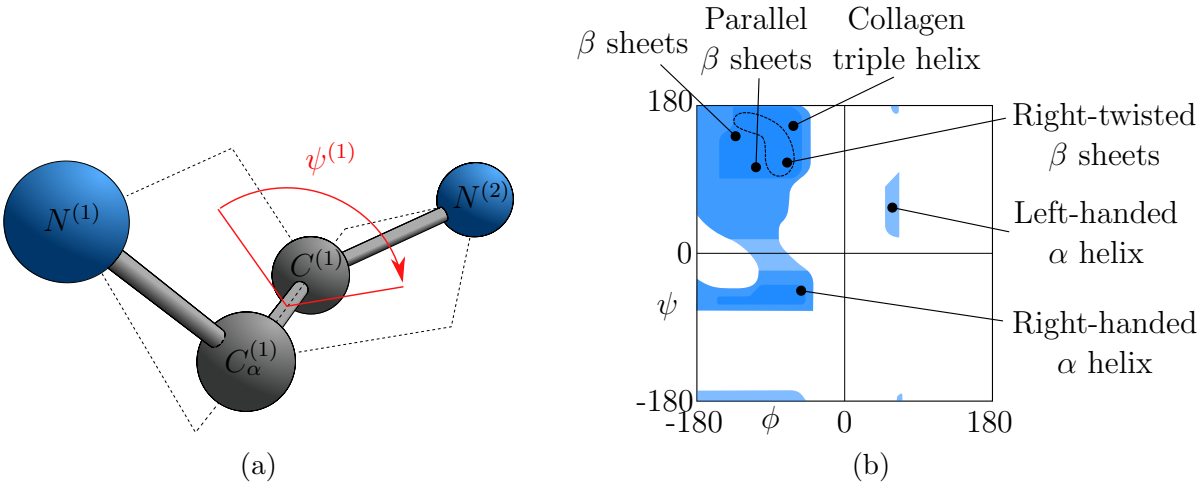
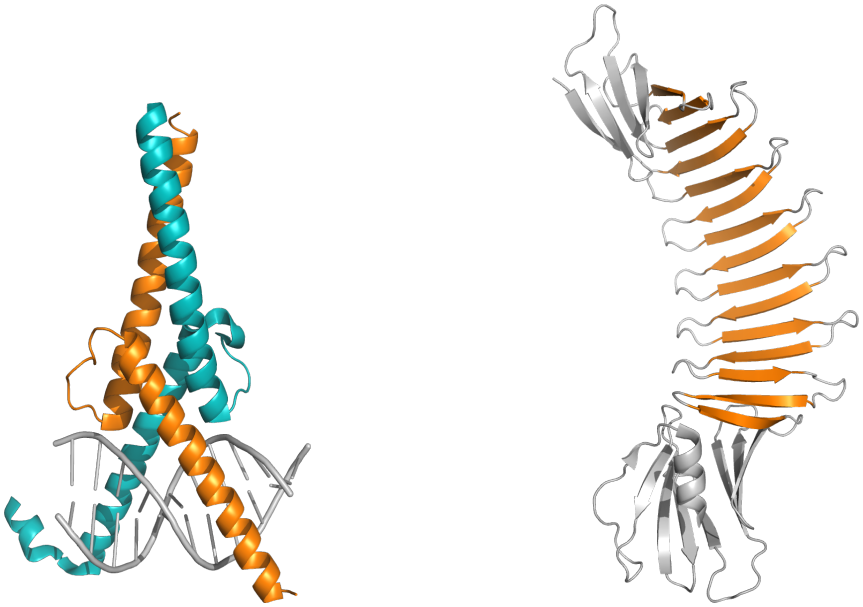


Figure 2.3: (a) Calculation of dihedral angles. The angle $\psi^{(1)}$ is calculated between the two planes defined by $(N^{(1)}, C_\alpha^{(1)}, C^{(1)})$ and $(C_\alpha^{(1)}, C^{(1)}, N^{(2)})$. (b) A Ramachandran plot. Allowed regions (blue) correspond to $\phi-\psi$ angle pairs that are commonly seen in protein structures. Labels indicate the approximate positions of major secondary structures. Reproduced from [17]

There are two main secondary structures that are commonly found within proteins: alpha helices and beta sheets (Figure 2.4). These configurations are particularly energetically favourable, as their structure is reinforced by the formation of intramolecular hydrogen bonds between residues (Figure 2.5).

Tertiary Structure

Alpha helices and Beta sheets typically contain hydrophobic and hydrophilic sections (they are amphipathic). These repulsions and attractions to water influence which



(a) α-helices in the "Leucine Zipper", a common structural motif.

(b) Extended β-sheet in a mutated form of OspA, highlighted in orange.

Figure 2.4: Protein secondary structures, such as α-helices and β-sheets are the two main secondary structures present in naturally occurring proteins.

parts of the secondary structures end up exposed to the environment surrounding the protein, and which parts are more likely to end up embedded deep within the folded protein. The attraction to water of a section of secondary structure is dependent on the residue sidechains.

Quaternary Structure

Proteins such as Haemoglobin consist of multiple peptide chains. In these proteins, the assembly of pre-folded single-chain subunits gives rise to quaternary structure. Subunits must be correctly arranged and each one correctly encoded in order for the entire protein to function properly. Misalignment of the subunits can cause a severe loss of ability for a protein to interact. In Aldehyde Dehydrogenase 2 (ALDH2) a tetrameric protein, one of several proteins used during the metabolic process for ethanol, a mutation of a single nucleotide results in Lysine instead of Glutamine being encoded as the 487th residue. This section of the folded structure is responsible for the binding of four subunits to form a stable protein. Unfortunately the mutation results in the subunits being incapable of binding correctly, resulting in a vastly

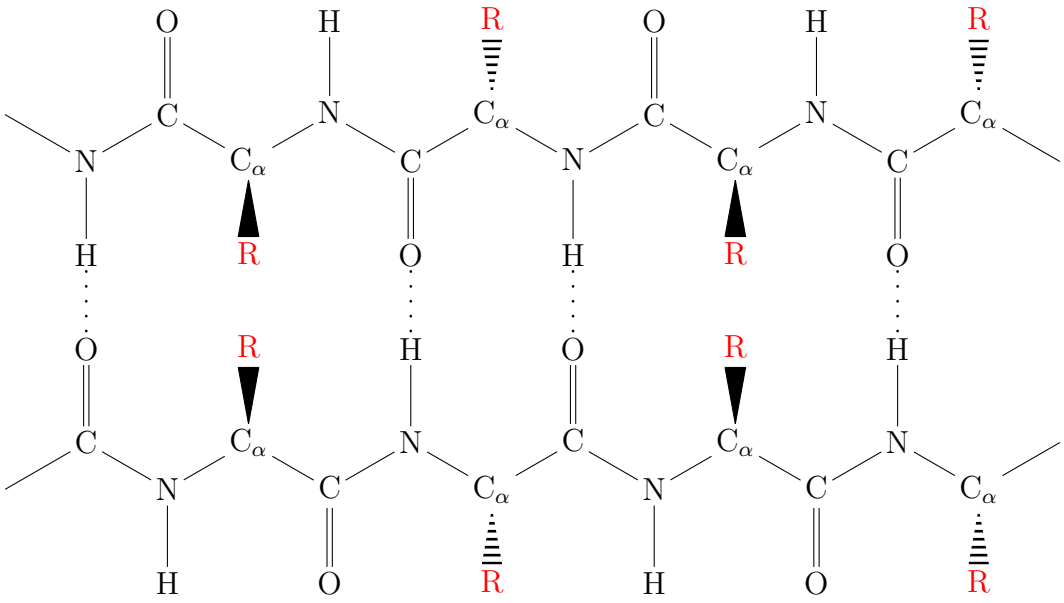


Figure 2.5: Hydrogen bonding in an antiparallel beta sheet. Adjacent chains of residues form intramolecular hydrogen bonds (...), forming a stable, flexible, structure.

reduced ability to process aldehyde, with versions of ALDH2 containing four mutated subunits being completely incapable of processing aldehyde [18]. This results in significant reduction in the human body's ability to process alcohol, resulting in the alcohol flush reaction, and highlights the importance of structural protein modelling.

Protein Rigidity and Intrisic Disorder

While the previous sections have discussed protein structure as a hierarchical folding process culminating in a single, solid structure, it's important to note that protein structure is not fixed, that the intramolecular forces holding the protein conformation are not static, and that proteins exist on a sliding scale of rigidity. A significant proportion of protein interaction and behaviour is driven by the flexibility of proteins as they interact with their environment. Fundamental biological processes such as cellular cargo transport (via kinesin proteins) require flexibility in the motor domain in order to function [19]. Similarly, various environmental factors such as pH and dissolved carbon dioxide level affect the shape of haemoglobin, switching it between two states - taut (T) and relaxed (R) which influence its oxygen affinity, causing the release of oxygen in high CO_2 environments. Additionally, the initial binding

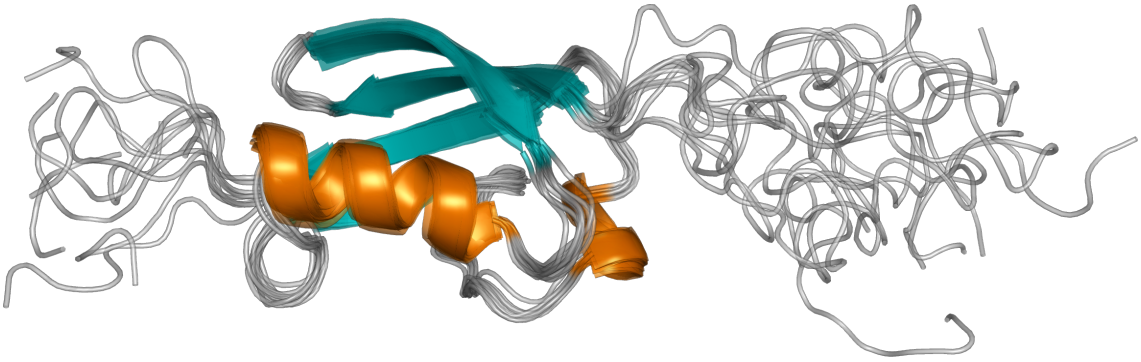


Figure 2.6: Intrinsic disorder in SUMO-1 protein. Ten alternative conformations derived from protein NMR are overlaid. These structures show an orderly central structure consisting of α-helices (orange) and β-sheets (teal). The N- and C- terminal regions show intrinsic disorder and do not assume a fixed shape. Images derived from PDB: 1A5R

of an oxygen atom induces a slight conformal shift, further increasing the oxygen affinity [20].

Intrinsically Disordered Proteins (IDPs) lack a fixed three-dimensional structure. These proteins have been found to be important in DeoxyriboNucleic Acid (DNA) regulation [21] and cell signalling [22]. The existence of IDPs disproves the idea that a protein must form a fixed three-dimensional structure in order to accomplish their biological functions, and also refutes the utility of rigid-body protein docking analysis as a catch-all method of reasoning about protein interaction.

2.1.3 Protein Folding, a Thermodynamic Perspective

Protein folding is a complex process, for which the space of possible protein shapes in extremely large. Nevertheless, proteins tend to follow specific folding pathways. In order to understand this apparent contradiction, the space of all possible protein configurations can be treated as a statistical energy landscape [23]. From this viewpoint, the folding of a protein can be seen as a form of stochastic gradient descent of the protein's configuration towards the minimum energy state. This process is driven by intramolecular forces such as van der Waals forces and hydrogen bonds. As a protein's natural environment consists of being surrounded by water and other molecules, the buffeting effects of brownian motion add a form of stochasticity to this gradient descent process, helping to overcome potential local minima in protein

configuration. The tendency of proteins to follow specific folding pathways can be ascribed to a "valley" effect. Much like how the shape of valley directs water from a large area into a single stream, a large swathe of protein configurational space can be captured by a single folding pathway that the protein will then follow. The *principle of minimal frustration* [23] states that it is evolutionarily favourable for folding landscapes to minimise the number of local minima and to increase the speed of protein folding, and therefore naturally occurring proteins and their landscapes should show these properties.

2.1.4 Protein-Protein Docking and Interaction

Protein-protein docking is a focal point of activity in computational biophysics and structural biology. In general, the structure of a protein complex is far more difficult to determine through experimentation than the structure of an individual protein [24]. Thus, computational techniques to model these interactions are paramount to biological understanding.

Classical Pipeline

Protein docking methods typically run over several steps. Firstly, a population of protein complexes is generated, on the order of thousands to millions of candidates, treating the individual proteins as rigid bodies. Secondly a scoring function is used to rank candidates until finally, a refinement process is used to optimise top ranked candidates with energy or geometric models.

Random sampling of the potential docking space is unlikely to produce high quality samples. Many different methods exist that seek to produce a higher quality initial population. Many methods rely on matching coarse resolution shapes of ligand and receptor against each other, achieved through a variety of strategies. An early breakthrough in searching for these matches by Katchalski-Katzie et al. [25] relied upon Fast Fourier Transform (FFT) methods to convolve the shapes of the two proteins, extensions to this technique such as FTDock [26], DOT [27], HEX [28], GRAMM [29] and ZDOCK [30] introduce electrostatic information of the two proteins to further filter the generated potential candidates. An alternate approach

is geometric hashing [31], in which each point in the target proteins is assigned an invariant representation before comparing them for similarity. Agreements between representations indicates a match between regions in the two proteins, and hence a potential docking site. Other geometric hashing methods such as LZerD [32] rely upon different geometric hashing strategies for better discrimination, and the method can be extended with templating methods such as PatchDock and SymmDock [33], and SnapDock [34] - querying a database of known dockings and associated geometric hashes of other proteins to infer potential candidates.

After (or during) potential candidate generation, scoring functions are used to rank the top candidates, before a refinement process is applied. This refinement process is typically approached as a black-box optimisation problem while trying to maximise a scoring function (or minimise an energy function). A variety of optimisation algorithms are used. In 2001, Gardiner et al. [35] proposed the use of genetic algorithms, with each chromosome representing a different degree of freedom in the ligand's movement. Other black-box optimisation algorithms proposed for refinement include: FPDock [36], applying Flower Pollination Algorithm (FPA) [37], where the ligand position is represented by a quaternion, scored by electrostatic and van der Waals potential. SwarmDock [38] and PSO@AUTODOCK [39] both use Particle Swarm Optimisation (PSO) [40] as an approach. This is extended by JabberDock [41, 42] by a "kick reseed" strategy initially proposed in [43] to randomly reinitialise the particle positions. LightDock [44] uses Glowworm Swarm Optimization (GSO), whose main benefit is the ability to optimise multiple minima simultaneously, helping to avoid being trapped in local minima.

Machine Learning Approaches

Machine learning approaches to Protein-Protein interaction fit into two main categories. The more mature approach, utilising a variety of techniques and not just neural networks, is to augment or replace parts of the classical pipeline with learned procedures. Alternatively, the entire pipeline can be replaced with deep learning methods. The standard textbook view holds that the residue sequence of a protein determines its structure. Based upon this view, an ideal neural network would be

capable of generating a full protein structure given a sequence of amino acids. In general, structure prediction is mainly concerned with the position of the protein's backbone atoms, determined by a repeating sequence of Nitrogen (N), Carbon-alpha (C_α) and Carbon (C) atoms. i.e.

$$\underbrace{N^{(1)}, C_\alpha^{(1)}, C^{(1)}}, \underbrace{N^{(2)}, C_\alpha^{(2)}, C^{(2)}}, ..., \underbrace{N^{(L)}, C_\alpha^{(L)}, C^{(L)}},$$

for a protein of length L. As shown in Figure 2.1, a (N, C_α, C) grouping corresponds to a particular residue, of which many are chained together by covalent bonds to form the protein backbone. (Figure 2.2).

Pipeline Augmentation

The classical pipeline can be augmented in one of two ways. Either the population of candidates can be improved through the use of data driven approaches, or the scoring mechanism can be replaced with a data driven model that prioritises candidates with interface sites similar to known protein interactions. PAIRPred [45] uses sequential and structural features as input to a pairwise Support Vector Machine (SVM) to predict the residue pairs that are likely to be bound together. This helps select potential candidates. Similarly, EL_SMURF [46] uses random forests to predict the binding sites. Due to class imbalance (most residues in a protein are not part of the binding site), Wang et al. also developed SMOTE as a data augmentation tool. Wei. et al. [47] approach class imbalance differently, an SVM classifier is used to determine the weights of training samples, before using those estimated weights in a random forest for binding site prediction. BIPSPI [48] uses XGBoost [49] for binding site prediction. This approach takes into account both receptor and ligand residues during prediction to produce partner-specific binding sites.

Neural network based approaches have also been considered, Fout et al. [50] uses Graph Convolutional Networks (GCNs) to generate local residue features before pairwise concatenation between receptor and ligand residues is classified as being a binding site. Other deep learning based approaches such as DLPred [51] use an Long Short Term Memory (LSTM), operating on sequence data only to predict binding

sites. As with most deep learning problems, large datasets significantly improve the quality of learned functions. In 2019, SASNet [52] introduced a significantly larger dataset - the Database of Interacting Protein Structures (DIPS), two orders of magnitude larger than datasets previously used. The SASNet model consists of voxelisation of residue sidechains, a small convolutional network, and pairwise comparison of residue features. While this method lacks contextual information about the residue neighbourhood, the model produced state of the art results. Other approaches use convolutional layers to produce per-residue feature descriptors that can then be compared against known binding site "hotspots". Xie et al. [53] introduces a Convolution-only binding site prediction method that relies on pre-calculated spatial features to improve performance. Zhu et al. [54] also introduce a convolution-only binding site prediction model, with improved performance due to the use of model ensembles. A database of known residue sequences and bindings is used to improve performance at runtime. Liu et al. [55] uses both graph neural networks and convolutional layers to produce high quality predictions of binding site.

It is also possible to use machine learning models to estimate the scores of a given docking candidate rather than relying on full score calculation. Classical machine learning approaches to this problem such as SVMs [56–58] have been attempted. While deep learning approaches such as DOVE [59], based on 3D convolutional networks, and EGCN [60] using graph convolutional networks and spatial relationships between residues have also attempted to approximate score.

In Hadarovich et al. [61], the contact map (a 2D representation of which residues are close together in 3D space) between two protein chains is predicted with a convolutional architecture applied to the distance map. The contact map is then used to predict the alignment via gradient descent. The distance and contact maps are invariant to rotation and translations.

2.1.5 Protein Representation in Neural Networks

The best way of representing protein structure in a neural network is highly dependent on task. Protein structures have strong constraints on atomic positions. Distances between covalently bonded atoms and angles formed by adjacent bonds

are typically treated as fixed. For tasks where protein structure is known, such as classification, representation of the protein structure is free to include more degrees of freedom than are present in the molecule. For tasks that involve the prediction of protein structure, these constraints need to be enforced. Many different approaches to this exist. In some, the network is free to predict any representation, then constraints are forced upon the prediction. In others, the constraints are intrinsic to the model, such that impossible configurations cannot be predicted.

XYZ Coordinates of Atoms

Each residue in a backbone consists of $N^{(i)}$, $C_\alpha^{(i)}$, $C^{(i)}$ atoms. We can represent the position in 3D space of each atom with a set of X,Y,Z coordinates ($\mathbb{R}^3$). By stacking these coordinates together, a residue can be represented as a 3 by 3 matrix - $\mathbb{R}^{3\times3}$. We can then further stack the residues of a whole backbone of length L together to arrive at an array X:

$$X \in \mathbb{R}^{3\times3\times L}. \tag{2.1}$$

Applications of this approach need to enforce bond lengths and angles between adjacent atoms. Ramaswamy et al. [62] unravel their initial protein representation into $\mathbb{R}^{3\times3L}$, as to leverage existing 1D convolution operations present in popular libraries. Their physical loss term, $\mathcal{L}_{\text{Phys}}$ (an implementation of the AMBER force-field [63]) contains parameters that enforce the bond length and bond angle constraints.

XYZ Coordinates of C_α and Orientation

Noting that covalent bond lengths and angles show minimal variation across protein structures [64], each residue can be approximated by keeping track of its orientation and the coordinates of the C_α atom. Constraints on distance and angle between adjacent residues still need to be satisfied, however this does reduce the amount of constraints that need to be learned or externally enforced. This approach is used in Alphafold 2 [65].

Dihedral Angles

Noting that bond length and angles are almost entirely fixed [64], the protein backbone's 3D structure can be almost entirely described through dihedral angles alone. Alphafold 1 [66] initially predicts a combination of dihedral angles and distance as an initial solution before optimising the relative distance potentials. However, the prediction of dihedral angles alone also introduces a "lever arm" effect, as small changes in a dihedral angle's value can greatly influence protein shape.

Distance Maps

Instead of representing atomic positions, we can instead represent the distances between pairs of atoms. True atomic positions (modulo rigid transformation) can be recovered through multidimensional scaling [67]. This approach has the advantage of learned features being invariant to rotation and translation of the protein complex, reducing the need for data augmentation. Distance maps allow for easy modelling of both long and short range interaction, as presenting pairwise information in an L^2 map means that standard Convolutional Neural Networks (CNNs) can be leveraged. Alphafold 1 [66] initially predicts distances between atoms in the form of a distance map.

2.2 Transformers

Transformers are a form of neural network that rely on an attention mechanism [68] in order to weight the contributions of a set of input tokens to the output. While transformers were initially used to great success as a language model [68], and continue to be the dominant form of model in Natural Language Processing (NLP) [69–71], the use of attention mechanisms has become widespread in computer vision [72–74] and machine learning models for proteins [75].

2.2.1 Attention Mechanisms

The core of any transformer model is the attention mechanism [68]. The attention mechanism can be seen as a way of mapping a query and a set of key-value pairs to an output. An attention mechanism defines an interaction between them such that the output for each query token is a sum of the values weighted by dot-product similarity between the query and keys. In Self-attention, the query, key and value vectors are derived from the same set of input vectors. Self attention takes in a set of values X, of length L for which each element (the vector $\vec{x_i}$) is multiplied by three different matrices.

- W^Q - the Query weight matrix to produce $\vec{q_i}$, a query vector
- W^K - the Key weight matrix, to produce $\vec{k_i}$, a key vector
- W^V - the Values weight matrix, to produce $\vec{v_i}$, a value vector

For each query vector $\vec{q_i}$, we take the dot product with every key vector $[\vec{k_1}, ..., \vec{k_L}]$, producing a vector of scores

$$[s_{i,1}, ..., s_{i,L}] = \vec{q_i} \cdot [\vec{k_1}, ..., \vec{k_L}]$$

We then divide the score values by $\sqrt{d_k}$, where d_k is the dimension of the key vectors in order to normalise the selectivity of the dot product and stabilize gradients. Softmax is then applied in order to normalise the attention to sum to 1. This results in a vector $\vec{a_i}$ of length L for which each value is the weighted contribution of each $\vec{v_i}$ to the final output $\vec{z_i}$:

$$\vec{a_i} = \text{softmax}\left(\frac{[s_{(i,1)}, ..., s_{(i,L)}]}{\sqrt{d_k}}\right), \tag{2.2}$$

$$\vec{z_i} = \sum_{j=0}^{L} a_{ij}\vec{v_j}. \tag{2.3}$$

We can also write this in matrix notation for the whole sequence, where:

$$W^Q X = Q, \tag{2.4}$$

$$W^K X = K, \tag{2.5}$$

$$W^V X = V, \tag{2.6}$$

which gives us the original notation:

$$Z = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V. \tag{2.7}$$

2.2.2 Cross Attention

While many attention-based architectures consider a single set of inputs, Sequence-to-Sequence tasks such as machine translation and certain protein-protein docking algorithms such as EquiDock [75] rely upon two separate input sets. In these cases, a Cross-Attention mechanism relies upon queries from the second set of inputs. For an element in the encoding $\vec{e_i}$ and a decoder input $\vec{x_i}$, a decoder head produces the following values:

$$\vec{q_i} = W^Q \vec{x_i}, \tag{2.8}$$

$$\vec{k_i} = W^K \vec{e_i}, \tag{2.9}$$

$$\vec{v_i} = W^V \vec{e_i}, \tag{2.10}$$

these are then used as in the standard self-attention mechanism.

2.2.3 Multi-head Attention

In multi-head attention, we have multiple attention "heads". These are identical in structure, but have different weight matrices. The outputs from these - $Z^0, Z^1, ...Z^m$ are concatenated together and multiplied by a final weight matrix.

2.2.4 Kernelisable Attention

While powerful and achieving state of the art accuracy in a wide variety of tasks, transformers are limited by large memory usage during training. This is due to the attention matrix consisting of n^2 entries for a sequence of length n. Given an attention matrix

$$\boldsymbol{A} = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right),$$

kernelisable attention methods assume that the matrix $\boldsymbol{A}$ can be rewritten such that each element $\boldsymbol{A}_{i,j}$ can be defined in terms of a kernel $K(\cdot,\cdot)$ and per-query scaling factor c_i:

$$\boldsymbol{A}_{i,j} = c_i\, K(\vec{q_i}, \vec{k_j}).$$

Kernel functions rely upon the property that they can be expressed as an inner product over vectors transformed by a function ϕ.

$$\boldsymbol{A}_{i,j} = \phi(\boldsymbol{q}_i) \cdot \phi(\boldsymbol{k}_j),$$

with $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ standing for the i^{th} and j^{th} query and key row-vector respectively. Kernel methods are able to extend this by a process known as the "kernel trick" [76]. This allows for defining functions between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ that cannot be defined in terms of a dot product. For example, the Radial Basis Function (RBF) kernel $K(\boldsymbol{q}, \boldsymbol{k})$ is defined in terms of an exponential transform of the euclidean distance between the two values:

$$K(\boldsymbol{q}, \boldsymbol{k}) = \exp\left(-\frac{\|\boldsymbol{q} - \boldsymbol{k}\|^2}{2\sigma^2}\right),$$

where σ is a parameter determining how quickly the value of $K(\boldsymbol{q}, \boldsymbol{k})$ tends to zero as $\|\boldsymbol{q} - \boldsymbol{k}\|^2$ increases. However, as is important in kernelisable attention mechanisms, kernels without an explicit ϕ can be approximated [77] with a finite-dimensional embedding. Defining Q' and K' as query/key matrices with $\phi : \mathbb{R}^{d_k} \to \mathbb{R}^r$ as a non-linear mapping to a higher dimensional space performed on each row-vector, the full softmax attention mechanism can be approximated:

$$\text{softmax}\left(QK^\top\right)V \simeq (Q'K'^\top)V = Q'(K'^\top V).$$

In this approximation, Q', $K'^\top$ and V have dimensions $n \times r$, $r \times n$ and $n \times d_k$ respectively. Due to the associativity of matrix multiplication, the approximation of the attention matrix $\boldsymbol{A}$ does not have to be explicitly calculated. The size of the matrix product $(K'^\top V)$ is then independent of the length of the sequence, with size $r \times d_k$. The memory requirements of the full attention approximation are then linear with the sequence length. This allows for significantly longer sequences to be modelled without the memory constraints associated with standard attention mechanisms.

As part of showing that autoregressive transformers can be approximated by Recurrent Neural Networkss (RNNs), Katharopoulos et al. [78] implement the feature map $\phi(\cdot)$ as a linear projection to a high-dimensional space followed by a smooth, unbounded non-linearity such as ELU or GELU [79, 80]. Furthermore, Katharopoulos et al. go on to show that by linearising the attention mechanism, any autoregressive transformer can be expressed as an RNN with the following update equations, where x_i denotes the i-th input, y_i the i-th output, and s_i, z_i the hidden state at timestep i:

$$\begin{aligned} s_0 &= 0, \\ z_0 &= 0, \\ s_i &= s_{i-1} + \phi\left(x_i W_K\right)\left(x_i W_V\right)^T, \\ z_i &= z_{i-1} + \phi\left(x_i W_K\right), \\ y_i &= f_l\left(\frac{\phi\left(x_i W_Q\right)^T s_i}{\phi\left(x_i W_Q\right)^T z_i} + x_i\right). \end{aligned} \tag{2.11}$$

This allows for for real-time symbol-by-symbol evaluation, with constant memory usage with respect to sequence length, and linear inference time. However, Katharopoulos' choice of $\phi(\cdot)$ is somewhat limiting, and does not perform as well as standard attention. In Rethinking Attention with Performers [81], Choromanski et al. investigate functions to approximate standard softmax attention. Issues with ap-

proximation using trigonometric functions are identified, with the potential for negative valued approximations resulting in an inability to train. Instead, Choromanski proposes Fast Attention Via positive Orthogonal Random features (FAVOR+) a modification of the commonly used random fourier feature method [77, 82–84] of kernel approximation. FAVOR+ avoids negative valued approximations by using exponential functions ($\exp(x), \exp(-x)$) instead of the trigonometric functions ($\sin(x), \cos(x)$) used to approximate kernels in Rahimi et al. [77]. As these exponential functions are always positive, the approximation made using them will also be positive. To further improve the approximation by reducing variance, the random feature projections are forced to be orthogonal to each other. In the case of more requested features than dimensions in the projection, a second set of random orthogonal features are considered. Performer models are typically implemented by fine-tuning existing language models, which can then be extended with longer context windows given the savings in memory. When training, it's suggested to resample the random feature vectors every so often to prevent overfitting.

2.2.5 Protein Language Models

Transformer based models are commonly applied to NLP tasks. However, protein sequences can also be considered as a series of tokens, similar to natural text. This has lead to much research into Protein Language Models (PLMs). The basics of a PLM are very similar to those of an NLP model. The protein sequence is tokenized by splitting it up into its individual residues. PLMs are typically trained in an unsupervised manner on protein sequences, both autoregressively through next token prediction and through masked token prediction i.e. BERT [85]. While autoregressive token prediction can be used for the generation of novel protein sequences [86, 87], a key use of unsupervised PLMs is in learning embeddings of protein sequences that can be used for other downstream tasks. Rao et al. [88] shows that unsupervised PLMs are capable of learning structural information through sequence alone. Rives et al. [89] expands on this by training a larger model on more data, showing that features such as secondary and tertiary structure can be predicted from PLM embeddings. Furthermore Elnaggar et al. [90], train a large collection of different

unsupervised autoregressive (Transformer-XL [91], XLNet [92]) and autoencoding (BERT [85], Albert [93], Electra [94], T5 [95]) language models on protein sequences. These showed the ability of learned protein embeddings to capture biophysical features and to perform better on downstream tasks. Following the introduction of new adversarial masking techniques in NLP [94,96], McDermott et al. [97] apply the same techniques to protein sequences, showing stronger performance of these models on the same benchmarks. While not directly concerned with protein structural modelling, there is a strong history of NLP advances being repurposed to improve the state of the art in protein sequence modelling.

2.3 Geometric Deep Learning

Proteins occupy an interesting space in machine learning, where many techniques from more theoretical areas such as geometric and topological learning can be applied. The basic sequence of a neural network can be expressed as a Graph Neural Network (GNN) over the individual atoms that make up each residue, with edges representing covalent bonds. From a manifold and group theory perspective, protein structure is also interesting as it follows the symmetries of $SE(3)$ - the group of rotations and translations of 3D space. Proteins do not follow the symmetries of $E(3)$ the group of rotations, translations and reflections of 3D space, as protein structure is chiral. Invariance to reflection is undesirable in these problems as the chirality of molecular structures has a significant effect on biological interactions (notably thalidomide). A protein's overall structure and behaviour is not affected by its position or orientation in 3D space and thus shows invariance in $SE(3)$, however interactions between residues and between proteins are heavily driven by their relative orientations and positions.

Capturing graph and $SE(3)$ structure and symmetries as part of the network architecture leads to reductions in parameter count and better generalisation for the same reasons that lead to CNN architectures excelling at image and volumetric tasks. This section serves as an introduction to the terminology and concepts needed for an understanding of deep learning protein models. This section will rely upon

the categorisations proposed by Bronstein et al. [98]

2.3.1 Graph Neural Networks

Graphs consist of a set of vertices $v_i \in \mathcal{V}$ and a set of (potentially directed) edges $(v_i, v_j) = e_{ij} \in \mathcal{E}$. Each vertex v_i has associated with it a set of features: inputs, referred to as $\mathbf{x}_i$ and hidden/output layers referred to as $\mathbf{h}_i^l$ where l is the layer index. Some networks also include edge information. GNNs are a form of neural network that operate on graphs by applying learned functions to edges and vertices. Most importantly, the ordering that is used when vertices are processed by the neural network does not affect how they are processed. Any GNN layer will produce a set of values that are influenced by both vertex values and the connectivity structure of the graph.

For ease of distinction, network layers presented in this section will use $\mathbf{x}_i$ to denote the input features for a vertex and $\mathbf{h}_i$ to denote the output features for a vertex of a layer. This does not lose generality as layers can be stacked to produce networks of greater capacity.

GNNs update vertex features $\mathbf{x}_u \rightarrow \mathbf{h}_u$ by applying a permutation invariant function to the features of each vertex and its neighbours – $\phi\left(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u}\right)$. This function can take one of three forms; convolutional, attentional, or message-passing. Each of these methods follows the same basic structure: Firstly, calculate features for each of the adjacent vertices, then aggregate them with a permutation invariant operation $\bigoplus$ e.g. sum, mean, max, min, etc.

Convolutional GNNs

In Convolutional GNNs, also known as GCNs, the basic structure of a network [99–101] relies on applying a non-linear function ϕ to every vertex u and an aggregation of its neighbours such that each of its transformed neighbours $\phi(\mathbf{x}_v)$ has an equal contribution. Each contribution is scaled by the same fixed term c_{uv} that is non-zero when the edge e_{uv} exists. The final output can be viewed as a learned function ϕ of

the input vertex data and the aggregated, transformed neighbours:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{uv}\psi(\mathbf{x}_v)\right).$$

Attentional GNNs

Attentional GNNs replace the fixed weight c_{uv} with a scalar value derived from the vertex and its neighbour $a(\mathbf{x}_u, \mathbf{x}_v)$, this learned function is derived from a modification of the attention mechanism present in transformers [102–104]. This value is often softmax-normalised across all neighbours and $\bigoplus$ taken as summation. This allows the weights to be feature dependent, updating $\mathbf{x}_u$ by learning what features are potentially relevant:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v)\psi(\mathbf{x}_v)\right).$$

Message Passing GNNs

Message-passing GNNs compute arbitrary vectors for each $\phi(\mathbf{x}_u, \mathbf{x}_v)$ pair. The key difference between message-passing and attentional GNNs is the magnitude *and direction* of the update vector ($\phi(\mathbf{x}_u, \mathbf{x}_v)$) is dependent on the value $\mathbf{x}_u$ in message-passing networks, whereas only the magnitude of the update vector is dependent on $\mathbf{x}_u$ in attentional GNNs.

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

The increased flexibility of message-passing models is particularly useful when looking at graphs over points in 3D space. The messages passed between nodes can be made dependent on the relative positions between them, allowing for richer feature updates; especially when node features encode non-scalar information. It should also be noted that each of the stated GNN mechanisms is a strict subset of the following one. Attentional GNNs can approximate Convolutional GNNs with a "flat" attention, and Message Passing GNNs can approximate Attentional GNNs by

making the direction of the update vector invariant to $\mathbf{x}_u$.

There are two main motivations for the use of GNNs in modelling protein-protein interaction. Firstly, molecular structure consists of many covalent bonded atoms, which can be easily interpreted as a graph structure with each atom representing a node, and each bond an edge. Secondly, computational limits mean that fully modelling the interaction of every atom with every other atom in a problem can be intractable due to $O(N^2)$ time and memory complexity. Therefore the assumption is made that atomic interactions can be modelled locally and that many layers of local interactions can capture the long-range interactions of each atom.

In E(n) (n-dimensional euclidean translations, rotations and reflections) equivariant GNNs, we first need to note that the nodes/vertices exist within a vector space $\mathbb{R}^n$. Every node i has an associated position $\mathbf{x}_i$.

Early research in GNNs applied them directly to molecular structures. The work of Duvenaud et al. [105] introduces the basic structure of modern GNNs models, a standard feed-forward convolutional network in which nodes are updated with weighted sums of neighbour features.

One of the limiting factors in molecular analysis operating purely on graph topology, is that the physical position of atoms greatly influences how the molecule interacts. This can be seen in various forms of stereoisomerism. Stereoisomers preserve the molecular formula and bonds of a molecule, but differ in the 3D positions of the constituent atoms. However, embedding 3D positions as graph features brings with it the issue of learning to generalise over rigid rotations and translations of a molecule. As distance is preserved under rigid transformation, one common approach is to augment the problem with a distance matrix.

Shi et al. [106]. proposes an energy-based model for learning molecular conformation. In this paper, equivariance is preserved by augmenting a Graph Isomorphism Network (GIN) [107] - a form of message passing neural network, with edge length information. However, several forms of isomerism can preserve bond length, e.g. E-Z and conformational isomers. The paper acknowledges limitations with relying on covalent bonds as the only edges in the network and augments the graph representation with "virtual" 2-hop and 3-hop edges. Alternatively, the paper E(n) equivariant

Graph Neural Networks (E(n)-GNNs) [108] overcomes the same limitation in a different way. The paper introduces an Equivariant Graph Convolutional Layer (EGCL), a message passing network where the message is dependent on the distances between a node and its neighbours. The model also updates the 3D positions of nodes based on the passed messages after each layer. This update is equivariant under rigid transformation and also serves to distinguish steroisomerisms that preserve bond length.

An Equivariant Graph Convolutional Layer (EGCL) takes in $(\mathbf{h}^l, \mathbf{x}^l)$ and generates $(\mathbf{h}^{l+1}, \mathbf{x}^{l+1})$ - i.e. it generates a **new set of coordinates** on every layer. These new coordinates don't change graph connectivity, but they are used to infer $\mathbb{R}^n$ features.

An EGCL takes the following from:

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \ ||\mathbf{x}_i^l - \mathbf{x}_j^l||^2 \ , a_{ij}\right) \tag{2.12}$$

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \tag{2.13}$$

$$\mathbf{h}^{l+1} = \phi_h\left(\mathbf{h}^l, \mathbf{m}_i\right). \tag{2.14}$$

Also generate x^{l+1} :

$$\mathbf{x}_i^{l+1} = \mathbf{x}^l + C\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right) \tag{2.15}$$

Effectively speaking, $\mathbf{m}_{ij}$ becomes dependent on the distance between nodes i, j, and $\mathbf{x}^{l+1}$ is updated with a weighted sum of the vectors to neighbouring nodes. C is chosen as $1/(M-1)$ to normalise this sum.

GeoDiff [109] applies a denoising diffusion model to the problem of molecular conformation using a form of EGCL, however this requires manipulation of the diffusion chain to maintain equivariance.

Equidock [75] extends E(n)-GNNs with Graph Matching Networkss (GMNs) [110], using both self-attention and cross-attention between two protein graphs in order to determine a potential alignment between the two for protein-protein docking. Given

two graphs G_1 and G_2, we have corresponding vertex sets V_1 and V_2. Consider vertex $v_i \in V_1$ and $v_j \in V_2$. Each of these vertices will have a corresponding hidden vector $\mathbf{h}_i, \mathbf{h}_j$ created through some sort of graph neural network.

The attention between v_i and v_j is not symmetric. The new hidden vector $\mathbf{h}_i^{t+1}$ is modified by an attention mechanism that sums over the hidden vectors for vertices in V_2. The attention weighting $a_{j\to i}$ is calculated by:

$$a_{j\to i} = \frac{\exp\left(s_h\left(\mathbf{h}_i, \mathbf{h}_j\right)\right)}{\sum_{j'} \exp\left(s_h\left(\mathbf{h}_i, \mathbf{h}_{j'}\right)\right)}$$

where s_h is a similarity function and the denominator is summed over all j' in V_2.

Both of these approaches are limited in that only scalar information can be passed though the network, which makes it difficult to predict vector valued information such as the velocity or acceleration vectors of atoms in a molecule. Polarisable atom interaction Neural Network (PaiNN) [111] overcomes this by calculating both scalar ($\mathbb{R}^F$) and vector ($\mathbb{R}^{F\times 3}$) features, passing updates through the network while keeping the atom positions constant (in contrast to Satorras et al. [108]). TorchMD-NET [112] extends PaiNN through the use of an attention mechanism to evaluate molecular potentials.

A further extension of the type of data passed through GNNs was proposed in Tensor Field Networks [113]. Thomas et al. define operations on point-clouds, where every point has a set of features representing a geometric tensor expressed as terms in a spherical harmonic series. This approach has been adapted to GNNs for molecular and protein prediction in several papers. SE(3)-Transformers [114] uses a modification of a Graph Transformer Network (GTN) [115], where the tensor product between an edge direction's spherical harmonic representation and a neighbour's are used to derive query and value vectors. Frank et al. [116] introduce So3karates, an attention based model whose basic architecture is derived from PaiNN. However, spherical harmonic information is now present, and a novel attention mechanism allows for message passing between spatially distant atoms by considering atoms with similar spherical harmonic features to be "neighbours".

2.3.2 Equivariance

In machine learning, Universal Approximation Theorems (UATs) have shown that networks of arbitrary width [117] or arbitrary depth [118] with a variety of non-linear activation functions [117, 119, 120] are capable of approximating a broad range of functions. While this result guarantees that the target function *can* be learned, it does not guarantee that the target function *will* be learned. The core limitation in applying UATs to real-world applications is that for any particular problem only finite training data exists. The true value of the target function is only known at a finite set of points and there exist many incorrect functions that fit the training data perfectly. Thus, models can fail to generalise, and give rise to the phenomenon of over-fitting. There are two main approaches to overcoming this limitation if certain symmetries of the target function are known, augmenting the data to force the network to learn them [121] through random transformations, or limiting the possible functions the network can learn by embedding the symmetries in the network. These symmetries result in invariance or equivariance of the target function.

For example, consider classifying an image of a digit into its numerical value. It is safe to assume that the position of the glyph in the image does not change the class that the glyph is from (i.e. a two is still a two even if it has been shifted 5 pixels across). The target function in this case has a property known as translation *invariance*. Given data I, a translation g from the group of translations G, we have that the classification function f behaves identically if the image has been translated or not (Figure 2.7a)

$$f(g(I)) = f(I).$$

Similarly, if we consider an image segmentation or object detection problem, it is clear that our target function is not invariant to translation. We expect the shift in our input to correspond to an equivalent shift in our output. This is known as translation *equivariance* (Figure 2.7b).

$$f(g(I)) = g'(f(I)).$$

There is a distinction to be made here. The representation of the translation acting

on the image $g(I)$, does not have to necessarily be the same as the representation of the translation acting upon the output of the function f. The output may not be in the same domain as the image. For example, an object detection function may output a pair of coordinates $(x, y), (x', y')$ defining the bounding box of the object. In this case, a shift of the object within the input image I by a pixels rightward, would result in the output $(x + a, y), (x' + a, y')$. The translation g acting on I (shifting the pixels of an image) and the translation g' acting on $f(I)$ (adding a to the x coordinates), correspond to the same translation, but are not the same function as they operate on different domains.

(a) Invariant function f (b) Equivariant Function f

Figure 2.7: Invariance and Equivariance: invariant functions (a) map a transformed representation to the same value, equivariant functions (b) map a transformed representation to a value transformed equivalently in the new domain.

Knowledge of invariance or equivariance of a target function can be exploited by data augmentation — applying the known transformations to the data during training. While this strategy is easily implemented, it requires training a network with many examples of transformed data in order to successfully learn function symmetries. Alternatively, symmetries can be enforced by building these properties into the network layers. The most well known cases of invariant and equivariant layers are pooling and CNN layers respectively. Translation of image data (within the pooling window) does not affect the output of a pooling layer, and translation of image data changes the output of a CNN layer in the same way (in this case, $g = g'$ if applied separately to each channel). While image classification tasks are a form of invariant target function, the very nature of the task requires inferring information from spatial features. Thus, classification networks are commonly built of multiple equivariant CNN layers and small-window pooling operations to produce information about localised image features, followed a global pooling operation to

make the classification invariant to image translation. In contrast, segmentation algorithms such as U-Net introduced by Ronneberger et al. [122] use invariant pooling operations as a way of introducing long-range contextual information to CNN layers. A key advantage of building symmetries into the network's layers is a large reduction in the number of parameters needed to train, and a large reduction in the space of learnable functions, reducing the risk of over-fitting and increasing generalisability. It is important to note that translation is not the only form of symmetry, and that rotational and mirror symmetries also exist.

One of the key problems in computer vision and object recognition is the variability of objects due to pose, position and rotation. Early research into equivariant models focused on building image recognition models that were unaffected by these factors, as they formed natural extensions to the translation equivariance of CNNs. Deep Symmetry Networks [123] were one of the earliest examples of an attempt to apply equivariant techniques to the space of 2D affine transformations by embedding them in a 6D space. Instead of using a feed-forward network on a fixed grid, a *symnet* layer consists of several feature maps that each define a continuous Gaussian kernel that convolves a set of points over a set of different points in the same symmetry space. Classification of an image is achieved by applying gradient ascent on the feature map for each class to find the orientation with the maximum value, then comparing between them to determine which orientation and class is most likely. Henriques et al. [124] attempt to generalise convolutions to a large space of natural images by applying spatial transformations (such as log-polar) to the input image such that the translational equivariance of standard CNNs can be applied to the resultant image. Equivariance in the full breadth of spatial transformations is beyond the scope of most papers in the field, and research is typically focused on 2D and 3D rigid transformations ($SO(2)$ and $SO(3)$ respectively). TI-Pooling [125] approaches 2D rotation invariance by effectively making data augmentation part of the network. Each input image undergoes 24 uniformly spaced transformations that are then fed through the same convolutional network separately before non-linear invariant aggregation via max() function is used. A simple feed-forward network then serves to classify the image in an equivariant fashion.

A competing approach is to embed equivariant features into the network itself, with several similar but distinct approaches. Group Equivariant Convolutional Networks (GECNs) [126] focuses on making 2D convolutions equivariant to 90 degree rotations. This is achieved by augmenting each convolutional filter with rotated versions of itself and permuting the input features as to maintain symmetry over all translations and rotations by 90 degrees ($p4$ symmetry). Similarly, Dieleman et al. [127] attempts to exploit the same $p4$ symmetry. However, instead of augmenting convolutional filters, they augment the images themselves by feeding rotated versions of them through each layer in the network before applying pooling operations. While this has similarities to the TI-Pooling approach, it's important to note that features for each rotation are recombined between every convolutional layer rather after. Cohen et al. introduce Steerable CNNs [128], this extends the augmented filter bank by noting that there exists an orthogonal basis of 3x3 convolutional filters such that each element of the basis has a known behaviour under roto-reflection. By decomposing the filters in such a way, equivariant filters can be learned as linear combinations (with some constraints) of the responses an image generates to the basis filters. Oriented Response Networks (ORNs) [129] can be seen as a different exploration of the augmented filter-bank idea found in GECNs. However, instead of operating on the discrete rotational symmetry group $p4$, an ORN is able to operate with any number of orientations (typically 8). Each point in a feature map effectively becomes a representation of a scalar value in each of the 8 directions. By examining the intensity of each feature, and the aggregate average direction, class and orientation of an object in an image can be predicted. Harmonic Networks [130] address shortcomings in the previous two extensions to the filter bank network. While Steerable CNNs untangle filters in order to produce a basis from which equivariant filters can be learned, they lack full rotational invariance and are limited to 90-degree rotational and reflection symmetries. ORNs are capable of resolving more precise rotational information, however full rotational invariance is not possible, and can only be approximated by increasing the number of orientations and thus computational cost. Harmonic Networks take a different approach and consider a basis of convolutions defined by complex circular harmonics. A filter is

parametrised in terms of radius, phase offset and order. Each filter is made discrete on a grid so that the convolution operation can be applied using standard CNN library functions. Using circular harmonic features allows for arbitrary precision in feature angle, and results in a smooth representation of the feature intensity in each direction. Using higher order harmonics increases the effective angular resolution of the feature intensity, potentially needed in intermediate layers, but is not required for final prediction of class or orientation.

2.3.3 Spherical Harmonics

Spherical Harmonics are used to great extent throughout the 3D equivariance literature, a working knowledge of their properties and the basic methods of manipulating them is essential for an understanding of how many models operate. This section serves as a brief overview of the relevant areas in order to facilitate that understanding.

Spherical Harmonics are a set of orthogonal functions defined on the surface of the 3D sphere. Spherical harmonics can be derived as solutions to Laplace's equation in spherical coordinates:

$$\nabla^2 f = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial f}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 f}{\partial\varphi^2} = 0,$$

where $f : \mathbb{R}^3 \to \mathbb{C}$, i.e. a complex scalar field over 3D space. f is defined such that it can be decomposed into a radial and angular component $f(r, \theta, \varphi) = R(r)Y(\theta, \varphi)$. In spherical coordinates, harmonics are commonly written in the form $Y_l^m(\theta, \varphi)$, a function describing the mth harmonic ($m \in [-l, l]$) of order l. Commonly used for solving Partial Differential Equations (PDEs), spherical harmonics form an orthonormal basis for scalar functions defined on the sphere. This basis relies on a inner product $\langle \cdot, \cdot \rangle$ defined for functions on the sphere as:

$$\langle f, g \rangle = \int_S f(\theta, \varphi)\, g(\theta, \varphi) dS,$$

where $dS = \sin\theta\, d\theta\, d\varphi$. In particular, the spherical harmonics are orthonormal,

such that:

$$\langle Y_{l_1}^{m_1}, Y_{l_2}^{m_2} \rangle = \delta_{l_1 l_2} \delta_{m_1 m_2},$$

where $\delta_i j$ is the Kroneker delta. Analogously to Fourier series, well behaved functions can be decomposed into a set of spherical harmonic coefficients, a function $f(\theta, \varphi)$ can be written as an (in)finite sum:

$$f(\theta, \phi) = \sum_{l_1=0}^{L_1} \sum_{m_1=-l_1}^{l_1} f_{l_1 m_1} Y_{m_1}^{l_1}(\theta, \phi),$$

where the coefficients can be derived from the inner product:

$$f_{lm} = \langle f, Y_l^m \rangle = \int_S f(\theta, \varphi)\, Y_l^m(\theta, \varphi) dS.$$

The coefficients f_{lm} can then be used as a compact approximation of the scalar field f. Importantly, the behaviour of these coefficients when applying operations to the scalar fields such as rotation or multiplication is key to using them in neural networks. Equations for real spherical harmonics (given Cartesian coordinates x, y, z) can be derived from the Herglotz generating function [131]:

$$Y_l^m(x, y, z) = \sqrt{\frac{2l+1}{2\pi}} \Pi_l^{|m|}(z) \begin{cases} \sum_{p=0}^{|m|} \binom{|m|}{p} x^p y^{|m|-p} \sin\left((|m|-p)\frac{\pi}{2}\right) & m < 0 \\ \frac{1}{\sqrt{2}} & m = 0 \\ \sum_{p=0}^{m} \binom{m}{p} x^p y^{m-p} \cos\left((m-p)\frac{\pi}{2}\right) & m > 0 \end{cases} \tag{2.16}$$

Where:

$$\Pi_l^m(z) = \sqrt{\frac{(l-m)!}{(l+m)!}} \sum_{k=0}^{\lfloor (l-m)/2 \rfloor} (-1)^k 2^{-l} \binom{l}{k} \binom{2l-2k}{l} \frac{(l-2k)!}{(l-2k-m)!} r^{2k-l} z^{l-2k-m} \tag{2.17}$$

Tensor Product of Spherical Harmonics

If we have two functions, defined on the sphere, $f(\theta, \phi)$ and $g(\theta, \phi)$, we can decompose them and their product $h(\theta, \phi) = f(\theta, \phi) g(\theta, \phi)$ - the tensor product, into

spherical harmonics.

$$
\begin{aligned}
f(\theta,\phi) &= \sum_{l=0}^{L}\sum_{m=-l}^{l} f_{lm}Y_m^l(\theta,\phi) \\
g(\theta,\phi) &= \sum_{l=0}^{L}\sum_{m=-l}^{l} g_{lm}Y_m^l(\theta,\phi) \\
h(\theta,\phi) &= \sum_{l=0}^{L}\sum_{m=-l}^{l} h_{lm}Y_m^l(\theta,\phi)
\end{aligned}
\tag{2.18}
$$

This is analogous to the multiplication of two time domain signals, we can view spherical harmonics as the decomposition of a signal over a sphere into orthogonal components, the same way we decompose a time-domain signal using a Fourier transform.

Basic Fourier theory shows that multiplication in the time domain of two signals is the same as convolution in the frequency domain (and vice versa). Given two functions $f, g : \mathbb{R} \to \mathbb{R}$ defined as $\lambda_f \cos(\omega_f t)$ and $\lambda_g \cos(\omega_g t)$ respectively, the product in the time domain will result in a function defined in terms of two frequencies: $f(t)g(t) = \frac{\lambda_f\lambda_g}{2}\cos((\omega_f - \omega_g)t) + \frac{\lambda_f\lambda_g}{2}\cos((\omega_f + \omega_g)t)$. Note here how the coefficients of the two frequencies have interacted, with the intensity of the resulting frequencies being a linear combination of the product of the intensities of $f(t)$ and $g(t)$, i.e. $\frac{1}{2} \times \lambda_f\lambda g$. This mapping coefficient, $\frac{1}{2}$, henceforth referred to as P is constant. For time domain signals this linear mapping is easy to realise, and signals decomposed into their constituent frequencies can be multiplied together. For the purposes of analogy with spherical harmonic definitions, we can define this mapping coefficient in terms of input frequencies and output frequencies, resulting in the following general equation for the product of two cosine functions:

$$
\begin{aligned}
\cos(\omega_1 t)\cos(\omega_2 t) &= \sum_{\omega_3} P^{\omega_1,\omega_2,\omega_3}\cos(\omega_3 t) \\
P^{\omega_1,\omega_2,\omega_3} &= \begin{cases} \frac{1}{2} & \text{if } \omega_1 + \omega_2 = \omega_3, \\ \frac{1}{2} & \text{if } |\omega_1 - \omega_2| = \omega_3, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{2.19}
$$

That is, the coefficient of the $\cos(\omega_3 t)$ term is only non-zero at frequencies $\omega_1 + \omega_2$ and $|\omega_1 - \omega_2|$, matching the original trigonometric identity. We note here that cos is an even function, so the sign of $\omega_1 - \omega_2$ does not matter. This toy example is useful in demonstrating the behaviour of how angular frequencies interact.

In spherical harmonic analysis, a signal is defined on a sphere in terms of its component harmonics/frequencies. These component frequencies go up to order L and consist of $(2l + 1)$ harmonics per order. As with time series analysis, the multiplication of two harmonics results in components spread across multiple different harmonics. The tensor product describes how the coefficients h are related to f and g. The basic definition consists of nested summations over both l and m terms. Given known spherical harmonics $Y^{l_1}_{m_1}$ and $Y^{l_2}_{m_2}$, the tensor product can be written as follows [132]:

$$Y^{l_1}_{m_1}(\theta,\phi)Y^{l_2}_{m_2}(\theta,\phi) = \sum_{l=|l_1-l_2|}^{l_1+l_2} \sum_{m=-l}^{l} Q^{l_1,l_2,l}_{m_1,m_2,m} Y^{l}_{m}(\theta,\phi) \tag{2.20}$$

$$Q^{l_1,l_2,l}_{m_1,m_2,m} = \sqrt{\frac{(2l_1+1)\,(2l_2+1)}{4\pi\,(2l+1)}} C^{l_1,l_2,l}_{0,0,0} C^{l_1,l_2,l}_{m_1,m_2,m}. \tag{2.21}$$

The term $Q^{l_1,l_2,l}_{m_1,m_2,m}$ here is analogous to the $P^{\omega_1,\omega_2,\omega_3}$ term from Equation 2.19. Instead of the piecewise function defined before, we are left with a complex equation involving the term $C^{l_1,l_2,l}_{m_1,m_2,m}$. These are the Clebsch-Gordan coefficients. The Clebsch–Gordan coefficients arise from angular momentum coupling in quantum mechanics [133], however a full understanding of their derivation is unnecessary for their application in equivariant networks.

Now that we have a relationship between harmonics, we can consider how the coefficients of our terms from Equation 2.18 are related. Thankfully, this can easily be decomposed into a product-sum over the coefficients and Q terms.

$$h_{lm} = \sum_{l_1 m_1} \sum_{l_2 m_2} Q^{l_1,l_2,l}_{m_1,m_2,m} f_{l_1 m_1} g_{l_2 m_2}$$

Parts of $\mathbf{Q}$ are dependent only on l, these parts are invariant to rotation of the

underlying functions f and g. Many equivariant networks [113, 114, 134] replace this part with learnable parameters k_{l_1,l_2,l_2} such that:

$$Q^{l_1,l_2,l}_{m_1,m_2,m} = k_{l_1 l_2 l} C^{l_1,l_2,l}_{m_1,m_2,m}.$$

This is particularly useful as a set of spherical harmonic coefficients g^l_m can be learned as a "filter" to pick up on features defined on the sphere, analogous to the weights of a CNN layer.

Wigner *D*-matrix

Applying equivariance principles to spherical harmonics relies on being able to reason about how spherical harmonics and their products behave under rotation. The behaviour of a set of spherical harmonics of order l under rotation is defined in terms of a matrix $D^{(l)}_{mm'}$ known as a Wigner D-matrix of order l. The Wigner D-matrix describes the rotation of a set of spherical harmonics as a linear combination of their coefficients [135], and is both orthonormal and unique for a given rotation. Thomas et al. [113] show that for any given pair of spherical harmonic coefficients, transforming both by the same Wigner D-matrix and taking the tensor product is equivalent to applying the same Wigner D-matrix to the result of non-transformed coefficients. It is this property that is key to building 3D rotationally equivariant networks.

2.3.4 Equivariance in 3D

Data science in 2D is dominated by image analysis, and the use of 2D arrays for representing data. In 3D, data suffers from the curse of dimensionality and the memory requirements of volumetric information increase dramatically. Hence, research into equivariance in 3D spans a wider range of data representations and overlaps significantly with GNN research on point-clouds. Due to network complexity and the higher dimensionality of 3D space (3 translation and 3 rotation dimensions) vs 2D space (2 translation, 1 rotation dimension), group symmetry principles inform a large amount of contemporary network architectures for protein and molecular

problems. Layers are typically $E(3)$ equivariant, and $SE(3)$ invariant.

On volumetric data, 3D G-CNNs [136] extends GECNs to 3D by augmenting convolutional filters with rotated and reflected versions. However, due to the nature of rotations in 3D, even when constrained to 90 degrees, this filter bank becomes unworkably large, with 24 augmentations needed per filter to capture orientation preserving rotations of the cube, taking up large quantities of the parameter budget for fewer features. CubeNet [137] functions similarly, but pays specific attention to ensuring that the composition of rotated filters are mapped to the correct channel of a particular feature, while also exploring subgroups of the 24 orientations to reduce filter bank complexity. Weiler et al. [138] extend Harmonic Networks [130] into 3 dimensions. Instead of using circular harmonics to derive equivariant features in SO(2), spherical harmonics are used to derive equivariant features in SO(3). Spherical harmonics are used to define an equivariant basis, as each harmonic can be represented by a continuous function. Rotational transformations of features can be expressed as transformations of the spherical harmonic decomposition by applying a Wigner D-matrix. Each feature is represented by a 2nd order tensor, a 3x3 matrix, which is decomposed into 0th (scalar), 1st (vector) and 2nd order terms as these three terms behave differently under rotation. 0th, 1st and 2nd order basis kernels are derived from spherical harmonics weighted with a radial Gaussian function in order to limit their size. These kernels are discretised into standard 3D convolutions, and learned linear combinations of them used as with Harmonic Networks to convolve equivariantly. It is important to note that with point-cloud data, the sample positions are irregular, therefore the discretisation step of Weiler et al. [138] cannot be applied.

Kondor [139] introduces N-body Networks, in which the point-cloud is recursively partitioned into overlapping subtrees. Each layer of the subtree, starting with the layer corresponding to individual points applies two forms of operations, a pairwise operation between each pair of points in a given leaf, and an aggregation operation to pass information up the tree to the next layer. These operations are all equivariant to rotation or translation, such that the final, global, aggregated values can be used to predict scalar and vector properties. The seminal paper in equivariance

on point-clouds is Thomas et al.'s 2018 work Tensor Field Networks [113]. Tensor Field Networks (TFNs) function as GCNs over point-clouds, with graph connectivity derived from k-nearest-neighbour relationships. As with other 3D equivariant approaches, each point has an associated set of spherical harmonic features. Standard neural networks represent scalar features through a series of channels, however sometimes we wish to predict properties like translation, view direction etc. In these cases we can use a more complex set of features. Tensor field networks treat every point in space as having an associated tensor. This tensor is described by F spherical harmonics representing separate features. Given n points in $\mathbb{R}^3$ and F channels, a TFN consists of an array of $n \times 3$ real valued coordinates describing each point's position, and an array of $n \times F \times l$ real valued feature values, where l is the highest spherical harmonic order. Tensor Field Networks builds on results from Harmonic Networks [130] - 2D rotational equivariance with circular harmonics; and SchNet [140] a 3D invariant network that uses pairwise distances between atoms.

The convolutions differ from previous approaches by defining rotationally equivariant filters that depend on the relative positions between two points. Given two points a, b with locations $\vec{r_a}, \vec{r_b}$ a filter is dependent on $\vec{r_{ab}} := \vec{r_a} - \vec{r_b}$. A filter is defined as a set of spherical harmonics, one for each order of the input. Filters consist of angular and distance components, such that a filter $F_{cm}^{l_f,l_i}$ is of the form:

$$F_{cm}^{(l_f,l_i)}(\vec{r}) = R_c^{(l_f,l_i)}(r) Y_m^{(l_f)}(\hat{\vec{r}}) \tag{2.22}$$

where $r = |\vec{r}|$, $\hat{\vec{r}} = \vec{r}/r$. $Y_m^{(l_f)}(\hat{\vec{r}})$ is the coefficients of order l_f given by decomposing a unit impulse in the direction of r into its spherical harmonics, $Y_m^{(0)}(\hat{\vec{r}}) = c$, the scalar term is equal to a constant, and $Y_m^{(1)}(\hat{\vec{r}}) \propto \hat{\vec{r}}$, the vector term is proportional to the direction of the vector. $R_c^{(l_f,l_i)}(r)$ are a set of learned functions from $\mathbb{R}_{\geq 0} \to \mathbb{R}$ such that a filter responds with different strengths to different orders of harmonics. For each channel/feature c, and each pair of harmonic orders(l_f, l_i) (filter and input respectively), the (learned) function $R_c^{(l_f,l_i)}(r)$ takes in a scalar distance/vector length and returns a scalar $\mathbb{R}_{\geq 0} \to \mathbb{R}$. We then multiply the spherical harmonics of order l_f generated by the normalised vector $\hat{r}$ by this scalar term to arrive at a spherical harmonic representation of $\vec{r}$ which is equivariant to rotation. This is

because the $Y_m^{(l_f)}(\hat{r})$ term rotates with the associated Wigner D-matrix, and the $R_c^{(l_f,l_i)}(r)$ term is invariant to rotation. Typically the radial function $R_c^{(l_f,l_i)}(r)$ uses a weighted Gaussian to limit the impact of long-range dependencies and allow for optimisations by only considering the point interactions within a certain range.

A TFN layer consists of aggregating the tensor product of each filter with a points neighbours. Given vectors $\vec{r}_a$ with differences $\vec{r}_{ab} = \vec{r}_a - \vec{r}_b$, a convolutional layer is defined by taking tensor products of the spherical harmonic features of nearby points $b \in S$ with filter features generated by the vector towards each point $\vec{r}_{ab}$. This allows for the spherical harmonics to interact in a way where both the direction towards the nearby point and the spherical harmonics of that point contribute to the update.

$$\mathcal{L}_{acm_0}^{(l_0)}\left(\vec{r}_a, V_{acm_0}^{(l_i)}\right) = \sum_{m_f,m_i} C_{(l_f,m_f)(l_i,m_i)}^{(l_o,m_o)} \sum_{b\in S} F_{cm_f}^{(l_f,l_i)}(\vec{r}_{ab}) V_{bcm_i}^{(l_i)} \tag{2.23}$$

Where the subscripts i, f and o denote representations of the input, filter and output respectively, $C_{(l_f,m_f)(l_i,m_i)}^{(l_o,m_o)}$ denotes Clebsch-Gordan coefficients. This spherical-harmonic filter, message-passing based paradigm for operating on point-clouds is re-used extensively throughout later literature.

Cormorant [134], uses a similar Clebsch-Gordan approach, but also introduces non-linearities on the spherical-harmonic vertex representations by taking the tensor product of each vertex with itself. Unfortunately, as noted in the paper, this quadratic, unbounded, non-linearity is the likely cause of large training instabilities, and poor weight initialisation or poor choice of optimisation algorithm frequently results in exploding losses or convergence to a poor fit.

Fuchs et al. [114] introduce $SE(3)$-Transformers, in which the TFN equation (2.23) is modified to use an attention mechanism rather than an unweighted summation over neighbouring nodes. This allows for greater specificity in message passing updates.

Townshend et al. [141] notes that TFN networks can also be used to predict geometric values such as force field vectors. This is applied to molecular structure refinement in order to optimise molecular predictions. Batzner et al. [142] address a similar problem to Townshend et al. but note that directly predicted force fields are

not guaranteed to be conservative. A conservative vector field is the gradient of some potentially unknown scalar function. Without this property, a force field would be unsuitable for use in simulations. Instead, Batzner et al. uses a global pooling operation is used in order to predict global molecular system energy. Backpropagating through the network can be used to recover gradients on atomic positions, resulting in a gradient field that is conservative, thus the model can be used in molecular dynamics simulations.

2.3.5 Learning on Manifolds

One of the core hypotheses of machine learning is that data in a high-dimensional space exists either on, or near to a low-dimensional manifold embedded in that space [143]. This assumption is used in a wide variety of unsupervised learning approaches [144]. While significant effort has been put into this area of research, it is important to note that there are many datasets for which the manifold is known. Data distributions defined on space such as the sphere, and the space of 2D and 3D rotations are commonly found in a broad range of machine learning applications from pose prediction in augmented reality and special effects [145] to protein docking and structural biology problems. However, as neural networks apply linear and non-linear transformations to data defined on $\mathbb{R}^n$, expressing manifold inputs and outputs in ways that can be smoothly embedded in $\mathbb{R}^n$ is an active area of research, and improper embedding can lead tp suboptimal performance. Zhou et al. [146] show how this is the case for several different embeddings of 3D rotations. Neural network models are typically trained through Maximum Likelihood Estimation (MLE), that is, network losses are a form of Negative Log-Likelihood (NLL) given an assumption of the underlying distribution of the data. For example, a regression problem is typically optimised by minimising the Mean Squared Error (MSE) between prediction and target. In this case, MSE is proportional to the NLL of target data when predicting the mean of a Gaussian distribution of unknown (but fixed) variance [147]. Classification losses, such as (binary) cross entropy loss, calculate the NLL for the parameters of Bernoulli and categorical distributions respectively. However, these distributions cannot be used for the optimisation of distributions defined on man-

ifolds. Of greatest importance to protein docking are distributions defined on the 3D sphere and on the space of 3D rotations - $SO(3)$. Instead models can be trained by predicting parameters of a known distribution on these manifolds (in particular, predicting the mean) and minimising the NLL of sampling the true value from the predicted distribution.

Probability Distributions on the Sphere

The von Mises-Fisher distribution is a continuous distribution over the unit sphere $\mathcal{S}^{p-1} \subset \mathbb{R}^p$ [148], where the probability of sampling the unit length vector $\mathbf{x}$ is defined as

$$p(\mathbf{x}|\mu, \kappa) = C_p(\kappa) \exp\left(\kappa \mu^T \mathbf{x}\right), \tag{2.24}$$

where μ a unit-vector mean direction parameter, and κ a scalar concentration parameter. The normalising constant $C_p(\kappa)$ is equal to $\frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)}$ [149], where I_v denotes the modified Bessel function of the first kind of order v. If we assume a fixed or unknown κ, then optimising the NLL can be done by predicting μ and minimising the function $-\mu^\top \mathbf{x}$. This reflects typical assumptions when doing regression on $\mathbb{R}^n$ data, i.e. the use of MSE as a NLL loss, assuming data is normally distributed. This is a common pattern in many distributions over the sphere, defining an expression over the parameters that computes a real number, exponentiating that value so that it is always positive, then normalising that value by a parameter-dependent constant so that the distribution integrates to 1. Similarly, the Bingham distribution is an antipodally symmetric probability distribution on the n-sphere [150]. The Bingham distribution can be viewed as a distribution over lines that pass through the origin in $\mathbb{R}^p$. The Bingham distribution, defined as:

$$p(\mathbf{x}) = \frac{1}{{}_1F_1\left(\frac{1}{2}, \frac{n}{2}, Z\right)} \exp\left(\mathbf{x}^\top M Z M^\top \mathbf{x}\right), \tag{2.25}$$

is parameterised in terms of an orthonormal matrix M (a matrix in which columns are orthogonal vectors, each of length one) that controls orientation, and a diagonal shape and concentration matrix Z whose elements control how spread out the dis-

tribution is. This distribution is commonly used to represent uncertainty in axial data, e.g. preferred crystal orientations in minerals [151], geological magnetic analysis [152], and scene reconstruction [153]. The normalising constant ${}_1F_1\left(\frac{1}{2}, \frac{n}{2}, Z\right)$ is a confluent hypergeometric function of matrix argument [154], equal to the integral over the sphere of $\int_{SO(3)} \exp\left(\mathbf{x}^\top Z\mathbf{x}\right) d\mathbf{x}$, and has the form of an infinite sum [154]. Calculation of this term is therefore computationally expensive.

Probability Distributions on $SO(3)$

The manifold $SO(3)$ is typically represented as orthogonal 3×3 matrices of determinant 1 as this can be used to easily rotate vectors through matrix multiplication. However, unit length quaternions are also commonly used in computer graphics to represent 3D rotations. A rotation of angle θ around the unit-length axis $\mathbf{u}$ can be represented by the quaternion:

$$\cos\frac{\theta}{2} + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2}, \tag{2.26}$$

where $(1, \mathbf{i}, \mathbf{j}, \mathbf{k})$ form the basis vectors a quaternion system. Notably, the coefficients of these basis vectors $(\cos\frac{\theta}{2}, u_x\sin\frac{\theta}{2}, u_y\sin\frac{\theta}{2}, u_z\sin\frac{\theta}{2})$ lie on the surface of $\mathcal{S}^3$ – the unit sphere in $\mathbb{R}^4$. Thus any distribution defined on $\mathcal{S}^3$ may be used to represent a distribution over $SO(3)$.

Since the unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same 3D rotation, the Bingham distribution's antipodal symmetry correctly captures the space of quaternion rotations. Gilitschenski et al. [155] fully parametrise a Bingham distribution with the output from a neural network, and are able to quantify uncertainty over 3D rotations in their predictions of 3D object orientation. The heavy computational burden of calculating the normalising constant ${}_1F_1\left(\frac{1}{2}, \frac{n}{2}, Z\right)$ is alleviated by pre-computing a range of values (this is feasible as the diagonal matrix Z only has three degrees of freedom) and interpolating between them. However, the 4×4 rotation matrix M is not unique, leading to discontinuity issues introduced by predicting alternate (but still correct) M values for two nearby orientations [146]. This issue applies to any rotational model using quaternions, thus distributions defined directly on $SO(3)$ should be considered instead.

The Matrix-Fisher distribution $\mathcal{MF}(F)$ on SO(3) parametrises the space of rotations with a Probability Density Function (PDF) of the form

$$p(\mathbf{x}) = \frac{1}{c(F)} \exp\left(\text{tr}\left(F^\top \mathbf{x}\right)\right),$$

where $F \in \mathbb{R}^{3\times3}$ parameterises this family of distributions, and $\mathbf{x}$ is a rotation matrix. The normalising factor $c(F)$ is calculated such that $p(\mathbf{x})$ integrates to 1. Notably, the matrix F has no constraints on its values. Outputs from a neural network are free to take any values they wish, and do not need to be modified to lie on $SO(3)$. Singular Value Decomposition (SVD) of the matrix parameter F, i.e.:

$$F = U\Sigma V^\top \tag{2.27}$$

where U and V are orthonormal matrices and Σ is diagonal, is used to extract interpretable information about the distribution [156]. Following a normalisation procedure to ensure $U, V \in SO(3)$, the mean of the distribution can be calculated from $UV^\top$.

The normalising constant $c(F)$ would typically require evaluation of the integral $\int_{SO(3)} \exp\left(\text{tr}\left(F^\top Q\right)\right) dQ$. However, calculation of the normalising constant $c(F)$ can be simplified to a line integral as it is invariant to rotation of the distribution, and is thus only dependent on the values in the singular matrix S. The normalising factor, $c(F) = c(S)$, with singular values s_1, s_2, s_3 can be defined for any ordering of i, j, k to $1, 2, 3$ as:

$$\frac{1}{2}\int_{-1}^{1} I_0\left[\frac{1}{2}(s_i - s_j)(1-u)\right] \times I_0\left[\frac{1}{2}(s_i + s_j)(1+u)\right] e^{s_k u} du, \tag{2.28}$$

where I_0 is the 0th order modified B essel function. However, this is still quite computationally intensive to evaluate. Nevertheless, for the purposes of Energy Based Models (EBMs) (see Section 2.4), the Matrix-Fisher distribution does have one key advantage, the gradient of the log-probability is tractable and easily calculated, and thus can be used for Langevin dynamics with ease. Langevin dynamics uses the gradient of the log-probability of a distribution to produce samples from it. The

Matrix-Fisher's gradient of log-probability can be derived as follows:

$$p(R) = \frac{1}{c(F)} e^{\mathrm{tr}(F^\top R)} \tag{2.29}$$

$$\log(p(R)) = -\log(c(F)) + \mathrm{tr}(F^\top R) \tag{2.30}$$

$$\Delta_R \log(p(R)) = \Delta_R \mathrm{tr}(F^\top R) \tag{2.31}$$

The isotropic Gaussian on SO(3) is defined by a mean rotation g and standard deviation ϵ. Unlike the Matrix-Fisher distribution, the convolution of two isotropic Gaussian distributions is also a isotropic Gaussian distribution. It also has the property of being a fundamental solution to the diffusion equation on SO(3):

$$\frac{\partial \phi(\mathbf{r}, t)}{\partial t} = \nabla^2 \phi(\mathbf{r}, t), \tag{2.32}$$

where $\phi(\mathbf{r}, t)$ is the density of the diffusing system at location $\mathbf{r} \in SO(3)$ at time t. Given two isotropic Gaussian distributions, parametrised as $\mathcal{I}(\mu_1, \epsilon_1^2)$, $\mathcal{I}(\mu_2, \epsilon_2^2)$, the distribution of the rotation taken from composing rotations from these distributions is $\mathcal{I}(\mu_2\mu_1, \epsilon_1^2 + \epsilon_2^2)$. The linearity in the variance makes this distribution especially useful for diffusion models.

An isotropic Gaussian distribution $\mathcal{I}(\mu, \epsilon^2)$ can be decomposed into two separate distributions $\mathcal{I}(\mu, 0), \mathcal{I}(I, \epsilon^2)$. As the first distribution has a variance of 0, it can be viewed as multiplying from the left the identity-mean distribution by a fixed rotation matrix μ. For the identity-mean distribution, the isotropic Gaussian is defined in axis-angle form. We can decompose our rotation into axis-angle form and define an angular PDF $f(\theta)$:

$$f(\theta) = \frac{1 - \cos\theta}{\pi} \sum_{l=0}^{\infty} (2l+1) \exp(-l(l+1)\epsilon^2) \frac{\sin((l + \frac{1}{2})\theta)}{\sin(\theta/2)}, \tag{2.33}$$

where the axis of rotation is sampled uniformly. This equation converges due to the negative exponential term, but small values of ϵ, can result in finite approximations requiring high numbers of evaluations of the inner equation before approximating the true distribution sufficiently.

While the group SO(3) is non-commutative, the isotropic nature of the distribution means that the order of multiplication of these matrices does not matter. This is provable by considering the same rotation constructed from the same distribution in two ways, only differing in order of application of their rotations. Given post-multiplication by the mean (R) and pre-multiplication, we know that $Z_1 R = R Z_2$, where Z_1 and Z_2 are samples from the identity-mean distribution. Pre-multiplying both sides by R^{-1} shows that $R^{-1} Z_1 R = Z_2$. Thus Z_1 and Z_2 are similar matrices and have the same trace. As the trace of a rotation matrix is a function of angle of rotation, then the angle of rotation is equal and the two matrices Z_1 and Z_2 differ only in axis of rotation, which is sampled uniformly. As this is true for any chosen rotation, the two distributions must be equal. The infinite summation converges quickly for $\epsilon > 1$, however for small ϵ this sum can take thousands of terms to converge. This is an issue during diffusion process sampling as the diffusion process requires the application of many small steps. Matthies et al. [157] propose an approximation of high accuracy for small ϵ.

$$f(\theta) = \frac{\left(1 - \cos(\theta)\right)}{\pi} \sqrt{\pi} \epsilon^{-\frac{3}{2}} e^{\frac{\epsilon}{4}} e^{-\frac{\left(\frac{\theta}{2}\right)^2}{\epsilon}} \cdot \frac{\left[\theta - e^{-\frac{\pi^2}{\epsilon}} \left((\theta - 2\pi) e^{\frac{\pi\theta}{\epsilon}} + (\theta + 2\pi) e^{-\frac{\pi\theta}{\epsilon}} \right)\right]}{2 \sin\left(\frac{\theta}{2}\right)}$$

Sampling from the isotropic Gaussian distribution can be achieved through numerical integration of the PDF to generate an approximate Cumulative Distribution Function (CDF), then using inverse transform sampling to turn a sample from the uniform distribution $u \sim \mathcal{U}(0, 1)$ into a sample from the angular component of the isotropic Gaussian distribution. For small ϵ, the approximation defined above may be used. The axis of rotation can easily be chosen through any form of uniform sampling from the sphere.

2.4 Energy Based Models

Energy Based Models (EBMs) [158] are a class of generative models that are based on the Boltzmann distribution. Given a system consisting of states y_i for which each has an associated energy $\epsilon(y_i)$, the Boltzmann distribution describes the probability

that the system can be found in state y_i as a function of the state's energy, $\epsilon(y_i)$ and the system's temperature T:

$$p(y_i) \propto e^{-\epsilon(y_i)/k_B T},$$

where k_B is the Boltzmann constant. As temperature increases, the probability of each state evens out until all states are equally probable. With finite states, the Boltzmann distribution can be normalized by dividing by the sum of the exponentials. The equation can be further reduced with a change of sign, and by simplifying the constants involved, i.e. $\beta = -\frac{1}{k_B T}$:

$$p(x_i) = \frac{e^{-\epsilon(y_i)/k_B T}}{\sum_j e^{-\epsilon(y_j)/k_B T}} = \frac{e^{\beta\epsilon(y_i)}}{\sum_j e^{\beta\epsilon(y_j)}}$$

Note that the final form of this equation is the same as *softmax* normalisation used in classification networks - i.e $\epsilon(x_i)$ is the ith output of a classification network $f_\theta(\mathbf{x})[i]$ prior to softmax normalisation. Without the softmax normalisation, a classification network is effectively computing the energy associated with each possible state for a given system. Training a classification network can be construed as a limited form of EBM, and adaptations to the training mechanism allow for classification networks to be used as generative models [159].

EBMs further extend this idea by considering probability distributions of data on continuous spaces, i.e.

$$p(x) = \frac{e^{-\epsilon(x)/kT}}{\int_{\bar{x}\in\mathcal{X}} e^{-\epsilon(x)/kT}} \tag{2.34}$$

Given a data distribution p_d, and an EBM distribution p_θ parametrised by θ, we seek to minimise the NLL loss: $\mathcal{L}(\theta) = \mathbb{E}_{\vec{x}\sim p_d}\left[-\ln p_\theta\left(\vec{x}\right)\right]$. In general, the integral in the denominator of Equation 2.34 makes this intractable. Thus, training of EBMs must rely upon proxy objectives rather than optimising the equation itself. Contrastive Divergence (CD) is a popular proxy objective, in which the gradient of loss is constructed such that on each optimisation step the energy values of data samples

are decreased and the energy of values sampled from the model are increased:

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\vec{x}^+ \sim p_d}\left[\nabla_\theta E_\theta\left(\vec{x}^+\right)\right] - \mathbb{E}_{\vec{x}^- \sim p_\theta}\left[\nabla_\theta E_\theta\left(\vec{x}^-\right)\right], \tag{2.35}$$

where $\vec{x}^- \sim p_\theta$ is a sample from the EBM, typically found through Markov Chain Monte Carlo (MCMC) sampling. This equation approximates the gradient of the NLL loss [160, 161].

Training larger architectures through CD requires efficient sampling from p_θ. MCMC methods such as random walk and Gibbs sampling [162] suffer from long mixing times when applied to high dimensional data. More recently, stochastic gradient Langevin dynamics [163, 164] have been proposed [165, 166] to sample from p_θ, using the following iterative process:

$$\vec{x}_0 \sim p_0(\vec{x}) \qquad \vec{x}_{i+1} = \vec{x}_i - \frac{\alpha}{2}\frac{\partial E_\theta\left(\vec{x}_i\right)}{\partial \vec{x}_i} + \vec{\epsilon}, \tag{2.36}$$

where p_0 is typically a uniform distribution over the input domain, α is the step size, and $\vec{\epsilon}$ is sampled from an isotropic multivariate normal distribution i.e. $\vec{\epsilon} \sim \mathcal{N}\left(\vec{0}, \alpha I\right)$. As the number of updates $N \to \infty$ and step-size $\alpha \to 0$, the distribution of samples converges to p_θ [164]. While more practical, Langevin MCMC still requires a large number of steps to converge. Several solutions to this problem have been proposed.

Persistent CD [165, 167] relies on maintaining a replay buffer of previously generated samples while randomly resetting some to noise. At each step, as the parameters θ are updated, samples are refined to fit the updated distribution p_θ. This relies on the assumption that the learned distribution p_θ only ever changes gradually, such that Langevin MCMC samples from a previous iteration already form a good approximation to samples from the current distribution, and thus only a short Langevin MCMC chain is needed to update them. As p_θ is typically implemented as a neural network optimised through gradient descent, this assumption holds.

2.4.1 Score Matching and Denoising Diffusion

While Langevin MCMC has allowed for EBMs to scale to high dimensional data, training times are still slow due to needing to sample from the model distribution. Unfortunately this is also exacerbated by the finite-length sampling process. Langevin MCMC only guarantees convergence to the distribution in the limit, at infinite steps, and a finite-length sampling process can still result in samples far away from the model's distribution [168]. An alternative approach is score matching [11]. Score matching relies on minimising the difference between the derivatives of the data (p_d) and model's (p_θ) log-density functions. The score - $s(\mathbf{x}) = \Delta_x \log p(\mathbf{x})$ is independent of any normalising constants needed to define a distribution, i.e. the intractable denominator of a Boltzmann distribution. This allows for EBMs to be built using the learned score function [169] by minimising the Fisher divergence between p_θ and p_d,

$$\mathcal{L} = \frac{1}{2}\mathbb{E}_{p_d(\mathbf{x})}\left[\left\| s_\theta(\mathbf{x}) - s_d(\mathbf{x})\right\|_2^2\right], \tag{2.37}$$

however, the score function of data is not typically known, only samples of data taken from an unknown distribution. While various methods such as spectral approximation [170], sliced score matching [171], and finite difference score matching [172] exist to estimate the score function, the most notable of these is denoising score matching [173]. Denoising score matching allows the score to be approximated using partially-noised data samples. In particular, when this noise is sampled from a normal distribution, i.e. $q = \mathcal{N}\left(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I\right)$, Equation 2.37 simplifies to:

$$\mathcal{L} = \frac{1}{2}\mathbb{E}_{p_d(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}\left(\mathbf{x},\sigma^2 I\right)}\left[\left\| s_\theta(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}\right\|_2^2\right]. \tag{2.38}$$

This allows for s_θ to estimate the noise, thereby allowing it to be used as a generative model [174, 175]. Since the Langevin update step uses $\Delta_{\mathbf{x}} \log p(\mathbf{x})$, it is possible to sample from a score matching model using only Langevin dynamics [176]. This is only possible, however, when trained over a large variety of noise levels so that $\tilde{\mathbf{x}}$ covers the whole space.

Closely related to denoising score matching approaches are diffusion models [8,

12,177,178]. Diffusion models define a forward process aka a diffusion process $q()$ as a discrete Markov chain that gradually adds Gaussian noise to data sampled from the data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, and a second, *learned* reverse process $p_\theta()$ that attempts to *remove* noise from data. The forward diffusion process is conditional on an initial sample $\mathbf{x}_0$. q for $t \in 1:T$ is defined as,

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{2.39}$$

At each timestep t the conditional distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is modelled as being a normal distribution with variance β_t, and mean $\sqrt{1-\beta_t}\mathbf{x}_{t-1}$,

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}). \tag{2.40}$$

The sequence $\beta_1, ..., \beta_T$ is known as the variance schedule and determines how much noise to add at each step. Most importantly, the forward process (a sequence of conditional probabilities) can be reduced into a closed form for the distribution $q(\mathbf{x}_t|\mathbf{x}_0)$. For brevity, two new schedule terms are derived from β_t values,

$$\alpha_t = 1 - \beta_t, \tag{2.41}$$

$$\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s. \tag{2.42}$$

Note that $1-\bar{\alpha}_t$ is equal to the cumulative variance of the diffusion process. The probability distribution at time t is then derived as,

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}).$$

This closed form allows for fast sampling of the forward process. The forward process allows for the gradual diffusion of data into random noise. The reverse model p_θ is defined at time T as a fully diffused random Gaussian noise variable:

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

We define our final result $p_\theta(\mathbf{x}_0)$ as the product of a series of conditional probabilities,

$$p_\theta(\mathbf{x}_0) := p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t).$$

The conditional probability $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is defined as:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

Note that μ_θ and Σ_θ have learned parameters θ, and are typically implemented as a neural network. The network is trained by optimising the negative log likelihood of sampling true data from the reverse process.

$$\mathcal{L} = \mathbf{E}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right], \tag{2.43}$$

$$\mathcal{L} = \mathbf{E}\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right]. \tag{2.44}$$

As we are able to sample $q(\mathbf{x}_t|\mathbf{x}_0)$ at arbitrary timesteps, we can generate random terms form this summation and optimise $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ through Stochastic Gradient Descent (SGD). We can also reduce the variance by re-writing the above equation for $\mathcal{L}$ as the sum of Kullback–Leibler (KL) divergence [179] terms,

$$\mathcal{L}_T := D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0) \;||\; p(\mathbf{x_T})), \tag{2.45}$$

$$\mathcal{L}_{t-1} := D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \;||\; p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)), \tag{2.46}$$

$$\mathcal{L}_0 := -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1). \tag{2.47}$$

$\mathcal{L}_T$ is a comparison between the distribution of $\mathbf{x}_T$ of the forward process q given a sample $\mathbf{x}_0$, and the distribution of the starting point $\mathbf{x}_T$ of the reverse process p. Note that $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. Therefore $\mathcal{L}_T$ is only dependent on the choice of $\beta_1, ...\beta_T$, the variance schedule. Minimising this term constrains the end state of the forward process to be a standard normal distribution. The equation $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is the distribution of the *previous step* in the forward process, given a known sample

from the data distribution $\mathbf{x}_0$ and a known current step $\mathbf{x}_t$. This is compared with the distribution the reverse process predicts from $\mathbf{x}_t$, i.e. $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$; $\mathcal{L}_t$ is a measure of how similar the predicted distribution p of $\mathbf{x}_{t-1}$ is to the possible previous values q of $\mathbf{x}_{t-1}$ The equation for $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is derived from conditional probabilities,

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}). \tag{2.48}$$

The mean $\tilde{\mu}$ is calculated by "blending" between $\mathbf{x}_t$ and $\mathbf{x}_0$.

$$\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_{t-1}}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \tag{2.49}$$

The variance $\tilde{\beta}_t$ is the ideal variance of the reverse process if there existed only one sample from $q((x))$, i.e. $\mathbf{x}_0$ is fixed. It is defined as:

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t.$$

By minimising $\mathcal{L}_{t-1}$ the reverse distribution is better at predicting the previous timestep. This can be minimised by making the parameters θ predict $\mathbf{x}_{t-1}$ better, or by reducing β_t. Note that the values for $\beta_{1:T}$ are constrained by the first term L_T to have a cumulative variance of 1. While many different variance schedules fit this property, and later papers have investigated optimising the variance schedule [180], a fixed, known variance schedule is often used for more stable training. Therefore, we try and minimise the value of the combined equation:

$$E_q\left[L_T + \sum_{t>1} L_{t-1} + L_0\right],$$

where the terms L_T, L_{t-1} and L_0 are the three losses defined in Equations 2.45, 2.46 and 2.47 respectively. Note that the KL divergences are between Gaussian distributions, thus there exist well known closed form expressions to calculate each of the terms. By fixing the forward process β_t values, L_T becomes constant, so can be removed from any optimisation strategy. The choice of β_t values is constrained, but multiple schedules have been proposed throughout the literatures [8, 180]. In-

stead of using a neural network to estimate the variance for the reverse process, the covariance matrix is fixed to $\sigma_t^2\mathbf{I}$. Ho et al. [8] evaluate both β_t and $\tilde{\beta}_t$ as σ_t^2 and observe similar performance. These are the two extreme choices corresponding to upper and lower bounds on reverse process entropy for data with coordinate-wise unit variance. As mentioned before, the KL divergence between two normal distributions has a closed form. Given two distributions in $\mathbb{R}^k$, $q \sim \mathcal{N}(\mu_q, \Sigma_q)$ and $p \sim \mathcal{N}(\mu_p, \Sigma_p)$, the KL divergence $D_{KL}(q||p)$ is given by:

$$D_{KL}(q||p) := \frac{1}{2}\left[\log\frac{\det(\Sigma_p)}{\det(\Sigma_q)} - k + (\mu_q - \mu_p)^T\Sigma_p^{-1}(\mu_q - \mu_p) + \mathrm{tr}(\Sigma_q^{-1}\Sigma_p)\right]$$

While this equation is complex; fixed, isotropic standard deviation terms means that the loss, is only dependent on the terms involving the predicted mean μ_p, the rest of the terms can be discarded as constants C.

$$\mathcal{L}_{t-1} = E_q\left[\frac{1}{2}(\mu_q - \mu_p)^T\Sigma_p^{-1}(\mu_q - \mu_p)\right] + C$$

Note that Σ_p is equal to $\sigma_t^2 I$, due to choosing a fixed covariance matrix. Thus, $\mathcal{L}_{t-1}$ simplifies further to

$$\mathcal{L}_{t-1} = E_q\left[\frac{1}{2\sigma_t^2}||(\mu_q - \mu_p)||^2\right] + C,$$

where μ_q is the "blending" we defined earlier to determine the mean at time $t-1$ from the *forward process*

$$\mu_q = \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)$$

and μ_p is the predicted mean at time $t-1$ from the neural network/reverse process

$$\mu_p = \mu_\theta(\mathbf{x}_t, t).$$

In practice, the loss term does not use the $\frac{1}{2\sigma_t^2}$ scaling and instead models are defined as predictors of the noise ϵ used to generate noised samples. The $\mathcal{L}_0$ term used in the original Denoising Diffusion Probabilistic Model (DDPM) paper takes

advantage of the fact that samples are being generated in the image domain. A small tweak to the direct analysis of a Gaussian present in the original L_0 term is applied. Effectively, this is a "binning" process, looking at each coordinate in the predicted distribution of $\mathbf{x}_0$ and integrating over the range of acceptable values. In the case of image generation, pixel intensities are in the range $0, 1, ..., 255$. For purposes of prediction, and to reduce any biases introduced by the diffusion process, images are scaled to the range $[-1, 1]$. The "acceptable" range for a true value x_0 is $x_0 \pm \frac{1}{255}$. If $x_0 = -1$ or $x_0 = 1$, the the accepted ranges are extended to $x_0 \in (-\infty, -\frac{254}{255})$ or $x_0 \in (-\frac{254}{255}, \infty)$ respectively. This modification to the $\mathcal{L}_0$ term is necessary as a Gaussian distribution has a infinite support over all real numbers, whereas the target domain of images limits values to $[-1, 1]$.

2.4.2 Energy Based Molecular Models

Several applications of EBMs to molecular and protein modelling have been attempted, however as many molecular models also attempt to implement equivariance in 3D, a disconnection between translation invariance, and the zero-mean approach of DDPMs arises.

ConfGF

Shi et al. [106] propose a model ConfGF that learns gradient fields for molecular conformation generation. Annealed Langevin dynamics sampling is used to sample values from a learned score function. To avoid equivariance issues, the log-density of atomic coordinates $\mathcal{C}$ given by $\Delta_{\mathcal{C}} \log p(\mathcal{C}|\mathcal{G})$, where $\mathcal{G}$, the molecular graph, is decomposed via chain rule,

$$\Delta_{\mathcal{C}} \log p(\mathcal{C}|\mathcal{G}) = f_{\mathcal{G}} \circ g_{\mathcal{G}}(\mathcal{C}) = f_{\mathcal{G}}(\mathbf{d}),$$

where $g_{\mathcal{G}}(\mathcal{C})$ denotes a function mapping atomic coordinates to a set of interatomic distances and $f_{\mathcal{G}}(\mathbf{d})$ is a GNN that estimates the negative energy of a molecule based on the interatomic distances $\mathbf{d}$. This interatomic distance matrix is invariant to rotation and translation of the underlying atomic coordinates $\mathcal{C}$. The prediction

of the negative energy of the molecule can then be backpropagated through the network $f_{\mathcal{G}}$ and the interatomic generation function $g_{\mathcal{G}}$ to create per-atom gradients for Langevin dynamics.

GeoDiff

In the GeoDiff model, by Xu et al. [109], diffusion models are applied to the problem of molecular conformation generation. Given a known molecular graph, GeoDiff generates possible molecular conformations through a diffusion model over the atomic coordinates. This approach uses an equivariant $SE(3)$ GNN to preserve network symmetries. GeoDiff functions by defining invariant probability densities, such that for any $g \in SE(3)$, the transform T_g has the property,

$$\forall g \in SE(3) : p(\mathbf{x}_t) = p(T_g(\mathbf{x}_t)).$$

The probability of a data sample at time t is invariant to any roto-translation $g \in SE(3)$. The initial density $p(\mathcal{C}^T)$ must also be invariant too. This is achieved by building on ideas from Equivariant flows by Kohler et al. [181], by moving the Center of Mass (CoM), the mean atomic coordinate, to the origin of the coordinate system. Secondly, the Markov transition kernels need to be equivariant. For any timestep, given a set of atomic coordinates $\mathcal{C}^t$ at time t, the following equation needs to hold:

$$p(\mathcal{C}^{t-1}|\mathcal{G}, \mathcal{C}^t) = p(T_g(\mathcal{C}^{t-1})|T_g(\mathcal{G}), T_g(\mathcal{C}^t)).$$

Using the same technique as before, the CoM is set to $\mathbf{0}$ at every time step to achieve translational invariance. Rotational equivariance of the Markov transition kernels is achieved through two factors, the reset CoM results in the scaling factor applied to the mean being applied equivariantly, and the covariance matrix of the diffusion process is isotropic, so the noise introduced at each step is invariant to rotation. As with DDPMs, the reverse process is defined through a network predicting the mean $\mu_\theta(\mathcal{C}^t, t)$ and a fixed isotropic variance schedule.

$$\mu_\theta(\mathcal{C}^t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathcal{C}^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta \left(\mathcal{G}, \mathcal{C}^t, t \right) \right),$$

Where $\mathcal{G}$ represents the known graph structure of the atomic system. Thus, to achieve equivariance, $\epsilon_\theta\left(\mathcal{G}, \mathcal{C}^t, t\right)$ must be an $SE(3)$ equivariant model. The model builds off work by Thomas et al. [113] and Satorras et al. [108] and defines a Graph Field Network (GFN). A GFN consists of equivariant GCN layers. In the l-th layer, GFN takes node embeddings $\mathbf{h}^l \in \mathbb{R}^{n\times b}$ a combination of atom and timestep information, and coordinate embeddings $\mathbf{x}^l \in \mathbb{R}^{n\times 3}$; and outputs $\mathbf{h}^{l+1}, \mathbf{x}^{l+1}$ as follows:

$$\mathbf{m}_{ij} = \Phi_m\left(\mathbf{h}_i^l, \mathbf{h}_j^l, ||\mathbf{x}_i^l - \mathbf{x}_j^l||^2, e_{ij}; \theta_m\right), \tag{2.50}$$

$$\mathbf{h}_i^{l+1} = \Phi_h\left(\mathbf{h}^l, \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}; \theta_h\right), \tag{2.51}$$

$$\mathbf{x}_i^{l+1} = \sum_{j\in\mathcal{N}(i)} \frac{1}{d_{ij}}\left(\mathbf{c}_i - \mathbf{c}_j\right)\Phi_x(\mathbf{m}_{ij}; \theta_x). \tag{2.52}$$

Note that $\mathbf{c}_i$ is the untransformed co-ordinates, i.e. $\mathbf{x}_i^0$. Unlike Thomas et al. [113] and other equivariant networks, this model does not learn any higher order features and is focused entirely on scalar and vector features. While the network architecture and Markov transition kernels are equivariant, the standard DDPM training objective is not. In a typical DDPM model, the prediction target is the noise vector ϵ [182]. However, ϵ in the forward diffusion process isn't constrained by equivariance. The term ϵ is typically calculated with:

$$\epsilon = \frac{\mathcal{C}^t - \sqrt{\bar{\alpha}_t}\mathcal{C}^0}{\sqrt{1-\bar{\alpha}_t}}.$$

GeoDiff proposes two solutions to this issue, firstly rotationally aligning $\mathcal{C}^0$ via the Kabsh algorithm [183] to $\mathcal{C}^t$. Instead of using ϵ, we align $\mathcal{C}^0$ to $\mathcal{C}^t$ to produce the aligned conformation $\hat{\mathcal{C}}^0$. Since this is aligned with $\mathcal{C}^t$, the noise vector $\hat{\epsilon}$ will also be equivariant.

$$\hat{\epsilon} = \frac{\mathcal{C}^t - \sqrt{\bar{\alpha}_t}\hat{\mathcal{C}}^0}{\sqrt{1-\bar{\alpha}_t}}$$

The second approach suggested relies upon the same strategy as ConfGF, but assumes that the noise applied to the distance matrix $\mathbf{d}$ is also normally distributed [184]. This approach results in higher quality samples.

2.5 Conclusion

An overview of proteins was presented along with existing classical and machine learning methods of predicting protein docking. Sections 2.2 and 2.3 presented an overview of the neural network architectures and distributions that could be potentially applied to the task of predicting protein dockings. In the final section, Energy Based Models were introduced. Generative modelling through Energy Based Models shares similarities with the thermodynamic perspective of protein folding, as both follow similar equations to arrive in low-energy (high probability) states. A trained protein Energy Based Model would ideally learn the physics of protein interactions, resulting in high-quality predictions.

CHAPTER 3

Shape Tracing

3.1 Introduction

Proteins are biopolymers directly responsible for the vast majority of essential cellular functions. Any organism, from a single bacterium to the human body, contains millions of proteins with roles as diverse as sensing, transportation, catalysis, defence or structural support. These biological tasks are often carried out via the formation of specific complexes of minimal energy. One of the hardest challenges in computational structural biology is predicting how individual proteins with a known atomic structure arrange into such assemblies.

Despite the advent of increasingly sophisticated methods over the last 20 years [185], the problem is far from solved. A yearly community-led assessment of current docking algorithms, CAPRI, reveals that at present, no algorithm exists that is capable of consistently yielding accurate results [186]. Existing approaches to this problem can be generally divided into two categories. In the first, proteins are represented explicitly as a collection of atoms. The objective here is to identify a protein arrangement that minimizes a scoring function composed of a sum of physical terms such as electrostatics, van der Waals, desolvation energy and other empirical quan-

tities. In the second, proteins are represented as a geometrical shape derived from the known atomic structure, and the associated scoring function typically maximizes shape complementarity. Independently from the chosen representation and scoring function, protein docking is a hard optimization problem. Indeed, protein-protein interactions feature a multitude of potential binding sites associated with a local energy minimum. The exploration of this complex search space, with high Lipschitz constants, is traditionally tackled either by brute force [30], or via derivative-free optimization algorithms such as Particle Swarm Optimization [187] or Monte Carlo [188]. Many candidate solutions generated during the docking process will feature either intersecting or contactless protein pairs, and optimization will often converge to local minima.

We present a method for the rapid exploration of the search space associated with the matching of two three-dimensional surfaces of arbitrary roughness and demonstrate its usage for protein docking. Instead of taking the traditional route of explicitly representing a protein surface [41, 189], we represent the receptor surface (i.e. the largest protein) implicitly, allowing for easy intersection and distance query. The ligand (i.e. the smaller protein) can then be marched by the lower bound of the boundary distance, as with traditional sphere tracing, but with a modification to the bound that allows tracing of arbitrary non-convex shapes. Our method features two key contributions. First, it leverages on a novel extension of sphere tracing [190], derived without heuristic, for detecting collisions between approaching non-convex shapes. Second, it adopts an implicit approach for finding where surface contact area is maximized, shown to be effective in a foundational outer-loop Monte Carlo method.

3.2 Related work

Protein docking algorithms can be broadly classified according to their sampling strategies [191]. These are Fast-Fourier Transform (FFT) grid-based searches [30, 192–194], Monte Carlo (MC) [188, 195, 196], Genetic Algorithms [197, 198] and Particle Swarm Optimisation (PSO) [39, 41, 199]. An additional strategy, Geometric

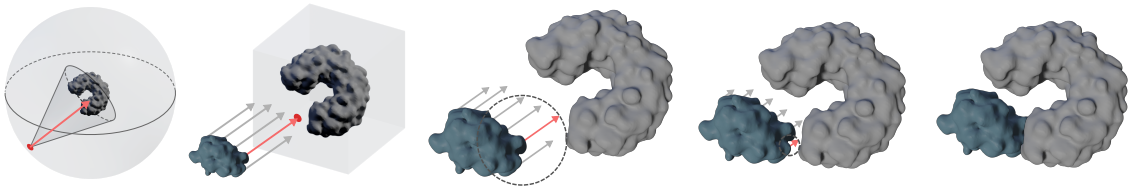


Figure 3.1: Docking with shape tracing: the source shape (ligand) is initialised on random points on a sphere around the target shape (receptor) with inward facing cones (1). The source shape is then analytically moved to be just inside the target's bounding box (2). The shape tracing algorithm iteratively (3-5) samples the target's signed distance function ϕ at surface positions. The shape is marched by the bound from the closest point (dashed circle, the minimum of this sample).

Hashing [33, 200], can only be adopted in conjunction with protein shape representations. Since proteins are inherently flexible molecules, conformational changes, from interfacial side-chain repacking to large-scale domain level rearrangements, should be accounted for by these algorithms. The majority of approaches use atomistic representations, which require computationally expensive additional minimization steps to promote side chains packing upon binding. Such considerations are needed as minor alterations in atomic positions may have profound consequences on the score of a pose. This step can be bypassed by modelling the protein as a shape, accounting for the uncertainty of side chains positions from the outset [201]. A method developed by Rudden and Degiacomi proposes a molecular surface representation, 'Spatial and Temporal Influence Density' (STID) [41], which maps points in space to a surface probability $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ based on a Molecular Dynamics simulation. This surface probability corresponds to the frequency in which a particular location was occupied during the Molecular Dynamics simulation. The map and subsequent surface complementarity scoring function [41] led to a success rate of 56% across a benchmark of 224 proteins, which is competitive with the current best atomic based methods [188, 196], and contrastingly performed remarkably well with flexible proteins. However, the time required to complete a full docking run using PSO was significant, often $>$ 1 day for larger complexes. Aside from computational time, there are still significant desirable improvements for protein docking algorithms. These include a scoring function which is exact and does not rely on heuristics, and an optimizer which can rapidly and reliably find correct solutions.

3.3 Methodology

The task of identifying the best arrangement of two non-convex protein shapes involves: (1) A shape tracing algorithm which efficiently marches the shape through space converging in a few steps and (2) A high-level optimizer to find solutions where contact between two proteins' surface area is maximised.

Shape tracing algorithm

The shape tracing algorithm updates points $\mathbf{x} \in \mathbb{R}^{n\times 3}$ on the boundary of the source shape (the ligand), moving them in a specified direction $\vec{v}$ until they collide:

$$S(\mathbf{x}, \phi, b, \vec{v}) = \mathbf{x}', \tag{3.1}$$

where ϕ is an input signed distance function (SDF) defined on a volumetric grid for the target shape (the receptor), and $b \in \mathbb{N}^3$ is the side length (in voxels) of each axis of this volume. The SDF ϕ can be calculated efficiently from the STID map [41] with the fast marching method as an approximate solution to the Eikonal equation. Any values sampled outside of the bounds of the SDF volume ϕ are set to ∞. The shape tracing method is outlined in Algorithm 3.1:

1. Compute the analytical intersections from rays cast on the source shape (ligand) at points $\mathbf{x}$ in direction $\vec{v}$ to the target (receptor) bounding box (lines 1-2) as in [202].

2. If any rays hit (line 4), advance all points $\mathbf{x}'$ by the closest distance (line 5). For glancing rays, push $\mathbf{x}'$ just inside box by the sign of $\vec{v}$ (line 6) as in [203].

3. Now we know one of the points in $\mathbf{x}'$ is inside the bounds of ϕ, find the closest distance to the receptor (line 7).

4. While the shapes are not touching $\delta > \epsilon$ and while the shape is still inside the bounds of ϕ (line 8), keep moving the whole shape $\mathbf{x}'$ by the closest distance (lines 9-10).

Input: $\mathbf{x}, \phi, b, \vec{v}$
Output: $\mathbf{x}'$
1 $t_{\text{nears}} = \max\big(\min\big((1/\vec{v}) \cdot (-\mathbf{x}\big), (1/\vec{v}) \cdot (b - \mathbf{x})\big)$
2 $t_{\text{fars}} \ = \min\big(\max\big((1/\vec{v}) \cdot (-\mathbf{x}\big), (1/\vec{v}) \cdot (b - \mathbf{x})\big)$
3 intersects $= \{t_{\text{nears}} > t_{\text{fars}}\}$
4 **if** intersects $\neq \{\}$ **then**
5 $\quad \delta = \min(t_{\text{nears}}[\text{intersects}])$
6 $\quad \mathbf{x}' = \mathbf{x} + \delta\vec{v} + \text{sign}(\vec{v})$
7 $\quad \delta = \min(\phi(\mathbf{x}'))$
8 $\quad$ **while** $\delta > \epsilon$ and $\delta \neq \infty$ **do**
9 $\quad\quad \mathbf{x}' = \mathbf{x}' + (\delta/2)\vec{v}$
10 $\quad\quad \delta = \min(\phi(\mathbf{x}'))$
11 $\quad$ **end**
12 **end**

Algorithm 3.1: Shape Tracing

The value for ϵ, 0 by default, can be increased for faster convergence if certain tolerances are acceptable, such as within 1 Å in docking. The $\delta/2$ in line 9 reduces the step (by a value proportional to the maximum derivative of ϕ), a common strategy in ray marching. While in theory we do not need to reduce this step as $|\nabla\phi| = 1$, in practice $|\nabla\phi| \approx 1$ due to the discretization of ϕ.

Outer-loop Monte Carlo docking

The shape tracing algorithm can be used to quickly move a shape through space without intersection. This is demonstrated in an outer-loop Monte Carlo docking method, which randomly rotates and moves the ligand to points on a sphere around the receptor, then fires the ligand towards the receptor at a random inward angle in a cone (Figure 3.1 left). This method is outlined in Algorithm 3.2:

1. Initialise a product manifold of two random points on a unit sphere (initial translation around the receptor and for the cone), and a random rotation (lines 3-5).

2. Rotate the ligand and translate it to the surface of the receptor's bounding sphere (lines 6-7).

3. Set the ray direction towards the receptor's centre, with some random variation γ to form a cone (lines 8-9).

4. Fire the ligand at the receptor (shape tracing), updating the positions $\mathbf{x}'$ (line 10).

5. Sum the contact surface area at the solution $\mathbf{x}'$ (Equation 3.2), and save the solution parameters if there is more contact than the previous best (lines 11-14).

Input: $\mathbf{x}_{\text{orig}}, \phi, b, \gamma$
1 $\alpha_{\text{best}} = 0$ ▷ surface area to maximize
2 **while** *true* **do**
3 $\quad \vec{t} =$ random point on unit sphere ▷ init translation
4 $\quad \vec{c} =$ random point on unit sphere ▷ for cone
5 $\quad R =$ random rotation matrix ▷ for ligand
6 $\quad s = \max(b)$ ▷ max receptor side length
7 $\quad \mathbf{x} = R\mathbf{x}_{\text{orig}} + \vec{t}s$ ▷ rotate & translate points
8 $\quad \vec{v} = (1-\gamma)(-\vec{t}) + \gamma\vec{c}$ ▷ construct cone
9 $\quad \vec{v} = \vec{v}/\|\vec{v}\|$
10 $\quad \mathbf{x}' = \textsc{ShapeTracing}(\mathbf{x}, \phi, b, \vec{v})$
11 $\quad \alpha_{\text{cur}} = \mathcal{L}(\phi, \mathbf{x}')$ ▷ contact area
12 $\quad$ **if** $\alpha_{\text{cur}} > \alpha_{\text{best}}$ **then**
13 $\quad\quad \alpha_{\text{best}} = \alpha_{\text{cur}}$
14 $\quad\quad$ save parameters
15 $\quad$ **end**
16 **end**

Algorithm 3.2: Outer-loop Monte Carlo Docking

The final score we maximize is the contact surface area between the receptor and the ligand: shape tracing, which prevents intersections, can also support symmetric profiles demonstrated by a C^∞ smooth delta, regularized by $\beta = 1$ (for 1 Å), with the loss $\mathcal{L}$:

$$\mathcal{L}(\phi, \mathbf{x}') = \int_\Omega \frac{\beta/\pi}{\beta^2 + \phi(\mathbf{x}')^2} \mathrm{d}\mathbf{x}' \tag{3.2}$$

This increases as the ligand approaches the receptor boundary, and is not influenced by points away from the surface (such as the back of the ligand).

3.4 Results & Discussion

We compared the performance of Monte Carlo with and without Shape Tracing in docking surfaces generated by STID maps. We took the average values of the best

dock over 10 runs and, for each run, sampling was terminated after 5000 iterations as further iteration yielded little improvement. Docking using shape tracing produced better solutions than naive sampling of ligand positions within the unit sphere (see Figure 3.2).

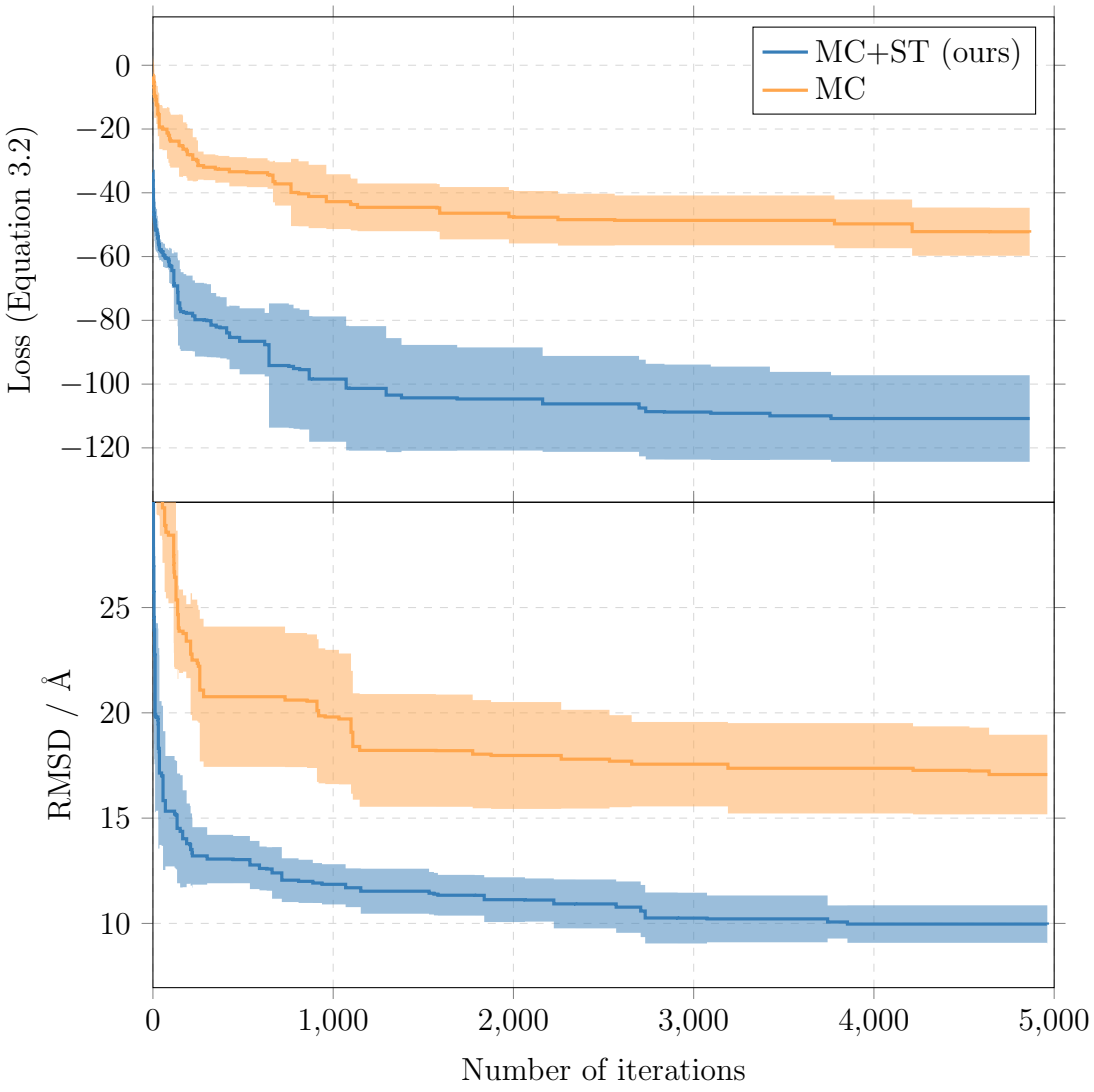


Figure 3.2: Combining Shape Tracing (ST) to a Monte Carlo (MC) search improves the performance of shape-based protein docking. MC+ST finds poses with both lower loss (top) and smaller RMSD with respect of the known docked pose (bottom). Example results from CAPRI unbound case 1AKJ are shown (ligand 2CLR, receptor 1CD8).

We investigated the relationship between cone width and solutions quality by varying the cone parameter γ. Best solutions after 5,000 iterations were averaged over 10 runs. A successful dock was defined as a ligand position within $\epsilon = 0.0001$ Å. We found that in general larger values of γ led to worse solutions (see Table

3.1) while a value of $\gamma = 0.05$ produced the best results. Slower tracing for higher values of γ is due to near-misses and oblique collisions requiring more steps of the marching algorithm than collisions perpendicular to the receptor's surface. While MC sampling without ST is significantly faster, it fails to produce many successful docks. In general, shape tracing produces more viable solutions to the docking problem.

Table 3.1: Wide cone angles have more misses and find worse solutions

	Baseline MC	Cone parameter γ				
	–	.00	0.05	0.10	0.15	0.20
Iterations/s	1040	83.8	83.5	82.4	79.5	75.9
Docks/s	0.001	83.8	83.5	82.4	79.5	75.6
Misses/s	–	0.00	0.00	0.012	0.04	0.351
Best RMSD at 5k	17.07	9.94	9.33	9.58	11.0	10.3
Std dev. at 5k	1.89	0.63	1.10	0.90	0.81	1.35

3.5 Limitations and future work

The current profile must be initialised away from the receptor, rather than also inside it, and therefore it can not find occluded solutions without modification. While in theory the current profile is sensible for finding exact solutions where contact is maximised without intersection, in practice successful docking poses often feature some intersection [41] due to the STID map not being able to consider inter-protein interactions that affect shape. Handling such cases can be achieved with an asymmetric energy profile that is negative for $x < 0$ instead of Equation 3.2, such that some degree of overlap is acceptable. This approach can be extended by replacing Equation 3.2 for a 1D function allowing intersections while minimising RMSD and other metrics (e.g. predicted interfacial residues) across a validation dataset.

3.6 Availability & Acknowledgements

The algorithm has been implemented using PyTorch on the GPU and is available at `https://github.com/cwkx/ShapeTracing` along with data for 14 additional test cases. The work was supported by the Engineering and Physical Sciences Research Council (EP/P016499/1).

3.7 Conclusion

We have found that shape marching with the updated bound significantly improves the convergence of finding non-intersecting solutions. The analytical ray-box intersection allows for quick evaluation of far-away solutions, and the shape tracing algorithm is able to march arbitrary non-convex shapes with a few inexpensive operations that can be calculated in parallel on a GPU. These collisions can then be used to seed other, more computationally intensive docking algorithms like PSO. In the future, work may aim at identifying an asymmetric energy profile which allows for some intersection to occur, to facilitate the discovery of biologically relevant docking poses.

CHAPTER 4

Diffusion Models on $SO(3)$

4.1 Introduction

Denoising diffusion probabilistic models are capable of generating high quality samples from complex distributions and have delivered encouraging results in audio synthesis and image applications. However, there are many problems such as pose estimation and protein docking for which the domain $\mathbb{R}^n$ is unsuitable. As many of these problems are roto-translational in nature, sampling rotations from a conditional diffusion model allows for a probabilistic model of possible poses. In this work, we introduce denoising diffusion models on the Lie group of 3D rotations, $SO(3)$.

DDPMs [8, 12] are a set of generative models inspired by non-equilibrium thermodynamics. The underlying idea consists of simulating a diffusion process that takes some form of observed data (e.g. images), denoted $\mathbf{x}_0$, with unknown distribution $q(\mathbf{x}_0)$ and transforms (diffuses) it into pure noise. A generative model can thus be found by learning the reverse process, turning noise back into the structure of the underlying data.

In practice, the diffusion is replaced by a non-homogenous discrete time Markov

chain with one-step transition density. The distribution at step t of the forward process Markov chain $q(\mathbf{x}_t)$ is conditional on only the step before it, $q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{4.1}$$

where $\beta_t, t = 1, \ldots, T$ denotes a variance schedule and $\mathcal{N}(y; \mu, \Sigma)$ a Gaussian density with argument y, mean μ and covariance matrix Σ. Under appropriate conditions, the final value $\mathbf{x}_T$ will approximately follow a Gaussian distribution $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$.

Denoising models learn an approximation of the reverse process $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ where $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The transition kernel $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ hence learns to predict the previous time step of the forward process and is parameterized by a normal distribution

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)). \tag{4.2}$$

The functions μ_θ and Σ_θ are implemented as outputs of a neural network with learned parameters θ. More recent work [8] suggests that taking the covariance matrix Σ_t in (4.2) fixed can result in better performance. Further reparameterization of the forward process $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + (1-\bar{\alpha}_t)\epsilon$. where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$ results in

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right). \tag{4.3}$$

The loss equation then can be simplified to a function of added noise,

$$L_t(\theta) = \mathbb{E}_{\tau, \epsilon, \mathbf{x}_0} \left[\left\| \epsilon - \epsilon_\theta(\mathbf{x}_\tau, \tau) \right\|^2 \right]. \tag{4.4}$$

4.2 Defining a diffusion on the rotation group $SO(3)$

A key component of training diffusion models is the ability to sample from the diffusion distribution at time t without having to calculate intermediate values. For the normal distribution in $\mathbb{R}^n$ (Euclidean diffusion), this can be accomplished easily through the previously derived closed form equations. However, this is not easily

generalised to the space of rotations $SO(3)$ due to several factors.

A naïve approach to diffusion on $SO(3)$ would be to use Euler angles, treating the rotations (ψ, ϑ, ϕ) as diffusion over an $\mathbb{R}^3$ space. However, the nature of diffusions over Euler angles means that they cannot correctly capture symmetries present in rotational systems. Alternatively, rotations may be represented as unit quaternions. However, a diffusion process treating the quaternion representation as $\mathbb{R}^4$ would require mapping the diffused position to the unit sphere at every step. The fast sampling scheme previously defined [8] would not be usable due to this nonlinear mapping. Similarly, a Gaussian distribution on $\mathbb{R}^4$ conditioned to lie on the 3-sphere is described by the Bingham distribution [150]. Additionally, while quaternions are a continuous representation of rotations, they form a double cover of $SO(3)$. This results in each rotation being represented by two equally valid quaternions, causing difficulties for neural networks [146].

Instead of attempting to use an $\mathbb{R}^n$ normal distribution on the space of rotations, we instead look for an analogous distribution defined on $SO(3)$. Before we proceed we outline some of the desirable properties of Gaussian distributions that are required for the distribution on $SO(3)$. For example, if $p_X(x), p_Y(y)$ are (independent) normal distributions defined as $\mathcal{N}(\mu_x, \sigma_x^2), \mathcal{N}(\mu_y, \sigma_y^2)$ respectively, then the distribution of their sum $p_Z(z)$ can be written as $Z \sim N(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$. For this reason, the normal distribution is said to be closed under convolution as the convolution of two normal densities (corresponding to the density of the sum) is also a normal distribution. This property is key in allowing us to derive the equation for the distribution of $q(\mathbf{x}_t|\mathbf{x}_0)$ and sample a diffusion model efficiently. A distribution for a diffusion on $SO(3)$ must also have this property in order to derive equations for efficient sampling of the forward process.

The Fisher matrix and Bingham distributions have previously been used in deep learning as probabilistic rotation estimators [155, 204] for ortho-normal matrix and quaternion representations of SO(3) respectively. However, neither of these distributions are closed under convolution [205, 206], making quantifying the rotational distribution at arbitrary timesteps difficult.

Instead, we consider the Isotropic Gaussian distribution on $SO(3)$ (IG) [207]

$g \sim \mathcal{IG}_{SO(3)}(\mu, \epsilon^2)$, parameterized by a mean rotation μ and scalar variance ϵ. The IG distribution can be parameterized in an axis-angle form, with uniformly sampled axes and rotation angle $\omega \in [0, \pi]$ with density

$$f(\omega) = \frac{1 - \cos\omega}{\pi} \sum_{l=0}^{\infty} (2l+1) e^{-l(l+1)\epsilon^2} \frac{\sin((l+\frac{1}{2})\omega)}{\sin(\omega/2)}. \tag{4.5}$$

Note that the uniform distribution on SO(3), denoted $\mathcal{U}_{SO(3)}$, is parameterized with uniform-axis and $f(\omega) = \frac{1-\cos\omega}{\pi}$ and needs to be included as a scaling factor when sampling from the distribution.

The IG distribution is both a natural extension of the central Limit Theorem (CLT) and the expected distribution of Brownian motion [208] on $SO(3)$, providing a strong motivation to use it for denoising diffusion models. While the variance of the IG distribution is defined as a scalar value, and thus has less flexibility than the Matrix-Fisher or Bingham distributions, Euclidean denoising diffusion models assume no correlation between dimensions, and are thus also parameterized with a scalar variance. Most importantly, the IG distribution is closed under convolution.

We can use this relationship to derive similar equations to the Euclidean diffusion process, allowing us to define the distribution at arbitrary timesteps for efficient sampling. Initially, data $\mathbf{x}_0 \in SO(3)$ is sampled from the distribution $q(\mathbf{x}_0)$ and diffused. As before, at $t = T$ we consider the data to be fully diffused.

To derive the distribution $q(\mathbf{x}_t|\mathbf{x}_0)$, note that the analogous Euclidean distribution (Equation 4.1) requires a scaling term to be applied to the $\mathbf{x}_0$ term. A scaling term applied directly to a rotation matrix is nonsensical, as the resulting value does not lie on the $SO(3)$ manifold. If $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ is viewed as a translation away from the origin, it implies that $\mathbf{x}_0 \sim s(\mathbf{x}_0)$ can be viewed as a rotation away from the identity rotation $\mathbf{I}$, and we can scale our rotations by interpolating the angle of rotation along the geodesic from the identity. For this, we rely upon the exponential and logarithmic maps between the Lie algebra $\mathfrak{so}(3)$, the set of skew-symmetric matrices and $SO(3)$. The Lie algebra is defined such that for any element $e^{tX} \in SO(3), t \in \mathbb{R}$, $X \in \mathfrak{so}(3)$. Intuitively, a rotation matrix R has an associated angle of rotation θ, and a rotation matrix $P = RR = R^2$ has an angle of rotation of 2θ. We follow

standard definitions [209] of the logarithm of a rotation matrix and define it as

$$\log R = \frac{\theta}{2\sin\theta}\left(R^\top - R\right)$$

where θ satisfies $1 + 2\cos\theta = \text{trace}(R)$. Matrices in $\mathfrak{so}(3)$ are skew-symmetric of form $S(v)$ with

$$S(v) = \begin{pmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{pmatrix}, \qquad v = [x, y, z],$$

where $\|v\|_2 = \theta$. From this definition of the rotation matrix logarithm, we are able to scale rotation matrices by converting them to values in the Lie algebra $\mathfrak{so}(3)$, element-wise multiplying by a scalar value and converting back to rotation matrices through matrix exponentiation. The composition of rotations is done through matrix multiplication in $SO(3)$, analogous to addition in Euclidean diffusion models:

$$\lambda(\gamma, \mathbf{x}) = \exp\left(\gamma \log\left(\mathbf{x}\right)\right). \tag{4.6}$$

Thus the function $\lambda(\gamma, \mathbf{x})$ is the geodesic flow from $\mathbf{I}$ to $\mathbf{x}$ by the amount γ. Applying these to equations from the original DDPM model we arrive at the following definitions:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{IG}_{SO(3)}(\lambda(\sqrt{\bar{\alpha}_t}, \mathbf{x}_0)), (1 - \bar{\alpha}_t)); \tag{4.7}$$

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{IG}_{SO(3)}(\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t), \tag{4.8}$$

and

$$\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \lambda\left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}, \mathbf{x}_0\right) \lambda\left(\frac{\sqrt{\alpha_{t-1}}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}, \mathbf{x}_t\right). \tag{4.9}$$

4.3 Experiments

4.3.1 Learning Data Distributions on $SO(3)$

To evaluate the validity of the proposed SO(3) diffusion process, a synthetic data distribution was created. This distribution consists of rotations about the z-axis of $\pm 90°$, with equal chance of either rotation. While simple, this model serves as a baseline to verify if multi-modal distributions can be successfully learned using the proposed $SO(3)$ diffusion process. A fully connected feed-forward network was trained using Algorithm 4.1. Inputs to the network consisted of raw rotation matrices and sine-cosine encoded t values.

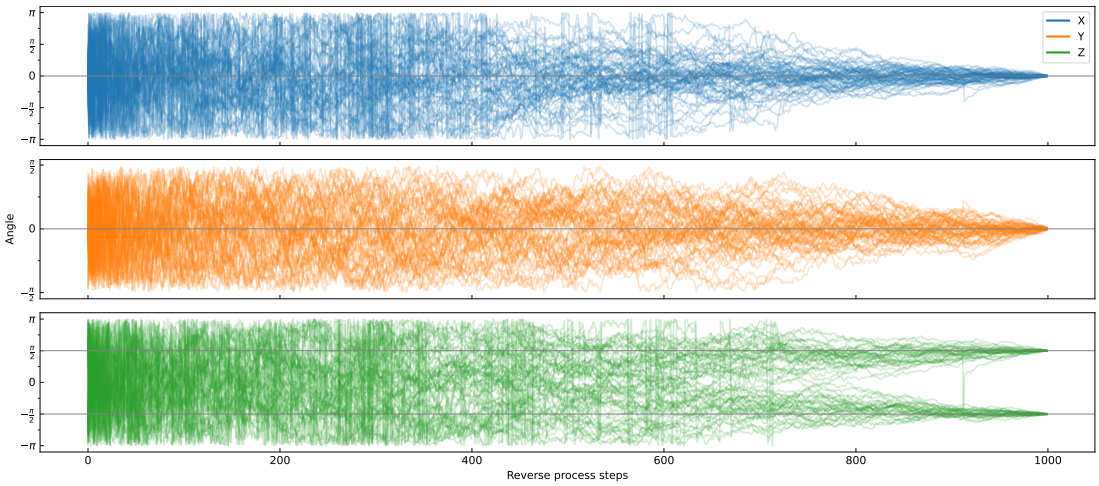


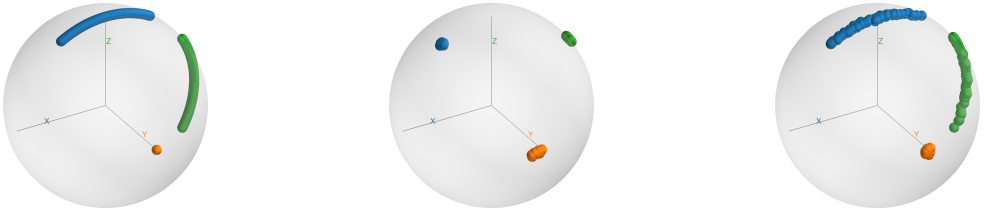
Figure 4.1: Decomposition of the learned $SO(3)$ reverse diffusion process into Euler angles shows r_θ has successfully learned the synthetic data distribution

Rotations generated from the reverse diffusion process closely approximate the initial data distribution. After generating 512 samples from the reverse process, the mean angular error between a sample and the closest rotation in the synthetic dataset was 0.0239 radians. The 512 samples were then categorised into z_{+90} (261 successes) and z_{-90} groups and modeled as a binomal distribution with probability of success p. Using a Beta($\alpha = 1, \beta = 1$) distribution as the conjugate prior, $\mathbb{P}(0.45 < p < 0.55 \mid z_{1:512}) = 0.963$, showing that the reverse process has also modeled the split of the data correctly.

While this shows that the denoising diffusion on $SO(3)$ is capable of learning a data distribution, it does not show whether it is any more capable than Euclidean

diffusion. To test this, we consider an idiomatic approach to diffusion of rotations: Euclidean diffusion of Euler angles; and show that this approach breaks down when the data distribution passes through singularities in the Euler representation.

We construct a data distribution of uniformly sampled rotations about the y-axis between $\frac{\pi}{3}$ and $\frac{2\pi}{3}$ radians (Figure 4.2a) and train two diffusion processes with equal network capacity until convergence. Figure 4.2b shows samples from the trained Euclidean–Euler process, revealing an inability to correctly capture the distribution, whereas the $SO(3)$ diffusion process (Figure 4.2c) provides a significantly better way of approximating the distribution and sampling.



(a) Target Distribution (b) Euler Angle Diffusion (c) $\mathcal{IG}_{SO(3)}$ Diffusion

Figure 4.2: Visualisation of the columns of sampled rotation matrices, corresponding to the transformation of basis vectors. Euler angle diffusion is unable to reconstruct the distribution correctly due to passing through the singularity at $\theta = \frac{\pi}{2}$. This failure on a simple distribution containing a singularity serves to show the limitations of Euler based diffusion.

To quantify the differences between the target distribution and samples generated from the diffusion process, we repeat this experiment with Quaternions sampled from Bingham distributions over $\mathcal{S}^3$, using the maximum mean discrepancy [210], with $k(x, y) = \exp(-||\log(x^{-1}y)||_F)$, a kernel over the $SO(3)$ geodesic distance.

Distribution	$\widehat{\text{MMD}}_b$	$p < 0.05$
Small Uncorrelated Rotations	0.0433	Yes
Large Uncorrelated Rotations	0.0027	Yes
Large Correlated Rotations	0.0317	Yes

Table 4.1: MMD estimates between generated samples and samples taken from Quaternion-Bingham distributions (See Appendix 4.7). Significance testing shows that the diffusion model correctly captures various distributions.

4.3.2 Rotational Alignment with $SO(3)$ Diffusion Processes

By using an $SO(3)$ diffusion process to represent the rotation of a pointcloud, we can use denoising diffusion models to generate solutions for a rotational alignment problem. In order to test this, an alignment problem using point-clouds of aircraft from ShapeNet [211] was designed. This dataset consists of pre-aligned point-clouds of various real and fictional aircraft. As the shape and style of these aircraft differ greatly, the task of rotating them to the correct orientation, in effect detecting the forward direction of the aircraft, relies on being able to generalize over common features. At each training iteration, rotations at arbitrary timesteps were generated and used to rotate samples from the dataset. The network architecture uses a non-causal transformer, with each token consisting of SIREN [212] encoded coordinates concatenated with a sinusoidal positional embedding of the timestep. We compare angular errors from Euler and $\mathcal{IG}_SO(3)$ diffusion models. After generating a single sample for each aircraft from both diffusion models, angular error compared to the known true orientation was calculated. We compare angular errors across the entire test set, and calculate the error of the test set at different percentiles.

	Percentile					
Method	1%	5%	10%	50%	90%	95%
Euler	0.64	1.09	1.37	2.26	2.97	3.05
$\mathcal{IG}_{SO(3)}$	**0.01**	**0.02**	**0.03**	**0.16**	**2.94**	**3.03**

Table 4.2: Distribution of angular error (radians) in an aircraft alignment task. Our method outperforms the Euler angle diffusion method across all percentiles. While both models failed to predict the aircraft orientation correctly for the worst 10% of aircraft, the $\mathcal{IG}_SO(3)$ model performs significantly better at predicting aircraft orientation across over half the test set.

4.4 Parameterization of Reverse Process

When training or sampling from a euclidean diffusion model, the stochastic part of the forward and reverse processes can be calculated by scaling a standard normal distribution. In DDPMs [8], the reverse process mean is given by Equation 4.3. As DDPMs resemble Langevin dynamics, with ϵ_θ predicting a learned gradient of

the data density, we run into the issue of what to predict in our $SO(3)$ model. We cannot directly predict values in $SO(3)$ as the gradient of a rotation matrix is not a rotation matrix. Instead, the gradient of a rotation matrix R lies on the tangent space $T_R SO(3)$ and takes the form $S(v)R$, where $S(v)$ is a skew symmetric matrix. Predicting the value $S(v)R$ directly involves predicting a point lying on a 3D hyperplane in a 9D space, and requires network knowledge of the rotation $R = \mathbf{x}_t$. We simplify the job of the network by predicting v instead, noting that $S(v)$ can also be interpreted as an angular velocity tensor. Secondly, sampling from $\mathcal{IG}_{SO(3)}(I, \lambda)$ cannot be done by scaling samples taken from $\mathcal{IG}_{SO(3)}(I, 1)$. Instead, during training (Algorithm 4.1), we sample our diffusion rotation matrix $R \sim \mathcal{IG}_{SO(3)}(\mathbf{I}, \sqrt{1-\bar{\alpha}_t})$, convert into skew-symmetric form through taking the matrix-logarithm and scale the target by $\frac{1}{\sqrt{1-\bar{\alpha}_t}}$ in order to have a target analogous to the standard normal distribution used when training a Euclidean diffusion model. This parameterisation allows us to follow a similar scheme to euclidean diffusion models when sampling (Section 4.6).

4.5 Sampling from the Isotropic Gaussian on $SO(3)$ distribution

First we note that a rotation sampled from $\mathcal{IG}_{SO(3)}(\mu, \epsilon^2)$ can be decomposed by sampling a pair of rotations from $(\mathcal{IG}_{SO(3)}(\mu, 0) = \mu,\ \mathcal{IG}_{SO(3)}(I, \epsilon^2))$ where I is the identity rotation. This corresponds to rotating samples taken from an identity-mean distribution by a constant μ. We further decompose $\mathcal{IG}_{SO(3)}(I, \epsilon^2)$ into an axis-angle form. The axis of rotation is uniformly distributed over $\mathcal{S}^2$, allowing for easy sampling, whereas the PDF of the angle of rotation $f(\omega) : \omega \in [0, \pi]$ is given by Equation 4.5.

Due to the $e^{-l(l+1)\epsilon^2}$ term in this equation, convergence of this equation is poor for small values of ϵ. We adapt the approximation derived in [157], suitable for

$\epsilon \leq 1$.

$$f(\omega) = \frac{\left(1-\cos(\omega)\right)}{\pi}\sqrt{\pi}\epsilon^{-\frac{3}{2}}e^{\frac{\epsilon}{4}}e^{-\frac{\left(\frac{\omega}{2}\right)^2}{\epsilon}}\cdot\frac{\left[\omega - e^{-\frac{\pi^2}{\epsilon}}\left(\left(\omega-2\pi\right)e^{\frac{\pi\omega}{\epsilon}} + \left(\omega+2\pi\right)e^{-\frac{\pi\omega}{\epsilon}}\right)\right]}{2\sin\left(\frac{\omega}{2}\right)} \tag{4.10}$$

We sample from an approximation of $f(t)$ using an inverse-transform sampling process, a numerical approximation of the CDF is taken through trapezoidal integration of the PDF, with a bias of more samples near 0 as to more accurately capture the distribution for small ϵ. Linear interpolation of uniform samples $[0, 1]$ are then used to approximate sampling from $f(t)$.

Once an angle has been sampled, an arbitrary axis is chosen from the uniform distribution, and the rotation composed with the mean rotation of the initial distribution. While in general the order of operations matters in SO(3), the isotropic nature of an identity-mean IG distribution means the order of matrix-multiplication does not affect the final distribution. Consider a sample from the distribution $\mathcal{IG}_{SO(3)}(\mu, \epsilon)$, which can be decomposed as $\mu\mathbf{z}_1$ or $\mathbf{z}_2\mu$. As $\mu\mathbf{z}_1 = \mathbf{z}_2\mu$, and $\mu\mathbf{z}_1\mu^{-1} = \mathbf{z}_2$, the matrices $\mathbf{z}_1$ and $\mathbf{z}_2$ are similar. Note that this implies $\mathrm{tr}(\mathbf{z}_1) = \mathrm{tr}(\mathbf{z}_2) = 1 + 2\cos\theta$, where θ is the angle of rotation. As $\mathbf{z}_1$ and $\mathbf{z}_2$ share the same angle of rotation and differ only in axis, which is uniformly sampled, the distributions of $\mu\mathbf{z}_1$ and $\mathbf{z}_2\mu$ are the same.

4.6 Training and Sampling algorithms for $SO(3)$ Diffusion

The neural network ϵ_θ learns an approximation of the gradient of the data density at each timestep t. In order to train this network we need to sample from $q(\mathbf{x}_t)$ efficiently. Once trained, the network is then used to generate samples matching the distribution $q(\mathbf{x}_0)$.

1: **repeat**

2: $t \sim \text{Uniform}(\{1, ..., T\})$

3: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$

4: $R \sim \mathcal{IG}_{SO(3)}(\mathbf{I}, \sqrt{1 - \bar{\alpha}_t})$

5: $S(v) = \frac{\log(R)}{\sqrt{1-\bar{\alpha}_t}}$

6: $\mathbf{x}_{scale} = \exp\left(\sqrt{\bar{\alpha}}_t \log \mathbf{x}_0\right)$

7: Take gradient descent step on:

$$\nabla_\theta ||v - \epsilon_\theta(R\mathbf{x}_{scale}, t)||_2$$

8: **until** converged

Algorithm 4.1: Training

1: $\mathbf{x}_T \sim \mathcal{U}_{SO(3)}$

2: **for** $t = T, ..., 1$ **do**

3: **if** $t > 1$ **then**

4: $R \sim \mathcal{IG}_{SO(3)}(\mathbf{I}, \tilde{\beta}_t)$

5: **else**

6: $R = I$

7: **end if**

8: $v = \epsilon_\theta(\mathbf{x}_t, t)$

9: $\mathbf{a}_1 = \exp(\frac{1}{\sqrt{\bar{\alpha}_t}} \log(\mathbf{x}_t))$

10: $\mathbf{a}_2 = \exp(S(\frac{1}{\sqrt{1-\bar{\alpha}_t}} v))$

11: $\tilde{\mathbf{x}}_0 = \mathbf{a}_1 \mathbf{a}_2^{-1}$

12: $\mathbf{x}_{t-1} = \tilde{\nu}(\mathbf{x}_t, \tilde{\mathbf{x}}_0) R$

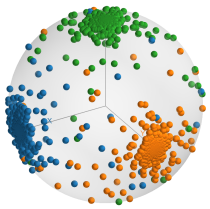
13: **end for**

Algorithm 4.2: Sampling, lines 9-11 reconstruct an estimate of $\mathbf{x}_0$, before using it to predict the mean of the previous timestep.

4.7 Visualisation of Quaternion-Bingham Distributions

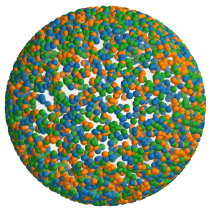
The distributions chosen in Table 4.1 were chosen to cover a range of possible distributions. A Bingham distribution is defined over S^n by an $(n + 1) \times (n + 1)$ covariance matrix. It can be thought of as samples from the Gaussian distribution

in $\mathbb{R}^{n+1}$ normalised to unit length. As unit vectors in $\mathbb{R}^4$ can be interpreted as quaternion rotations of the form $(x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k})$, we use this distribution to generate samples in $SO(3)$



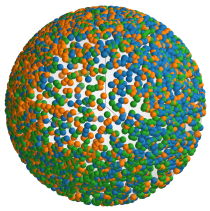
$$\begin{bmatrix} 10^3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Small Uncorrelated Rotations



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Large Uncorrelated Rotations



$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.9 & 0.9 \\ 0.0 & 0.9 & 1.0 & 0.9 \\ 0.0 & 0.9 & 0.9 & 1.0 \end{bmatrix}$$

(c) Large Correlated Rotations

Figure 4.3: Visualisation of the basis vectors of 512 samples of the each Quaternion-Bingham distribution and their respective co-variance matrices.

4.8 Conclusion

In conclusion, we have introduced a denoising diffusion probabilistic model over $SO(3)$ and shown that it is capable of correctly learning distributions over the space of rotations. Furthermore, we used this model as a framework for a synthetic alignment task, showing significantly lower errors than a naïve approach. This paper acts as an introduction to probabilistic sampling methods for alignment tasks. In the future, we aim to extend this method to $SE(3)$ and perform roto-translational alignment. In particular, we believe diffusion models are a suitable model for protein receptor-ligand interactions and can be used to probabilistically model the solution space to protein docking problems.

CHAPTER 5

Diffusion Models for Protein Docking

5.1 Introduction

Denoising diffusion probabilistic models are capable of generating high quality samples from complex distributions and have delivered encouraging results in audio synthesis and image applications. We consider alignment tasks, such as protein docking, for which traditional approaches consist of generating potential docking candidates as a form of sampling from an unknown distribution. We show that diffusion models can be reframed to generate candidate docks for alignment tasks.

However, the domain $\mathbb{R}^n$ is unsuitable. As many of these problems are roto-translational in nature, sampling rotations from a conditional diffusion model allows for a probabilistic model of possible poses. In this work, we introduce denoising diffusion models on the Lie group of 3D rotations, $SO(3)$.

DDPMs [8, 12] are a set of generative models inspired by non-equilibrium thermodynamics. The underlying idea consists of simulating a diffusion process that takes some form of observed data (e.g. images), denoted $\mathbf{x}_0$, with unknown distribution $q(\mathbf{x}_0)$ and transforms (diffuses) it into pure noise. A generative model can thus be found by learning the reverse process, turning noise back into the structure

of the underlying data.

In practice, the diffusion is replaced by a non-homogenous discrete time Markov chain with one-step transition density. The distribution at step t of the forward process Markov chain $q(\mathbf{x}_t)$ is conditional on only the step before it, $q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{5.1}$$

where $\beta_t, t = 1, \ldots, T$ denotes a variance schedule and $\mathcal{N}(y; \mu, \Sigma)$ a Gaussian density with argument y, mean μ and covariance matrix Σ. Under appropriate conditions, the final value $\mathbf{x}_T$ will approximately follow an Gaussian distribution $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$.

Denoising models learn an approximation of the reverse process $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ where $p(\mathbf{x}_T) = \mathcal{N}(0, \mathbf{I})$. The transition kernel $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ hence learns to predict the previous time step of the forward process and is parameterized by a normal distribution

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)). \tag{5.2}$$

The functions μ_θ and Σ_θ are implemented as outputs of a neural network with learned parameters θ. More recent work [8] suggests that taking the covariance matrix Σ_t in 5.2 fixed can result in better performance. Further reparameterisation of the forward process $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + (1-\bar{\alpha}_t)\epsilon$. where $\epsilon \sim \mathcal{N}(0, I)$ results in

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right). \tag{5.3}$$

The loss equation then can be simplified to a function of added noise.

$$L_t(\theta) = \mathbb{E}_{\tau,\epsilon,\mathbf{x}_0}\left[\left\|\epsilon - \epsilon_\theta(\mathbf{x}_\tau, \tau)\right\|^2\right]. \tag{5.4}$$

5.2 Sampling Alignments with Diffusion Models

Typically, reverse processes are used to generate high dimensional data from an unknown distribution that real-world data is sampled from. In typical tasks, this distribution's domain is a high dimensional euclidean space $\mathbb{R}^n$ where n is the dimen-

sion of the problem e.g. $n = 512 \times 512 \times 3$ for 2D colour images, $n = 128 \times 128 \times 128$ for greyscale volumetric data. However, in alignment tasks, those alignments lie on low dimensional manifolds such as $\mathbb{R}^2$, SO(3), SE(3), etc.. An alignment task can be thought of as determining the correct point on a known manifold by evaluating a transformation of the data (e.g. image, pointcloud, voxel data). The low dimensionality is not an issue for diffusion models, as early examples [12] performed well on low-dimensional datasets.

For example, consider a dataset D corresponding to a set of alignment problems. Each sample from the dataset d_i is a single docking problem. We wish to learn a diffusion process that given an alignment problem d_i, is able to generate samples of potential candidate alignments.

For all $t \in (0, T)$, $\mathbf{x}_t$ is a point on the low-dimensional manifold corresponding to the alignment task. By definition, $\mathbf{x}_0^i$ is the correct dock for an alignment problem d_i. We also define a domain-specific projection function $\phi(d_i, \mathbf{x}_t)$, that given an alignment problem d_i and data $\mathbf{x}_t$, produces a representation of the alignment problem transformed by $\mathbf{x}_t$. In the case of a alignment task on a euclidean manifold, the reverse process is then defined as

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\phi(d_i, \mathbf{x}_t), t)). \tag{5.5}$$

5.3 Pointcloud Gradient Prediction in SE(3)

In standard diffusion models, normal distribution samples for each dimension are independent of all other ones, thus a diffusion process in $\mathbb{R}^{n+m}$ with an identity covariance matrix, as typically used can be separated into $\mathbb{R}^n \times \mathbb{R}^m$. Similarly, we can decompose our diffusion process in $SE(3)$ into separate translational and rotational diffusion parts, i.e. $SE(3) = SO(3) \times \mathbb{R}^3$. To fully capture alignment in 3D space, the predicted data gradients must must lie on the tangent space of a point in $SE(3)$. As with $SO(3)$ tangent space prediction, predicting these values directly requires knowledge of the exact position of $\mathbf{x}_t$, which is untenable as in this framework $\mathbf{x}_0$ is the identity transformation of the "docked" configuration during

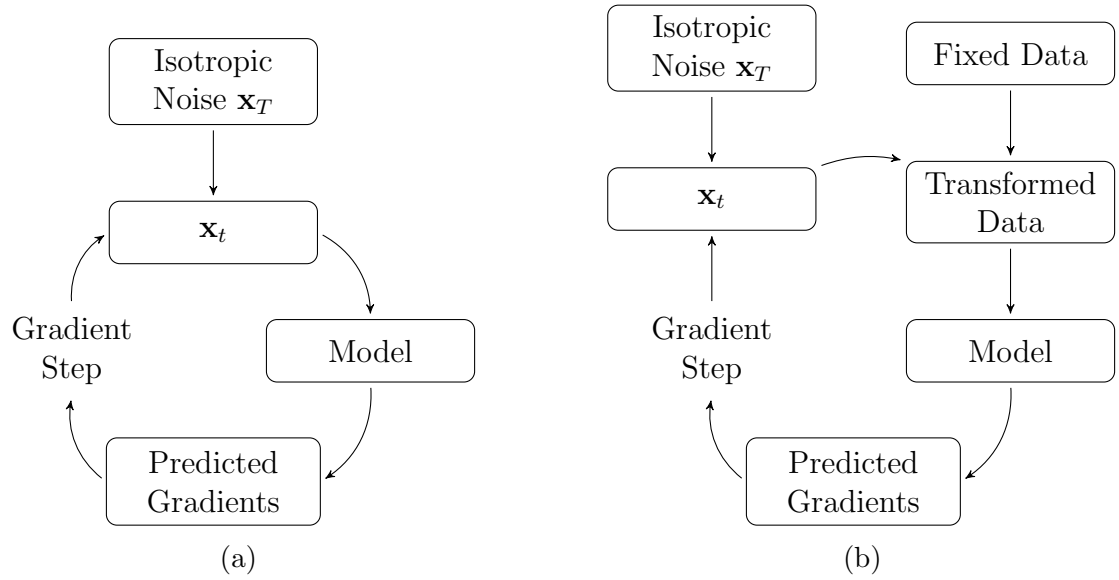


Figure 5.1: Comparison of (a) The traditional diffusion model sampling loop and (b) Diffusion models adapted for alignment.

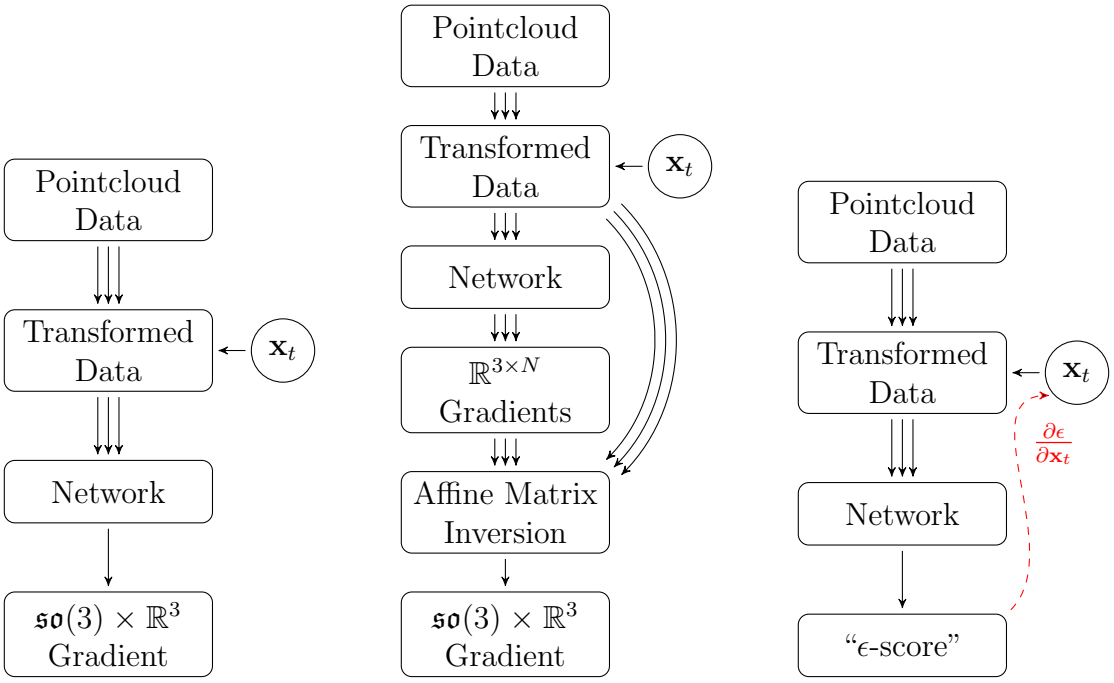
training, and thus knowledge of $\mathbf{x}_t$ results in "leakage" of information that can be used to directly reconstruct the true gradient rather than deriving it from the data.

Three approaches to predicting the rigid-body transform are proposed. The simplest method, direct prediction (Figure 5.2a) involves decomposing the gradient into two separate vectors $(\vec{v}, \vec{r})$. The vector $\vec{v}$ corresponds to the parametrisation of the $SO(3)$ reverse process in Section 4.2. The vector $\vec{r}$ is then treated as a standard euclidean diffusion model in $\mathbb{R}^3$ representing translational diffusion.

The second method, affine inversion (Figure 5.2b) effectively reduces the problem into predicting a per-point translational gradient, before reconstructing the $\mathfrak{so}(3) \times \mathbb{R}^3$ gradient. A rotation matrix R and a translation vector $\mathbf{b}$ can be used to construct an affine transformation matrix,

$$\begin{bmatrix} \mathbf{y}_i \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & R & & \mathbf{b} \\ 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \tag{5.6}$$

in order to apply rotation and translation to a known set of points $(\mathbf{x}_i)$. Given a known rotational $S(v) \in \mathfrak{so}(3)$ and translational $r \in \mathbb{R}^3$ gradient, it is also possible



(a) Direct (b) Affine Inversion (c) Backpropagation

Figure 5.2: Methods for Pointcloud $SE(3)$ gradient prediction, connections with multiple arrows imply information is carried through "per-point"

to calculate the per-point translational gradient $\Delta\mathbf{x}_i$ with a similar affine matrix,

$$\begin{bmatrix} \Delta\mathbf{x}_i \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & S(v) & & r \\ 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}. \tag{5.7}$$

If X is the matrix of augmented $\mathbf{x}_i$, and X_Δ the matrix of augmented $\Delta\mathbf{x}_i$ then affine gradient matrix A describes the relationship between the two as

$$X_\Delta = AX \tag{5.8}$$

If we predict $\Delta\mathbf{x}_i$ with out network, then reconstructing the matrix A can be done by solving the linear least squares problem

$$\min_A \|X^\top A^\top - X_\Delta\|_F \tag{5.9}$$

before projecting the found solution onto the space $\mathfrak{so}(3) \times \mathbb{R}^3$. Calculating solutions to linear least squares problems is a thoroughly explored, highly optimised and well

supported operation in many numerical computing libraries.

The third proposed method, backpropagation (Figure 5.2c) relies on predicting a "score" - ϵ similar to score matching methods found in EBMs [175]. This value is then backpropagated through the neural network to calculate the gradient of $\mathbf{x}_t$ with respect to ϵ. Given an $\mathbf{x}$ defined as rotation R and translation $\mathbf{b}$, the calculated rotational gradient differs from the other two approaches in being defined in terms of the tangent space $T_R SO(3)$. The gradient should take the form $S(v)R$, but may lie off of the tangent space as it is not enforced by this backpropagation operation. Thus, $S(v)$ must be calculated by applying the inverse rotation matrix then (as with the affine inversion method) projecting the result onto skew-symmetric form.

Of the three methods, backpropagation faces potential issues with memory usage during inference, as the values of the hidden layer activations must be kept in memory until the forward and backward passes are complete, instead of immediately discarded as with the other two approaches. Affine inversion may have difficulties training, as without regularisation, the $\mathbb{R}^{3\times N}$ gradients are not guaranteed to point in similar directions, and may result in poor solutions to the linear least squares problem (Equation 5.9). Overall direct prediction may be the best approach.

5.4 Protein Docking with Diffusion

Rigid protein docking can be viewed as a conditional diffusion model on the space $\mathbf{x}_0 \in SE(3)$. The distribution of docking poses $q(\mathbf{x}_0)$ is conditional on the sequences and structures of the receptor and ligand in question. Given a receptor R and ligand L we wish to sample from the conditional data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0|R, L)$ where $\mathbf{x}_0$ represents the position of the ligand in the docked position. A forward diffusion process, $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ steadily noises the position, such that at time $t = T$,

$$q(\mathbf{x}_T) = \mathcal{IG}_{SO(3)}(I, 1) \times \mathcal{N}(\mathbf{0}, 1). \tag{5.10}$$

That is, when fully diffused, the ligand's position is independent of the the distribution of the docked position $q(\mathbf{x}_0|R, L)$. We train a model p_θ that learns a reverse diffusion process, allowing us to iteratively sample each step $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, R, L)$.

5.4.1 Network Architecture

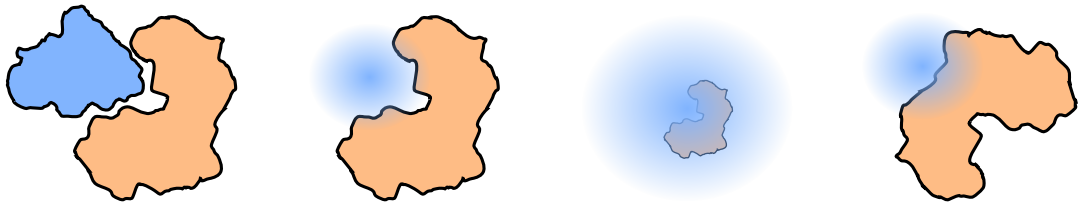
Network architecture choice remains an open question. EquiDock [213] use GNNs to predict docks, however the network architecture is invariant to the relative positions of the two proteins in space. This prevents the use of EquiDock-derived architectures for diffusion models. In general, GNN based protein-prediction architectures rely on nearest-neighbour graphs of the protein complex. For a diffusion model, this would require re-calculation at every step, with enforced inter-protein connections, as a naive nearest neighbour graph may result in separate subgraphs for each protein when the the receptor and ligand are far apart.

5.4.2 Diffusion Scaling

Euclidean diffusion processes sample $\mathbf{x}_T$ from the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$. However, protein structural data is typically defined in terms of Ångstroms (0.1 nm). There are some issues here, firstly that the scale of proteins means that a diffusion process using Ångstroms as the base unit would produce $\mathbf{x}_T$ positions still strongly clustered about the correct docking position (Figure 5.3b). This can be resolved by scaling the protein down so that that the distribution of $\mathbf{x}_T$ does not imply any particular position of the receptor to be the correct docking location (Figure 5.3c). However, the correct choice of "scale" is strongly influenced by the size of the proteins, a "universal" diffusion docker has to contend with a wide range of protein sizes. Thus, any form of diffusion scaling has to be presented to the model in a scale-agnostic way such that the model can correctly learn the scales of inter-molecular forces. This can also be achieved by diffusing the positions and rotations of both the receptor and ligand so that the docking location is randomly rotated from the ligand's position (Figure 5.3d).

5.5 Conclusion

In this chapter we have outlined approaches and potential challenges involved with extending $SO(3)$ diffusion to the modelling of $SE(3)$ alignment tasks. This would allow for diffusion models to predict protein receptor-ligand interactions. Further-



(a) (b) (c) (d)

Figure 5.3: A rigid docking problem with known solution (a) can have issues during training. Poorly scaled $\mathbf{x}_T$ translation distributions (b) allow the model to cheat by leaking information about the location of the binding site. Two options are proposed, either scaling the diffusion process (c) to prevent data leakage, or roto-translational diffusion over both receptor and ligand (d).

more, they may be used as a way of suggesting candidate docking positions from a learned probability distribution.

CHAPTER 6

Conclusions

Recent developments in the field of deep generative modelling has lead to increased interest in their application to models of protein kinematics. The primary scope of the prior work is extending and applying these recent developments to the space of protein kinematics. This thesis initially gives a detailed overview of the various topics and methods used in state-of-the-art algorithms within protein modelling, generative modelling, graph based and equivariant tasks. The thesis then shows a method of improving sample generation for classical docking pipelines, leveraging modern GPU parallelism to accelerate the production of higher quality intial docks. The thesis then continues by deriving a complete diffusion model over the space of 3D rotations - $SO(3)$ through careful choice of distribution and modification of a standard diffusion process. Finally, the thesis proposes a framework for the application of the $SO(3)$ diffusion model to conditional generative modelling of rigid protein-protein docking. The following section outlines the contributions and derivative work of the content present in this thesis.

6.1 Contributions

Work presented in Chapter 4 has been used successfully in a variety of generative protein modelling tasks. Luo et al. [214] successfully apply a joint sequence and structure diffusion model for the design of antibodies. Their method uses the proposed $SO(3)$ diffusion model to characterise the rotations of residues. Similarly, Yim et al. [215] present FrameDiff, where novel protein structures are generated through a derived equivariant diffusion process on $SE(3)$. Going further, Watson et al. [216] also use the results of Chapter 4 to describe the Brownian motion on the manifold of 3D rotation matrices, but extend their diffusion model to not only generate novel monomers, but symmetric multimers as well.

Drug discovery and modelling is typically interested in the interaction between protein binding sites and small molecules. Corso et al. [217] rely upon Chapter 4's $SO(3)$ diffusion to describe the global orientation of a small molecule, presenting an equivariant diffusion model capable of learning distributions of small-molecule protein binding sites. This is particularly useful for the investigation of symmetric protein complexes, as the model successfully learns the multi-modal distribution intrinsic to structures with multiple binding sites.

While primarily aimed at protein diffusion models, the methods presented have further applications elsewhere. In robotics, Ryu et al. [218] construct an EBM using results from the proposed $\mathcal{IG}_{SO(3)}$ diffusion model in order to produce a sample-efficient approach to visually learned robot manipulation of 3D objects.

Jagvaral et al. [219] notes a failure mode of the parametrisation of the reverse process from Chapter 4, and provides an alternative form. Jagvaral et al. then continue to use this method to generate synthetic datasets of galaxy and dark matter halo orientations that match the distributions of synthetic datasets produced by high-resolution hydrodynamic simulations [220]. This mirrors how $SO(3)$ diffusion models have been applied to protein modelling as an alternative to extremely computationally expensive molecular dynamics simulations.

6.1.1 Code Availability

Code for Chapters 3 and 4 is available at `https://github.com/cwkx/ShapeTracing` and `https://github.com/qazwsxal/diffusion-extensions` respectively.

6.2 Limitations and Future Work

Despite the promising performance and adoption of the proposed approaches, there are still some identifiable limitations, this section discusses the challenges and future directions of the current approaches presented in this thesis and the broader literature, including concurrent work to that presented here.

6.2.1 Necessity of $\mathcal{IG}_{SO(3)}$ Distribution

In Chapter 4, the isotropic gaussian on $SO(3)$ [157, 207] is proposed as a distribution suitable to create an analogous $SO(3)$ diffusion model. However, recent applications of diffusion models, using non-gaussian distributions [221] have shown that high quality samples can still be generated without the theoretical guarantees that gaussian diffusion processes possess. This is also true of protein based diffusion models, Anand et. al [222] use a diffusion model in which rotations are linearly interpolated between the true value and a uniform distribution, while still producing high quality results. As the calculation of isotropic gaussian terms is somewhat computationally intensive in comparison to linear interpolation, it may be prudent for newer models to use this simpler model.

6.2.2 Cryo-EM and X-ray Conditional Protein Generation

Given a known protein sequence, Cryogenic Electron Microscopy (Cryo-EM) and X-ray crystallography can be used to generate spatial occupancy maps [223, 224], effectively a low-resolution 3D model of the general shape of a protein. These maps can then be interpreted in order to construct an atomic model of the protein, which is then typically verified through molecular force field calculations [225,226]. However, as the residue sequence of this model is known, a diffusion model may be trained

to perform protein folding conditioned on the known spatial occupancy map. This would allow for generation of probable structural configurations of the residues that both fit the known 3D shape of the protein as well as being trained on structural motifs common among proteins.

6.2.3 Residue Prediction

Diffusion models have been applied to categorical predictive models [227] and are capable of describing multinomial distributions. In protein modelling, the representation of protein residues is typically done using a categorical distribution. By diffusing over residue type as well as position, a diffusion model would be capable of generating novel protein sequences as well as their structure. A diffusion model conditioned on binding partners; i.e. trained with a fixed receptor protein a diffused ligand protein; would be capable of generating novel protein sequences and structures that are capable of binding with a known protein, this would have immediate applications to drug discovery.

6.2.4 Residue Inpainting

One of the key advantages of diffusion models and score based generative modelling over other techniques is that the iterative nature allows for parts of the input samples to be fixed. Image inpainting - filling in or replacing sections of an image while maintaining the contents of a rest of the picture is easily achievable with diffusion models. Lugmayr et al. [228] achieve this by sampling masked sections of the input from the true image and unmasked sections from the reverse process. This produces images that are identical to the true image in masked regions and generate possible in-fills of the unmasked sections that are conditioned on the true values of the rest of the image. This can be applied to sequence and/or structure prediction of sections of an unknown protein. As each residue's position is independent, known sections of the backbone can be masked so that potential candidate residue sequences can be generated. Furthermore, as the binding sites of a protein are necessary for its interactions, the rest of the molecular structure acts only to influence their shape,

and potentially has sections with no effect at all. These sections can be unmasked and both position and type of amino acid can be generated from a residue inpainting diffusion process as a way to produce functionally similar yet unique synthetic proteins. The key application of this is in the avoidance of existing synthetic protein sequence patents, allowing for innovative drug design unencumbered by licensing issues.

6.2.5 Diffusion Bridges for Conformational Switching

Diffusion bridges are diffusion processes conditioned to initialise and terminate at two fixed states. These are typically sampled using Langevin MCMC methods [229]. Recent work in score matching and diffusion models [13, 230, 231] has resulted in the application of these diffusion bridges to synthetic datasets, and has shown techniques for stochastic interpolation between two samples using a learned diffusion process. Some proteins undergo conformational switches between two or more known states upon receiving an input signal such as photon reception, drug binding, or recognising molecules [232]. These protein switches are of particular interest as a change in protein structure influences the activity a protein has, regulating cellular signalling pathways. Given known conformations of a protein structure, a well trained generative diffusion model could be repurposed to generate diffusion bridges between these known states. As diffusion bridges are probabilistically favourable pathways between known states, diffusion bridges applied to protein conformation can be to generate physically realistic pathways that a folded protein may take between two known conformations.

Bibliography

[1] B. Wang, W. Yang, J. McKittrick, and M. A. Meyers, "Keratin: Structure, mechanical properties, occurrence in biological organisms, and efforts at bioinspiration," *Progress in Materials Science*, vol. 76, pp. 229–318, Mar. 2016. 1

[2] M. D. Shoulders and R. T. Raines, "Collagen Structure and Stability," *Annual Review of Biochemistry*, vol. 78, no. 1, pp. 929–958, 2009. 1

[3] F. B. Jensen, "The dual roles of red blood cells in tissue oxygen delivery: Oxygen carriers and regulators of local blood flow," *Journal of Experimental Biology*, vol. 212, pp. 3387–3393, Nov. 2009. 1

[4] M. W. Mosesson, "Fibrinogen and fibrin structure and functions," *Journal of Thrombosis and Haemostasis*, vol. 3, pp. 1894–1904, Aug. 2005. 1

[5] S. J. Pandol, *Digestive Enzymes*. Morgan & Claypool Life Sciences, 2010. 1

[6] G. S. Orf, C. Gisriel, and K. E. Redding, "Evolution of photosynthetic reaction centers: Insights from the structure of the heliobacterial reaction center," *Photosynthesis Research*, vol. 138, pp. 11–37, Oct. 2018. 1

[7] O. Narykov, S. Srinivasan, and D. Korkin, "Computational protein modeling and the next viral pandemic," *Nature Methods*, vol. 18, pp. 444–445, May 2021. 1

[8] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020. 1.1, 1.1, 2.4.1, 2.4.1, 4.1, 4.1, 4.2, 4.4, 5.1, 5.1

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks." https://arxiv.org/abs/1406.2661v1, June 2014. 1.1

[10] L. Pinheiro Cinelli, M. Araújo Marins, E. A. Barros da Silva, and S. Lima Netto, *Variational Autoencoder*, pp. 111–149. Cham: Springer International Publishing, 2021. 1.1

[11] A. Hyvärinen, "Estimation of Non-Normalized Statistical Models by Score Matching," *Journal of Machine Learning Research*, vol. 6, no. 24, pp. 695–709, 2005. 1.1, 2.4.1

[12] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *International Conference on Machine Learning*, pp. 2256–2265, PMLR, June 2015. 1.1, 2.4.1, 4.1, 5.1, 5.2

[13] J. Heng, V. De Bortoli, A. Doucet, and J. Thornton, "Simulating Diffusion Bridges with Score Matching," Oct. 2022. 1.1, 6.2.5

[14] F. H. Crick, "On protein synthesis," *Symposia of the Society for Experimental Biology*, vol. 12, pp. 138–163, 1958. 2.1.1

[15] J. Singh, J. Hanson, R. Heffernan, K. Paliwal, Y. Yang, and Y. Zhou, "Detecting Proline and Non-Proline Cis Isomers in Protein Structures from Sequences Using Deep Residual Ensemble Learning," *Journal of Chemical Information and Modeling*, vol. 58, pp. 2033–2042, Sept. 2018. 2.1.2

[16] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan, "Stereochemistry of polypeptide chain configurations," *Journal of Molecular Biology*, vol. 7, pp. 95–99, July 1963. 2.1.2

[17] A. L. Lehninger, D. L. Nelson, and M. M. Cox, *Lehninger Principles of Biochemistry*. New York: W.H. Freeman, 4th ed ed., 2005. 2.3

[18] J. Gao, Y. Hao, X. Piao, and X. Gu, "Aldehyde Dehydrogenase 2 as a Therapeutic Target in Oxidative Stress-Related Diseases: Post-Translational Modifications Deserve More Attention," *International Journal of Molecular Sciences*, vol. 23, p. 2682, Jan. 2022. 2.1.2

[19] C. V. Sindelar and K. H. Downing, "An atomic-level mechanism for activation of the kinesin molecular motors," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, pp. 4111–4116, Mar. 2010. 2.1.2

[20] D. Voet, J. G. Voet, and C. W. Pratt, *Fundamentals of Biochemistry: Life at the Molecular Level*. Hoboken, NJ: Wiley, 3rd ed ed., 2008. 2.1.2

[21] M. Mir, M. R. Stadler, S. A. Ortiz, C. E. Hannon, M. M. Harrison, X. Darzacq, and M. B. Eisen, "Dynamic multifactor hubs interact transiently with sites of active transcription in Drosophila embryos," *eLife*, vol. 7, p. e40497, Dec. 2018. 2.1.2

[22] P. E. Wright and H. J. Dyson, "Intrinsically Disordered Proteins in Cellular Signaling and Regulation," *Nature reviews. Molecular cell biology*, vol. 16, pp. 18–29, Jan. 2015. 2.1.2

[23] J. D. Bryngelson, J. N. Onuchic, N. D. Socci, and P. G. Wolynes, "Funnels, pathways, and the energy landscape of protein folding: A synthesis," *Proteins: Structure, Function, and Genetics*, vol. 21, pp. 167–195, Mar. 1995. 2.1.3

[24] I. A. Vakser, "Protein-Protein Docking: From Interaction to Interactome," *Biophysical Journal*, vol. 107, pp. 1785–1793, Oct. 2014. 2.1.4

[25] E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A. A. Friesem, C. Aflalo, and I. A. Vakser, "Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques.," *Proceedings of the National Academy of Sciences*, vol. 89, pp. 2195–2199, Mar. 1992. 2.1.4

[26] H. A. Gabb, R. M. Jackson, and M. J. Sternberg, "Modelling protein docking using shape complementarity, electrostatics and biochemical information," *Journal of Molecular Biology*, vol. 272, pp. 106–120, Sept. 1997. 2.1.4

[27] J. G. Mandell, V. A. Roberts, M. E. Pique, V. Kotlovyi, J. C. Mitchell, E. Nelson, I. Tsigelny, and L. F. Ten Eyck, "Protein docking using continuum electrostatics and geometric fit," *Protein Engineering, Design and Selection*, vol. 14, pp. 105–113, Feb. 2001. 2.1.4

[28] D. W. Ritchie and G. J. Kemp, "Protein docking using spherical polar Fourier correlations," *Proteins Struct. Funct. Genet.*, vol. 39, no. 2, pp. 178–194, 2000. 2.1.4

[29] I. A. Vakser, "Long-distance potentials: An approach to the multiple-minima problem in ligand-receptor interaction," *Protein Engineering, Design and Selection*, vol. 9, pp. 37–41, Jan. 1996. 2.1.4

[30] R. Chen, L. Li, and Z. Weng, "ZDOCK: An initial-stage protein-docking algorithm," *Proteins: Structure, Function, and Bioinformatics*, vol. 52, no. 1, pp. 80–87, 2003. 2.1.4, 3.1, 3.2

[31] D. Fischer, S. L. Lin, H. L. Wolfson, and R. Nussinov, "A geometry-based suite of moleculardocking processes," *Journal of Molecular Biology*, vol. 248, pp. 459–477, Jan. 1995. 2.1.4

[32] V. Venkatraman, Y. D. Yang, L. Sael, and D. Kihara, "Protein-protein docking using region-based 3D Zernike descriptors," *BMC Bioinformatics*, vol. 10, p. 407, Dec. 2009. 2.1.4

[33] D. Schneidman-Duhovny, Y. Inbar, R. Nussinov, and H. J. Wolfson, "PatchDock and SymmDock: Servers for rigid and symmetric docking," *Nucleic Acids Research*, vol. 33, no. SUPPL. 2, 2005. 2.1.4, 3.2

[34] M. Estrin and H. J. Wolfson, "SnapDock—template-based docking by Geometric Hashing," *Bioinformatics*, vol. 33, pp. i30–i36, July 2017. 2.1.4

[35] E. J. Gardiner, P. Willett, and P. J. Artymiuk, "Protein docking using a genetic algorithm," *Proteins: Structure, Function, and Bioinformatics*, vol. 44, no. 1, pp. 44–56, 2001. 2.1.4

[36] S. Sunny and P. B. Jayaraj, "FPDock: Protein–protein docking using flower pollination algorithm," *Computational Biology and Chemistry*, vol. 93, p. 107518, Aug. 2021. 2.1.4

[37] M. Kazemian, Y. Ramezani, C. Lucas, and B. Moshiri, "Swarm Clustering Based on Flowers Pollination by Artificial Bees," in *Swarm Intelligence in Data Mining* (A. Abraham, C. Grosan, and V. Ramos, eds.), Studies in Computational Intelligence, pp. 191–202, Berlin, Heidelberg: Springer, 2006. 2.1.4

[38] M. Torchala, I. H. Moal, R. A. G. Chaleil, J. Fernandez-Recio, and P. A. Bates, "SwarmDock: A server for flexible protein–protein docking," *Bioinformatics*, vol. 29, pp. 807–809, Mar. 2013. 2.1.4

[39] V. Namasivayam and R. Günther, "PSO@AUTODOCK: A fast flexible molecular docking program based on swarm intelligence," *Chemical Biology & Drug Design*, vol. 70, no. 6, pp. 475–484, 2007. 2.1.4, 3.2

[40] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, Nov. 1995. 2.1.4

[41] L. S. P. Rudden and M. T. Degiacomi, "Protein Docking Using a Single Representation for Protein Surface, Electrostatics, and Local Dynamics," *Journal of Chemical Theory and Computation*, vol. 15, pp. 5135–5143, Sept. 2019. 2.1.4, 3.1, 3.2, 3.3, 3.5

[42] L. S. P. Rudden and M. T. Degiacomi, "Transmembrane Protein Docking with JabberDock," *Journal of Chemical Information and Modeling*, vol. 61, pp. 1493–1499, Mar. 2021. 2.1.4

[43] M. T. Degiacomi and M. Dal Peraro, "Macromolecular symmetric assembly prediction using swarm intelligence dynamic modeling," *Structure (London, England : 1993)*, vol. 21, no. 7, pp. 1097–1106, 2013. 2.1.4

[44] B. Jiménez-García, J. Roel-Touris, M. Romero-Durana, M. Vidal, D. Jiménez-González, and J. Fernández-Recio, "LightDock: A new multi-scale approach to protein–protein docking," *Bioinformatics*, vol. 34, pp. 49–55, Jan. 2018. 2.1.4

[45] F. u. A. Afsar Minhas, B. J. Geiss, and A. Ben-Hur, "PAIRpred: Partner-specific prediction of interacting residues from sequence and structure," *Proteins: Structure, Function, and Bioinformatics*, vol. 82, no. 7, pp. 1142–1155, 2014. 2.1.4

[46] X. Wang, B. Yu, A. Ma, C. Chen, B. Liu, and Q. Ma, "Protein–protein interaction sites prediction by ensemble random forests with synthetic minority oversampling technique," *Bioinformatics*, vol. 35, pp. 2395–2402, July 2019. 2.1.4

[47] Z.-S. Wei, K. Han, J.-Y. Yang, H.-B. Shen, and D.-J. Yu, "Protein-protein interaction sites prediction by ensembling SVM and sample-weighted random forests," *Neurocomputing*, vol. 193, pp. 201–212, June 2016. 2.1.4

[48] R. Sanchez-Garcia, C. O. S. Sorzano, J. M. Carazo, and J. Segura, "BIPSPI: A method for the prediction of partner-specific protein–protein interfaces," *Bioinformatics*, vol. 35, pp. 470–477, Feb. 2019. 2.1.4

[49] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016. 2.1.4

[50] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein Interface Prediction using Graph Convolutional Networks," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. 2.1.4

[51] B. Zhang, J. Li, L. Quan, Y. Chen, and Q. Lü, "Sequence-based prediction of protein-protein interaction sites by simplified long short-term memory network," *Neurocomputing*, vol. 357, pp. 86–100, Sept. 2019. 2.1.4

[52] R. J. L. Townshend, R. Bedi, P. A. Suriana, and R. O. Dror, "End-to-end learning on 3D protein structure for interface prediction," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, no. 1401, pp. 15642–15651, Red Hook, NY, USA: Curran Associates Inc., Dec. 2019. 2.1.4

[53] Z. Xie, X. Deng, and K. Shu, "Prediction of Protein–Protein Interaction Sites Using Convolutional Neural Network and Improved Data Sets," *International Journal of Molecular Sciences*, vol. 21, p. 467, Jan. 2020. 2.1.4

[54] H. Zhu, X. Du, and Y. Yao, "ConvsPPIS: Identifying Protein-protein Interaction Sites by an Ensemble Convolutional Neural Network with Feature Graph," *Current Bioinformatics*, vol. 15, pp. 368–378, June 2020. 2.1.4

[55] Y. Liu, H. Yuan, L. Cai, and S. Ji, "Deep Learning of High-Order Interactions for Protein Interface Prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, (New York, NY, USA), pp. 679–687, Association for Computing Machinery, Aug. 2020. 2.1.4

[56] O. Martin and D. Schomburg, "Efficient comprehensive scoring of docked protein complexes using probabilistic support vector machines," *Proteins: Structure, Function, and Bioinformatics*, vol. 70, no. 4, pp. 1367–1378, 2008. 2.1.4

[57] P. Heuser and D. Schomburg, "Combination of scoring schemes for protein docking," *BMC Bioinformatics*, vol. 8, p. 279, Aug. 2007. 2.1.4

[58] S. Das and S. Chakrabarti, "Classification and prediction of protein–protein interaction interface using machine learning algorithm," *Scientific Reports*, vol. 11, p. 1761, Jan. 2021. 2.1.4

[59] X. Wang, G. Terashi, C. W. Christoffer, M. Zhu, and D. Kihara, "Protein docking model evaluation by 3D deep convolutional neural networks," *Bioinformatics*, vol. 36, pp. 2113–2118, Apr. 2020. 2.1.4

[60] Y. Cao and Y. Shen, "Energy-based graph convolutional networks for scoring protein docking models," *Proteins: Structure, Function, and Bioinformatics*, vol. 88, no. 8, pp. 1091–1099, 2020. 2.1.4

[61] A. Hadarovich, A. Kalinouski, and A. V. Tuzikov, "Deep Learning Approach with Rotate-Shift Invariant Input to Predict Protein Homodimer Structure," in *Bioinformatics Research and Applications* (Z. Cai, I. Mandoiu, G. Narasimhan, P. Skums, and X. Guo, eds.), Lecture Notes in Computer Science, (Cham), pp. 296–303, Springer International Publishing, 2020. 2.1.4

[62] V. K. Ramaswamy, S. C. Musson, C. G. Willcocks, and M. T. Degiacomi, "Deep Learning Protein Conformational Space with Convolutions and Latent Interpolations," *Physical Review X*, vol. 11, p. 011052, Mar. 2021. 2.1.5

[63] J. A. Maier, C. Martinez, K. Kasavajhala, L. Wickstrom, K. E. Hauser, and C. Simmerling, "ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB," *Journal of Chemical Theory and Computation*, vol. 11, pp. 3696–3713, Aug. 2015. 2.1.5

[64] R. A. Engh and R. Huber, "Structure quality and target parameters," in *International Tables for Crystallography*, ch. 18.3, pp. 474–484, John Wiley & Sons, Ltd, 2012. 2.1.5, 2.1.5

[65] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, Aug. 2021. 2.1.5

[66] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, pp. 706–710, Jan. 2020. 2.1.5, 2.1.5

[67] J. Pelé, H. Abdi, M. Moreau, D. Thybert, and M. Chabbert, "Multidimensional Scaling Reveals the Main Evolutionary Pathways of Class A G-Protein-Coupled Receptors," *PLoS ONE*, vol. 6, no. 4, 2011. 2.1.5

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. 2.2, 2.2.1

[69] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. 2.2

[70] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," July 2020. 2.2

[71] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962v2*, 2019. 2.2

[72] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," June 2021. 2.2

[73] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda, "Visual Transformers: Token-based Image Representation and Processing for Computer Vision," Nov. 2020. 2.2

[74] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, and R. Girshick, "Early Convolutions Help Transformers See Better," Oct. 2021. 2.2

[75] O.-E. Ganea, X. Huang, C. Bunne, Y. Bian, R. Barzilay, T. Jaakkola, and A. Krause, "Independent SE(3)-Equivariant Models for End-to-End Rigid Protein Docking," Mar. 2022. 2.2, 2.2.2, 2.3.1

[76] "A Primer on Kernel Methods," in *Kernel Methods in Computational Biology* (B. Schölkopf, K. Tsuda, and J.-P. Vert, eds.), The MIT Press, 2004. 2.2.4

[77] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems* (J. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), vol. 20, Curran Associates, Inc., 2007. 2.2.4, 2.2.4

[78] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, pp. 5156–5165, JMLR.org, July 2020. 2.2.4

[79] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," Feb. 2016. 2.2.4

[80] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)," July 2020. 2.2.4

[81] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking Attention with Performers," Nov. 2022. 2.2.4

[82] B. K. Sriperumbudur and Z. Szabo, "Optimal Rates for Random Fourier Features," Nov. 2015. 2.2.4

[83] D. J. Sutherland and J. Schneider, "On the Error of Random Fourier Features," June 2015. 2.2.4

[84] T. Yang, Y.-f. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou, "Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012. 2.2.4

[85] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019. 2.2.5

[86] 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," Mar. 2020. 2.2.5

[87] E. Kotsiliti, "De novo protein design with a language model," *Nature Biotechnology*, vol. 40, pp. 1433–1433, Oct. 2022. 2.2.5

[88] R. Rao, J. Meier, T. Sercu, S. Ovchinnikov, and A. Rives, "Transformer protein language models are unsupervised structure learners," Dec. 2020. 2.2.5

[89] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 118, p. e2016239118, Apr. 2021. 2.2.5

[90] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost, "ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. 2.2.5

[91] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context," June 2019. 2.2.5

[92] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," Jan. 2020. 2.2.5

[93] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," Feb. 2020. 2.2.5

[94] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," in *International Conference on Learning Representations*, Mar. 2020. 2.2.5

[95] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," July 2020. 2.2.5

[96] A. J. Bose, H. Ling, and Y. Cao, "Adversarial Contrastive Estimation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 1021–1032, Association for Computational Linguistics, July 2018. 2.2.5

[97] M. B. A. McDermott, B. Yap, H. Hsu, D. Jin, and P. Szolovits, "Adversarial Contrastive Pre-training for Protein Sequences," Jan. 2021. 2.2.5

[98] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges," May 2021. 2.3

[99] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016. 2.3.1

[100] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," Nov. 2016. 2.3.1

[101] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying Graph Convolutional Networks," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 6861–6871, PMLR, May 2019. 2.3.1

[102] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Honolulu, HI), pp. 5425–5434, IEEE, July 2017. 2.3.1

[103] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations*, Mar. 2023. 2.3.1

[104] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs," 2.3.1

[105] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional Networks on Graphs for Learning Molecular Fingerprints," Nov. 2015. 2.3.1

[106] C. Shi, S. Luo, M. Xu, and J. Tang, "Learning Gradient Fields for Molecular Conformation Generation," arXiv, June 2021. 2.3.1, 2.4.2

[107] K. Xu*, W. Hu*, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?," in *International Conference on Learning Representations*, Feb. 2022. 2.3.1

[108] V. G. Satorras, E. Hoogeboom, and M. Welling, "E(n) Equivariant Graph Neural Networks," Feb. 2022. 2.3.1, 2.3.1, 2.4.2

[109] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, "GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation," in *International Conference on Learning Representations*, Jan. 2022. 2.3.1, 2.4.2

[110] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph Matching Networks for Learning the Similarity of Graph Structured Objects," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 3835–3845, PMLR, May 2019. 2.3.1

[111] K. T. Schütt, O. T. Unke, and M. Gastegger, "Equivariant message passing for the prediction of tensorial properties and molecular spectra," June 2021. 2.3.1

[112] P. Thölke and G. D. Fabritiis, "Equivariant Transformers for Neural Network based Molecular Potentials," in *International Conference on Learning Representations*, Mar. 2022. 2.3.1

[113] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds," May 2018. 2.3.1, 2.3.3, 2.3.3, 2.3.4, 2.4.2, 2.4.2

[114] F. Fuchs, D. Worrall, V. Fischer, and M. Welling, "SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1970–1981, Curran Associates, Inc., 2020. 2.3.1, 2.3.3, 2.3.4

[115] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph Transformer Networks," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. 2.3.1

[116] J. T. Frank, O. T. Unke, and K.-R. Müller, "So3krates – Self-attention for higher-order geometric interactions on arbitrary length-scales," May 2022. 2.3.1

[117] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, Dec. 1989. 2.3.2

[118] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus,

S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017. 2.3.2

[119] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993. 2.3.2

[120] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991. 2.3.2

[121] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," June 2015. 2.3.2

[122] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. 2.3.2

[123] R. Gens and P. M. Domingos, "Deep Symmetry Networks," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014. 2.3.2

[124] J. F. Henriques and A. Vedaldi, "Warped convolutions: Efficient invariance to spatial transformations," 2017. 2.3.2

[125] D. Laptev, N. Savinov, J. Buhmann, and M. Pollefeys, "TI-POOLING: Transformation-invariant pooling for feature learning in convolutional neural networks," 2016. 2.3.2

[126] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proceedings of the 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 2990–2999, PMLR, June 2016. 2.3.2

[127] S. Dieleman, J. Fauw, and K. Kavukcuoglu, "Exploiting cyclic symmetry in convolutional neural networks," 2016. 2.3.2

[128] T. Cohen and M. Welling, "Steerable CNNs," 2017. 2.3.2

[129] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Oriented response networks," 2017. 2.3.2

[130] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. Brostow, "Harmonic networks: Deep translation and rotation equivariance," 2017. 2.3.2, 2.3.4

[131] O. T. Unke, M. Bogojeski, M. Gastegger, M. Geiger, T. Smidt, and K.-R. Müller, "SE(3)-equivariant prediction of molecular wavefunctions and electronic densities," 2021. 2.3.3

[132] B. C. Hall, *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. No. 222 in Graduate Texts in Mathematics, Cham ; New York: Springer, second edition ed., 2015. 2.3.3

[133] W. Greiner, B. Müller, and W. Greiner, *Quantum Mechanics*. No. v. 1-2 in Theoretical Physics, Berlin ; New York: Springer-Verlag, 2nd corr. ed ed., 1993. 2.3.3

[134] B. Anderson, T.-S. Hy, and R. Kondor, "Cormorant: Covariant Molecular Neural Networks," Nov. 2019. 2.3.3, 2.3.4

[135] M. A Morrison and G. A Parker, "A Guide to Rotations in Quantum Mechanics," *Australian Journal of Physics*, vol. 40, no. 4, p. 465, 1987. 2.3.3

[136] M. Winkels and T. S. Cohen, "3D G-CNNs for Pulmonary Nodule Detection," Apr. 2018. 2.3.4

[137] D. Worrall and G. Brostow, "CubeNet: Equivariance to 3D Rotation and Translation," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), Lecture Notes in Computer Science, (Cham), pp. 585–602, Springer International Publishing, 2018. 2.3.4

[138] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. S. Cohen, "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018. 2.3.4

[139] R. Kondor, "N-body Networks: A Covariant Hierarchical Neural Network Architecture for Learning Atomic Potentials," Mar. 2018. 2.3.4

[140] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions," Dec. 2017. 2.3.4

[141] R. J. L. Townshend, B. Townshend, S. Eismann, and R. O. Dror, "Geometric Prediction: Moving Beyond Scalars," June 2020. 2.3.4

[142] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, "E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials," *Nature Communications*, vol. 13, p. 2453, May 2022. 2.3.4

[143] C. Fefferman, S. Mitter, and H. Narayanan, "Testing the manifold hypothesis," *Journal of the American Mathematical Society*, vol. 29, pp. 983–1049, Oct. 2016. 2.3.5

[144] Y. Ma and Y. Fu, eds., *Manifold Learning Theory and Applications*. Boca Raton, Fla. : London: CRC ; Taylor & Francis [distributor], 2012. 2.3.5

[145] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), vol. 11218, pp. 282–299, Cham: Springer International Publishing, 2018. 2.3.5

[146] Y. Zhou, C. Barnes, L. Jingwan, Y. Jimei, and L. Hao, "On the continuity of rotation representations in neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2.3.5, 2.3.5, 4.2

[147] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics, New York: Springer-Verlag, 2nd ed ed., 1993. 2.3.5

[148] N. I. Fisher, T. Lewis, and B. J. J. Embleton, *Statistical Analysis of Spherical Data*. Cambridge University Press, first ed., Aug. 1987. 2.3.5

[149] K. V. Mardia and P. E. Jupp, *Directional Statistics*. Wiley Series in Probability and Statistics, Chichester ; New York: J. Wiley, 2000. 2.3.5

[150] C. Bingham, "An antipodally symmetric distribution on the sphere," *Annals of Statistics*, vol. 2, pp. 1201–1225, 1974. 2.3.5, 4.2

[151] K. Kunze and H. Schaeben, "The Bingham Distribution of Quaternions and Its Spherical Radon Transform in Texture Analysis," *Mathematical Geology*, vol. 36, pp. 917–943, Nov. 2004. 2.3.5

[152] T. C. Onstott, "Application of the Bingham distribution function in paleomagnetic studies," *Journal of Geophysical Research: Solid Earth*, vol. 85, no. B3, pp. 1500–1510, 1980. 2.3.5

[153] M. E. Antone and S. Teller, "Automatic Recovery of Camera Positions in Urban Scenes," Dec. 2000. 2.3.5

[154] P. Koev and A. Edelman, "The Efficient Evaluation of the Hypergeometric Function of a Matrix Argument," *Mathematics of Computation*, vol. 75, no. 254, pp. 833–846, 2006. 2.3.5

[155] I. Gilitschenski, R. Sahoo, W. Schwarting, A. Amini, S. Karaman, and D. Rus, "Deep Orientation Uncertainty Learning based on a Bingham Loss," in *Eighth International Conference on Learning Representations*, Apr. 2020. 2.3.5, 4.2

[156] T. Lee, "Bayesian attitude estimation with the matrix fisher distribution on SO(3)," *IEEE Transactions on Automatic Control*, vol. 63, no. 10, pp. 3377–3392, 2018. 2.3.5

[157] S. Matthies, J. Muller, and G. W. Vinel, "On the Normal Distribution in the Orientation Space," *Textures and Microstructures*, vol. 10, pp. 77–96, Jan. 1988. 2.3.5, 4.5, 6.2.1

[158] Y. Lecun, S. Chopra, R. Hadsell, M. A. Ranzato, and F. J. Huang, "A tutorial on energy-based learning," in *Predicting structured data* (G. Bakir, T. Hofman, B. Scholkopt, A. Smola, and B. Taskar, eds.), MIT Press, 2006. 2.4

[159] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky, "Your classifier is secretly an energy based model and you should treat it like one," in *International Conference on Learning Representations*, Mar. 2020. 2.4

[160] M. Á. Carreira-Perpiñán and G. Hinton, "On Contrastive Divergence Learning," in *International Workshop on Artificial Intelligence and Statistics*, pp. 33–40, PMLR, Jan. 2005. 2.4

[161] Y. Song and D. P. Kingma, "How to Train Your Energy-Based Models," Feb. 2021. 2.4

[162] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, pp. 1527–1554, July 2006. 2.4

[163] G. O. Roberts and R. L. Tweedie, "Exponential convergence of Langevin distributions and their discrete approximations," *Bernoulli*, vol. 2, pp. 341–363, Dec. 1996. 2.4

[164] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, (Madison, WI, USA), pp. 681–688, Omnipress, June 2011. 2.4, 2.4

[165] Y. Du and I. Mordatch, "Implicit Generation and Modeling with Energy Based Models," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. 2.4, 2.4

[166] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu, "A theory of generative ConvNet," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, (New York, NY, USA), pp. 2635–2644, JMLR.org, June 2016. 2.4

[167] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," Apr. 2016. 2.4

[168] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, and R. Zemel, "Learning the Stein Discrepancy for Training and Evaluating Energy-Based Models without Sampling," in *Proceedings of the 37th International Conference on Machine Learning*, pp. 3732–3747, PMLR, Nov. 2020. 2.4.1

[169] K. Swersky, M. Ranzato, D. Buchman, B. M. Marlin, and N. Freitas, "On autoencoders and score matching for energy based models," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, (Madison, WI, USA), pp. 1201–1208, Omnipress, June 2011. 2.4.1

[170] J. Shi, S. Sun, and J. Zhu, "A Spectral Approach to Gradient Estimation for Implicit Distributions," June 2018. 2.4.1

[171] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced Score Matching: A Scalable Approach to Density and Score Estimation," in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, pp. 574–584, PMLR, Aug. 2020. 2.4.1

[172] T. Pang, K. Xu, C. Li, Y. Song, S. Ermon, and J. Zhu, "Efficient learning of generative models via finite-difference score matching," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, (Red Hook, NY, USA), pp. 19175–19188, Curran Associates Inc., Dec. 2020. 2.4.1

[173] P. Vincent, "A Connection Between Score Matching and Denoising Autoencoders," *Neural Computation*, vol. 23, pp. 1661–1674, July 2011. 2.4.1

[174] S. Saremi, A. Mehrjou, B. Schölkopf, and A. Hyvärinen, "Deep Energy Estimator Networks," May 2018. 2.4.1

[175] Y. Song and S. Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution," *arXiv:1907.05600 [cs, stat]*, Oct. 2020. 2.4.1, 5.3

[176] B. Tzen and M. Raginsky, "Theoretical guarantees for sampling and inference in generative models with latent diffusions," in *Proceedings of the Thirty-Second Conference on Learning Theory*, pp. 3084–3114, PMLR, June 2019. 2.4.1

[177] G. Alain, Y. Bengio, L. Yao, J. Yosinski, É. Thibodeau-Laufer, S. Zhang, and P. Vincent, "GSNs: Generative stochastic networks," *Information and Inference: A Journal of the IMA*, vol. 5, pp. 210–249, June 2016. 2.4.1

[178] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized Denoising Auto-Encoders as Generative Models," in *Advances in Neural Information Processing Systems*, vol. 26, Curran Associates, Inc., 2013. 2.4.1

[179] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, pp. 79–86, Mar. 1951. 2.4.1

[180] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," *CoRR*, vol. abs/2102.09672, 2021. 2.4.1

[181] J. Köhler, L. Klein, and F. Noe, "Equivariant Flows: Exact Likelihood Generative Learning for Symmetric Densities," in *Proceedings of the 37th International Conference on Machine Learning*, pp. 5361–5370, PMLR, Nov. 2020. 2.4.2

[182] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv:2010.02502*, Oct. 2020. 2.4.2

[183] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, pp. 922–923, Sept. 1976. 2.4.2

[184] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2013. 2.4.2

[185] N. S. Pagadala, K. Syed, and T. Jack, "Software for molecular docking: A review," *Biophysical Reviews*, vol. 9, no. 2, pp. 91–102, 2017. 3.1

[186] M. F. Lensink, S. Velankar, and S. J. Wodak, "Modeling protein–protein and protein–peptide complexes: CAPRI 6th edition," *Proteins: Structure, Function, and Bioinformatics*, vol. 85, no. 3, pp. 359–377, 2017. 3.1

[187] S. Janson, D. Merkle, and M. Middendorf, "Molecular docking with multi-objective particle swarm optimization," *Applied Soft Computing*, vol. 8, no. 1, pp. 666–675, 2008. 3.1

[188] "The RosettaDock server for local protein-protein docking.," *Nucleic Acids Research*, vol. 36, no. Web Server issue, 2008. 3.1, 3.2

[189] S. Chaudhury, M. Berrondo, B. D. Weitzner, P. Muthu, H. Bergman, and J. J. Gray, "Benchmarking and analysis of protein docking performance in Rosetta v3. 2," *PloS one*, vol. 6, no. 8, 2011. 3.1

[190] J. C. Hart, "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces," *The Visual Computer*, vol. 12, no. 10, pp. 527–545, 1996. 3.1

[191] Zhang, Qiang, T. Feng, L. Xu, H. Sun, P. Pan, Y. Li, D. Li, and T. Hou, "Recent advances in protein-protein docking," *Current Drug Targets*, vol. 17, pp. 1586–1594, 2016. 3.2

[192] D. Kozakov, R. Brenke, S. R. Comeau, and S. Vajda, "PIPER: An FFT-based protein docking program with pairwise potentials," *Proteins Struct. Funct. Genet.*, vol. 65, pp. 392–406, Aug. 2006. 3.2

[193] "GRAMM-X public web server for protein-protein docking," *Nucleic Acids Research*, vol. 34, pp. W310–W314, July 2006. 3.2

[194] T. M. K. Cheng, T. L. Blundell, and J. Fernandez-Recio, "PyDock: Electrostatics and desolvation for effective scoring of rigid-body protein-protein docking," *Proteins Struct. Funct. Genet.*, vol. 68, pp. 503–515, Apr. 2007. 3.2

[195] M. Zacharias, "ATTRACT: Protein-protein docking in CAPRI using a reduced protein model," in *Proteins Struct. Funct. Genet.*, vol. 60, pp. 252–256, Proteins, Aug. 2005. 3.2

[196] C. Dominguez, R. Boelens, and A. M. J. J. Bonvin, "HADDOCK: A protein-protein docking approach based on biochemical or biophysical information," *Journal of the American Chemical Society*, vol. 125, pp. 1731–1737, 2003. 3.2

[197] E. J. Gardiner, P. Willett, and P. J. Artymiuk, "GAPDOCK: A genetic algorithm approach to protein docking in CAPRI round 1," *Proteins Struct. Funct. Genet.*, vol. 52, pp. 10–14, July 2003. 3.2

[198] "Software news and updates AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility," *Journal of Computational Chemistry*, vol. 30, pp. 2785–2791, Dec. 2009. 3.2

[199] I. H. Moal and P. A. Bates, "SwarmDock and the use of normal modes in protein-protein docking," *International Journal of Molecular Sciences*, vol. 11, no. 10, pp. 3623–3648, 2010. 3.2

[200] E. Mashiach, D. Schneidman-Duhovny, N. Andrusier, R. Nussinov, and H. J. Wolfson, "FireDock: A web server for fast interaction refinement in molecular docking.," *Nucleic Acids Research*, vol. 36, July 2008. 3.2

[201] Y. Yan and S.-Y. Huang, "Pushing the accuracy limit of shape complementarity for protein-protein docking," *BMC Bioinformatics*, vol. 20, no. 25, p. 696, 2019. 3.2

[202] T. L. Kay and J. T. Kajiya, "Ray tracing complex scenes," *ACM SIGGRAPH computer graphics*, vol. 20, no. 4, pp. 269–278, 1986. 1

[203] C. G. Willcocks, *Sparse Volumetric Deformation*. PhD thesis, Durham University, 2013. 2

[204] D. Mohlin, J. Sullivan, and G. Bianchi, "Probabilistic Orientation Estimation with Matrix Fisher Distributions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4884–4893, 2020. 4.2

[205] H. Schaeben and D. I. Nikolayev, "The Central Limit Theorem in Texture Component Fit Methods," *Acta Applicandae Mathematica*, vol. 53, pp. 59–87, Aug. 1998. 4.2

[206] J. Glover and L. P. Kaelbling, "Tracking 3-D Rotations with the Quaternion Bingham Filter," Mar. 2013. 4.2

[207] T. I. Savjolova, "*Preface to* novye metody issledovanija tekstury polikristalliceskich materialov," *Metallurgija, Moscow*, 1985. 4.2, 6.2.1

[208] Y. Qiu, "Isotropic Distributions for 3-Dimension Rotations and One-Sample Bayes Inference," *Graduate Theses and Dissertations*, Jan. 2013. 4.2

[209] J. R. Cardoso and F. S. Leite, "Exponentials of skew-symmetric matrices and logarithms of orthogonal matrices," *Journal of Computational and Applied Mathematics*, vol. 233, pp. 2867–2875, Apr. 2010. 4.2

[210] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. 25, pp. 723–773, 2012. 4.3.1

[211] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," *arXiv:1512.03012 [cs]*, Dec. 2015. 4.3.2

[212] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7462–7473, 2020. 4.3.2

[213] O.-E. Ganea, X. Huang, C. Bunne, Y. Bian, R. Barzilay, T. Jaakkola, and A. Krause, "Independent SE(3)-Equivariant models for end-to-end rigid protein docking," *CoRR*, vol. abs/2111.07786, 2021. 5.4.1

[214] S. Luo, Y. Su, X. Peng, S. Wang, J. Peng, and J. Ma, "Antigen-Specific Antibody Design and Optimization with Diffusion-Based Generative Models for Protein Structures," Oct. 2022. 6.1

[215] J. Yim, B. L. Trippe, V. De Bortoli, E. Mathieu, A. Doucet, R. Barzilay, and T. Jaakkola, "SE(3) diffusion model with application to protein backbone generation," Feb. 2023. 6.1

[216] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, N. Hanikel, S. J. Pellock, A. Courbet, W. Sheffler, J. Wang, P. Venkatesh, I. Sappington, S. V. Torres, A. Lauko, V. D. Bortoli, E. Mathieu, R. Barzilay, T. S. Jaakkola, F. DiMaio, M. Baek, and D. Baker, "Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models," Dec. 2022. 6.1

[217] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola, "DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking," Feb. 2023. 6.1

[218] H. Ryu, H.-i. Lee, J.-H. Lee, and J. Choi, "Equivariant Descriptor Fields: SE(3)-Equivariant Energy-Based Models for End-to-End Visual Robotic Manipulation Learning," Feb. 2023. 6.1

[219] Y. Jagvaral, F. Lanusse, and R. Mandelbaum, "DIFFUSION GENERATIVE MODELS ON SO(3)," Feb. 2023. 6.1

[220] Y. Jagvaral, R. Mandelbaum, and F. Lanusse, "Modeling halo and central galaxy orientations on the SO(3) manifold with score-based generative models," Dec. 2022. 6.1

[221] J. Wyatt, A. Leach, S. M. Schmon, and C. G. Willcocks, "AnoDDPM: Anomaly Detection with Denoising Diffusion Probabilistic Models using Simplex Noise," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (New Orleans, LA, USA), pp. 649–655, IEEE, June 2022. 6.2.1

[222] N. Anand and T. Achim, "Protein Structure and Sequence Generation with Equivariant Denoising Diffusion Probabilistic Models," May 2022. 6.2.1

[223] Y. Cheng, N. Grigorieff, P. A. Penczek, and T. Walz, "A Primer to Single-Particle Cryo-Electron Microscopy," *Cell*, vol. 161, pp. 438–449, Apr. 2015. 6.2.2

[224] C. Stoddart, "Structural biology: How proteins got their close-up," *Knowable Magazine*, Mar. 2022. 6.2.2

[225] S. J. Ludtke, M. L. Baker, D.-H. Chen, J.-L. Song, D. T. Chuang, and W. Chiu, "De novo backbone trace of GroEL from single particle electron cryomicroscopy," *Structure (London, England: 1993)*, vol. 16, pp. 441–448, Mar. 2008. 6.2.2

[226] T. C. Terwilliger*, P. D. Adams, P. V. Afonine, and O. V. Sobolev, "A fully automatic method yielding initial models from high-resolution electron cryo-microscopy maps," *Nature methods*, vol. 15, pp. 905–908, Nov. 2018. 6.2.2

[227] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, "Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 12454–12465, Curran Associates, Inc., 2021. 6.2.3

[228] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "RePaint: Inpainting using Denoising Diffusion Probabilistic Models," Aug. 2022. 6.2.4

[229] A. Beskos, G. Roberts, A. Stuart, and J. Voss, "Mcmc methods for diffusion bridges," *Stochastics and Dynamics*, vol. 08, pp. 319–350, Sept. 2008. 6.2.5

[230] V. De Bortoli, J. Thornton, J. Heng, and A. Doucet, "Diffusion Schr\"odinger Bridge with Applications to Score-Based Generative Modeling," Dec. 2021. 6.2.5

[231] V. R. Somnath, M. Pariset, Y.-P. Hsieh, M. R. Martinez, A. Krause, and C. Bunne, "Aligned Diffusion Schr\"odinger Bridges," Feb. 2023. 6.2.5

[232] J.-H. Ha and S. N. Loh, "Protein Conformational Switches: From Nature to Design," *Chemistry (Weinheim an der Bergstrasse, Germany)*, vol. 18, pp. 7984–7999, June 2012. 6.2.5