University of Oxford

Score-Based Diffusion Models for Protein Backbone Generation

by

Yuanhao Jiang

Wolfson College

A dissertation submitted in partial fulfilment of the degree of Master of Science in Statistical Science.

Department of Statistics, 24–29 St Giles,
Oxford, OX1 3LB

September 2025

This is my own work (except where otherwise indicated)

Candidate: Yuanhao Jiang

Signed:.....

Date:.....

Abstract

The abstract should go here.

Acknowledgements

I would like to thank the following:

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background on Diffusion Models

Diffusion models have recently emerged as one of the most powerful classes of deep generative models, achieving state-of-the-art performance in a wide range of domains, most prominently in image generation Ho et al. (2020); Dhariwal and Nichol (2021). These models construct a forward process that gradually corrupts data with Gaussian noise and train a neural network to approximate the reverse-time dynamics that iteratively remove noise. The resulting generative process is both probabilistically principled and empirically effective, offering advantages in sample quality and training stability compared to earlier paradigms such as variational autoencoders (VAEs) Kingma and Welling (2022) and generative adversarial networks (GANs) Goodfellow et al. (2020).

Score-based diffusion models Song et al. (2021b,a) improve discrete-time denoising diffusion probabilistic models with continuous-time stochastic differential equations (SDEs). Instead of directly parameterising the reverse process, they learn the score function—the gradient of the log-density of the noisy distribution—which enables the use of a variety of forward SDEs and corresponding reverse-time samplers. This framework has proven highly flexible and has been successfully applied to domains including image generation, audio synthesis, point clouds, and molecular data.

1.2 Motivation: Protein Structure Generation

Proteins are essential macromolecules whose biological functions are determined by their three-dimensional structures. The ability to generate novel protein structures has immense implications for biomedical science, including de novo protein design, drug discovery, and enzyme engineering. However, protein structures are highly constrained: bond lengths and angles must remain within narrow physical ranges, torsional angles follow characteristic distributions, and valid folds must satisfy global topological requirements. These constraints make the generative task significantly more challenging than in image or text domains.

Recent advances in deep learning, most notably AlphaFold Jumper et al. (2021), have demonstrated the power of data-driven models in predicting native structures from sequence. Yet, the problem of generating realistic, diverse backbones without explicit sequence conditioning remains an open challenge.

1.3 Diffusion Models for Protein Structures

The probabilistic formulation of score-based diffusion models makes them a natural candidate for modelling protein structures. Unlike GANs, diffusion models provide a likelihood-based framework, and unlike VAEs, they avoid restrictive parametric assumptions about the latent space. Moreover, their iterative denoising dynamics align well with the notion of refining approximate structures toward physically plausible conformations.

Recent work has demonstrated the potential of diffusion-based approaches in protein modelling and design. For example, RFdiffusion Watson et al. (2023) introduced a powerful

framework for de novo protein design, achieving unprecedented results in generating novel folds and functional structures. FoldingDiff Wu et al. (2024) proposed a diffusion-based model that explicitly learns the folding process, while DiffDock Yim et al. (2024) applied diffusion models to protein-ligand docking, highlighting the versatility of the paradigm in structural biology. These developments underscore the promise of diffusion generative models in capturing the rich geometric and biophysical constraints inherent to protein structures.

Applying diffusion models to proteins requires careful representation choices. In this work, we consider the protein backbone as a graph where nodes correspond to Cα atoms and edges encode both spatial proximity and sequential adjacency. This enables the use of graph neural networks (GNNs) Scarselli et al. (2009) as score models, incorporating geometric information through radial basis encodings, directional vectors, and positional embeddings. By training such models on large structural datasets, it becomes possible to learn score functions over protein conformational space and to generate new backbone structures through reverse diffusion.

1.4 Structure of the Dissertation

Section 2 presents the theoretical foundations of score-based diffusion, the representation of protein backbones, and the model architectures explored. Section 4 describes the experimental setup, training procedure, and evaluation metrics, and reports quantitative and qualitative results. And discusses the implications of our findings, limitations, and avenues for future research. Section 5 concludes the dissertation by summarising contributions and highlighting potential future directions.

2 Score-Based Diffusion for Generative Modelling

2.1 Background and Motivation

Generative modelling aims to learn a distribution from which new samples can be drawn that resemble those observed in a dataset. Classical approaches include variational autoencoders (VAEs) (Kingma and Welling, 2022), which maximise a variational lower bound on the likelihood, and generative adversarial networks (GANs) (Goodfellow et al., 2020), which frame generation as an adversarial game. While each of these paradigms has been successful, they also suffer from characteristic drawbacks such as blurry reconstructions (VAEs) and mode collapse (GANs).

Diffusion models represent a more recent class of generative models that address many of these limitations. The central idea is deceptively simple: one defines a forward process that gradually corrupts data with noise until the signal is destroyed, and then learns a reverse process that reconstructs data from noise. By formulating the forward process as a diffusion and training a neural network to approximate the reverse dynamics, diffusion models combine the flexibility of deep networks with the probabilistic rigour of latent-variable models.

In their original formulation, denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020) defined the forward process as a discrete-time Gaussian Markov chain. Subsequent work by Song et al. (2021b) showed that diffusion models can be generalised to continuous-time stochastic differential equations (SDEs). In this framework, the key object is the score function, i.e. the gradient of the log-density of the noisy distribution. Learning this score function allows one to simulate the reverse SDE and thereby generate new samples.

This chapter develops the theoretical foundations of score-based diffusion models in detail. We begin with the definition of the forward diffusion process, proceed to the derivation of the reverse-time dynamics, introduce score matching as the training objective, and describe practical sampling algorithms and noise schedules. Together, these elements form the mathematical basis for applying diffusion models to protein backbone generation in later chapters.

2.2 Forward Diffusion Processes

The first component of a diffusion generative model is the forward, or noising, process. This process gradually perturbs the data distribution into a tractable prior distribution, typically a standard Gaussian. The key requirement is that this transformation be both easy to sample from and analytically tractable, so that training objectives can be computed efficiently.

2.2.1 Discrete-Time Formulation

Denote clean data samples by $x_0 \sim p_{\text{data}}(x)$. Denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020) define the forward process as a Markov chain of N steps:

$$q\left(x_{1:N} \mid x_0\right) \coloneqq \prod_{i=1}^{N} q\left(x_i | x_{i-1}\right),$$

where each transition adds a small amount of Gaussian noise,

$$q\left(x_i \mid x_{i-1}\right) \coloneqq \mathcal{N}\Big(x_i; \sqrt{1-\beta_i}\, x_{i-1},\, \beta_i I\Big), \quad i = 1, ..., N. \tag{1}$$

This produces a sequence of progressively noisier variables $\{x_i\}_{i=1}^N$. Here, $\{\beta_i\}_{i=1}^N$ is a variance schedule with $\beta_i \in (0, 1)$ controlling the noise injected at each step.

A key property of the Gaussian forward process is that one can sample x_t at any arbitrary time step directly from the data x_0:

$$q\left(x_i \mid x_0\right) = \mathcal{N}\Big(x_i; \sqrt{\bar{\alpha}_i}\, x_0,\, (1-\bar{\alpha}_i) I\Big), \tag{2}$$

where $\alpha_i = 1 - \beta_i$ and $\bar{\alpha}_i = \prod_{s=1}^{i} \alpha_s$. This closed form is crucial for training, as it allows drawing noisy samples x_t without explicitly simulating all intermediate steps.

2.2.2 Continuous-Time Formulation

Song et al. (2021b) generalised the discrete diffusion process to continuous time by taking the limit as $N \to \infty$ and $\beta_i \to 0$. In this setting, the forward process can be expressed as a stochastic differential equation (SDE) of the form

$$dx = f(x, t)\, dt + g(t)\, dw, \tag{3}$$

where w is the Brownian motion, $f(x, t)$ is a drift term, and $g(t)$ controls the diffusion magnitude. The SDE produces a trajectory of continuously noisier variables $\{x(t)\}_{t\in[0,T]}$.

For DDPM Gaussian transition kernel eq. (1), the chain is equivalent to

$$x_i = \sqrt{1-\beta_i}\, x_{i-1} + \sqrt{\beta_i}\, z_{i-1}, \quad i = 1, ..., N,$$

where $z_{i-1} \sim \mathcal{N}(0, I)$. Define functions $\beta\left(\frac{i}{N}\right) \coloneqq N\beta_i$, $x\left(\frac{i}{N}\right) = x_i$ and $z\left(\frac{i}{N}\right) = z_i$, stepsize $\Delta t \coloneqq 1/N$ and finally reparameterise the time as $t = i\Delta t \in \{0, \Delta t, ..., N\Delta t = 1\}$, we have

$$\begin{aligned} x(t) &= \sqrt{1-\beta(t)\,\Delta t}\, x(t-\Delta t) + \sqrt{\beta(t)\,\Delta t}\, z\left(t-\Delta t\right) \\ &\overset{\Delta t \ll 1}{\approx} \left(1 - \frac{1}{2}\beta(t)\,\Delta t\right) x(t-\Delta t) + \sqrt{\beta(t)\,\Delta t}\, z\left(t-\Delta t\right), \end{aligned}$$

that is

$$x(t) - x(t-\Delta t) = -\frac{1}{2}\beta(t)\,\Delta t x(t-\Delta t) + \sqrt{\beta(t)\,\Delta t}\, z\left(t-\Delta t\right).$$

In the limit of $\Delta t \to 0$, the above equation converges to

$$dx = -\frac{1}{2}\beta(t)\,x dt + \sqrt{\beta(t)}dw. \tag{4}$$

Equation (4) is refered to as the Variance-preserving (VP) SDE, where $\beta(t)$ is the noise schedule (e.g. linear, cosine). Just like the discrete case of DDPM, VP SDE has a closed form solution. Define

$$u\left(x(t), t\right) = \exp\left(\frac{1}{2}\int_0^t \beta(s)\,ds\right) x(t)\,.$$

Then by Itô's chain rule, $y(t) \coloneqq u\left(x(t), t\right)$ has the SDE

$$\begin{aligned} dy &= \frac{1}{2}\beta(t)\exp\left(\frac{1}{2}\int_0^t \beta(s)\,ds\right) x(t)\,dt + \exp\left(\frac{1}{2}\int_0^t \beta(s)\,ds\right) dx \\ &= \exp\left(\frac{1}{2}\int_0^t \beta(s)\,ds\right)\left(\frac{1}{2}\beta(t)\,x(t)\,dt - \frac{1}{2}\beta(t)\,x(t)\,dt + \sqrt{\beta(t)}dw\right) \\ &= \exp\left(\frac{1}{2}\int_0^t \beta(s)\,ds\right)\sqrt{\beta(t)}dw. \end{aligned}$$

Integrate from 0 to t yields

$$y(t) = y(0) + \int_0^t \exp\left(\frac{1}{2}\int_0^s \beta(u)\,du\right)\sqrt{\beta(s)}dw_s.$$

Now undo the change of variables we have

$$\begin{aligned} x(t) &= \left[\exp\left(\frac{1}{2}\int_0^t \beta(s)\,ds\right)\right]^{-1} y(t) \\ &= \exp\left(-\frac{1}{2}\int_0^t \beta(s)\,ds\right)x(0) + \int_0^t \exp\left(-\frac{1}{2}\int_s^t \beta(u)\,du\right)\sqrt{\beta(s)}dw_s. \end{aligned}$$

The initial condition $x(0) = x_0$ is a sample from dataset, i.e., a constant, so the posterior of perturbation (or the prior of the reverse sampling) is a Gaussian distribution with mean

$$\mathbb{E}\left[x(t) \mid x(0) = x_0\right] = \exp\left(-\frac{1}{2}\int_0^t \beta(s)\,ds\right)x_0 \tag{5}$$

and variance

$$\begin{aligned} \mathrm{Var}\left[x(t) \mid x(0) = x_0\right] &= \int_0^t \exp\left(-\int_s^t \beta(u)\,du\right)\beta(s)\,ds I \\ &= \left(1 - \exp\left(-\int_0^t \beta(s)\,ds\right)\right) I \end{aligned} \tag{6}$$

that is

$$p(x_t|x_0) = \mathcal{N}\left(x_t :\ x_0 e^{-\frac{1}{2}\int_0^t \beta(s)ds}, \left(1 - e^{-\int_0^t \beta(s)ds}\right) I\right). \tag{7}$$

Obviously the variance is always bounded, hence the name variance-preserving. This continuous formulation provides a flexible mathematical foundation that unifies discrete DDPMs with stochastic differential equations, and it allows the use of stochastic calculus tools to analyse and manipulate diffusion models. In particular, it enables the derivation of the reverse-time SDE, which defines the generative process and will be introduced in the next section.

Other choices of f and g in eq. (3) yield different diffusion processes, Song et al. (2021b) shown three categories,

- **Variance-preserving (VP) SDE**: maintains the marginal variance of x_t at one throughout the process.
- **Variance-exploding (VE) SDE**: variance grows unbounded as $t \to T$, pushing the distribution to white noise.
- **Sub-variance-preserving (sub-VP) SDE**: an interpolation between VP and VE.

We will focus on VP SDE in this dissertation.

2.3 Reverse-Time Diffusion Processes

The forward diffusion process, whether in discrete or continuous form, progressively perturbs data until its distribution approaches a simple prior such as an isotropic Gaussian. To perform generative modelling, we require a process that inverts this corruption: starting from noise and evolving back toward the data distribution. This is formalised through the theory of time-reversal for diffusion processes.

2.3.1 Discrete-Time Reverse Process

In the discrete DDPM formulation Ho et al. (2020), the generative model is defined as a reverse Markov chain

$$p_\theta\left(x_{0:N}\right) \coloneqq p(x_N) \prod_{i=1}^{N} p_\theta(x_{i-1} \mid x_i),$$

with $p(x_N) = \mathcal{N}(x_N; 0, I)$ as the prior. The reverse conditionals are modelled as Gaussians

$$p_\theta\left(x_{i-1} \mid x_t\right) \coloneqq \mathcal{N}\big(x_{i-1}; \mu_\theta(x_i, i), \Sigma_\theta(x_i, i)\big),$$

where μ_θ and Σ_θ are learned using a neural network with parameter θ. Training then consists of minimising the Kullback-Leibler divergence between the true posterior $q(x_{i-1} \mid x_i, x_0)$ and the model distribution $p_\theta(x_{i-1} \mid x_i)$.

2.3.2 Continuous-Time Reverse SDE

In the continuous-time setting, Anderson (1982) established that the time reversal of an Itô SDE (3) is itself an SDE with modified drift term,

$$dx = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]\, dt + g(t)\, d\bar{w}, \tag{8}$$

where $\bar{w}$ is the Brownian motion running backward in time, $\nabla_x \log p_t(x)$ is the score function at time t, that is, the gradient of the log-density of the perturbed, marginal, data distribution $p_t(x)$. This equation defines the reverse-time SDE. It depends explicitly on the score function $\nabla_x \log p_t(x)$ of the forward process marginal distribution $p_t(x)$.

For VP SDE eq. (4), the reverse-time SDE is given by

$$dx = \left[-\frac{1}{2}\beta(t)\, x - \beta(t)\, \nabla_x \log p_t(x)\right] dt + \sqrt{\beta(t)} d\bar{w}.$$

Though it has no closed form sulution, if one can estimate the score function accurately for all times t, it becomes possible to simulate the reverse SDE and generate samples from the data distribution.

This result provides the key insight underlying score-based diffusion: generative modelling reduces to score estimation. Rather than directly modelling likelihoods or sampling distributions, we train a neural network $s_\theta(x, t)$ to approximate the score $\nabla_x \log p_t(x)$. The network is then used to guide the drift of the reverse SDE, producing realistic samples when integrated from Gaussian noise at $t = 1$ back to $t = 0$.

2.4 Score Matching and Training Objective

The reverse-time SDE depends on the score function $\nabla_x \log p_t(x)$, which is generally intractable. Thus, the central learning problem in score-based diffusion is to approximate this score with a neural network $s_\theta(x, t)$, called time dependent score network (TDSN) (Song et al., 2021b). Training requires a loss function that encourages s_θ to match the true score across noise levels.

2.4.1 Score Matching and Denoising Score Matching

For each time t, the score function of x_t can be trained via minimising the Fisher divergence

$$\mathbb{E}_{p(x_t)}\left[\left\|\nabla_{x_t} \log p\left(x_t\right) - s_\theta\left(x_t, t\right)\right\|_2^2\right].$$

The unknown term $\nabla_{x_t} \log p\left(x_t\right)$ (true score) can be eliminated with the score matching objective Hyvärinen (2005)

$$J_t^{\mathrm{SM}}(\theta) = \mathbb{E}_{p(x_t)}\left[\left\|\operatorname{tr}\left(\nabla_{x_t} s_\theta\left(x_t, t\right)\right) - \frac{1}{2}\|s_\theta(x_t, t)\|_2^2\right\|_2^2\right].$$

Note that the term $\operatorname{tr}\left(\nabla_{x_t} s_\theta\left(x_t, t\right)\right)$ can be computationally intensive. Vincent (2011) showed that training a denoising autoencoder to reconstruct clean data from corrupted observations is equivalent to score matching under certain conditions. This connection underpins the training of diffusion models: the neural network is trained to denoise x_t into x_0, thereby learning the score function implicitly. Hence, in the case of VP SDE, since we have the analytic form of the posterior, eq. (7), the denoising score matching objective is given by

$$J_t^{\mathrm{DSM}} = \mathbb{E}_{x_0 \sim p_{\mathrm{data}}(x_0)} \mathbb{E}_{x_t \sim p(x_t|x_0)}\left[\left\|s_\theta\left(x_t, t\right) - \nabla_{x_t} \log p\left(x_t|x_0\right)\right\|_2^2\right]$$

$$= \mathbb{E}_{x_0 \sim p_{\text{data}}(x_0)} \mathbb{E}_{x_t \sim p(x_t|x_0)} \left[\left\| s_\theta\left(x_t, t\right) + \frac{x_t - \mu_t}{\sigma_t^2} \right\|_2^2 \right]$$

where μ_t and σ_t^2 are given by eqs. (5) and (6), respectively.

The denoising score matching objective J_t^{DSM} is for a specific time t. To train the score model across all time $t \in [0, 1]$ in one go, the overall objective is given by a weighted sum (or average) (Song et al., 2021b)

$$J^{\text{DSM}} = \mathbb{E}_{t \sim \mathcal{U}(0,1)} \left[\lambda(t)\, J_t^{\text{DSM}} \right],$$

where $\lambda(t)$ is a positive weighting function.

The learning problem for score-based diffusion therefore reduces to denoising score matching: the neural network $s_\theta(x, t)$ is trained to approximate $\nabla_x \log p_t(x)$ across different noise levels. Once trained, this network can be used to drive the reverse SDE, enabling generation of new samples from Gaussian noise.

2.5 Sampling Procedures

Once the score network $s_\theta(x, t)$ has been trained, new samples can be generated by simulating the reverse diffusion process. This requires integrating the reverse-time SDE (8) starting from Gaussian noise $x(t = 1) \sim \mathcal{N}(0, I)$ and evolving toward $t = 0$.

2.5.1 Reverse SDE Sampling via Euler-Maruyama Method

The simplest approach is to discretise the reverse SDE using the Euler-Maruyama method. For a decreasing sequence of time steps $\{t_i\}_{i=0}^N$ with $t_N = 1$ and $t_0 = 0$, the update rule is

$$x_{t_{i-1}} = x_{t_i} + \left[f(x_{t_i}, t_i) - g(t_i)^2 s_\theta(x_{t_i}, t_i) \right] \Delta t + g(t_i) \sqrt{\Delta t}\, z_i, \tag{9}$$

where $z_i \sim \mathcal{N}(0, I)$ and $\Delta t = t_{i-1} - t_i$. This procedure generates approximate samples from the data distribution. While straightforward, its quality depends heavily on the number of discretisation steps: smaller steps yield higher fidelity at the cost of computational time.

2.5.2 Predictor-Corrector Sampling

The Predictor-Corrector sampler was proposed by Song et al. (2021b) as an improved sampling method. It is based on combining two components:

1. **Predictor step:** one update using the reverse SDE (e.g., via Euler-Maruyama method, eq. (9)), advancing the trajectory toward lower noise levels.
2. **Corrector step:** one or more steps of stochastic refinement, implemented as Langevin Monte Carlo:

   $$x \leftarrow x + \epsilon\, s_\theta(x, t) + \sqrt{2\epsilon}\, z, \quad z \sim \mathcal{N}(0, I),$$

   where ϵ is a step size. This step locally improves sample quality by pushing the state toward regions of higher density according to the trained score network.

By alternating predictor and corrector steps, the algorithm achieves a better trade-off between quality and efficiency than pure reverse SDE sampling. In practice, only a few corrector iterations per time step are needed to significantly improve sample fidelity.

In this dissertation, we use predictor-corrector sampling as our default procedure.

2.6 Noise Schedules

The design of the noise schedule plays a central role in the performance of score-based diffusion models. The schedule determines how the variance of the forward diffusion process evolves over time, which directly influences both training stability and the quality of generated samples.

Linear Schedules

In the original DDPM formulation (Ho et al., 2020), the variance schedule $\{\beta_i\}_{i=1}^N$ was chosen to increase linearly from a small value to a larger one over N steps. Equivalently, in the continuous SDE formulation (Song et al., 2021b), this corresponds to a linear growth of the noise rate

$$\beta(t) = \beta_0 + (\beta_1 - \beta_0)\, t.$$

While simple, this schedule can be suboptimal: early time steps may inject too little noise, leading to poor coverage of high-noise regions during training, while later steps may over-noise the data, resulting in wasted computation.

Cosine Variance-Preserving Schedule

Nichol and Dhariwal (2021) proposed a cosine schedule for the noising process in DDPM, which has since become widely adopted (Dhariwal and Nichol, 2021; Saharia et al., 2022; Watson et al., 2023). The proposed cosine schedule corresponds to using

$$\bar{\alpha}_i = \frac{\cos^2\left(\frac{\pi}{2} \cdot \frac{i/N+s}{1+s}\right)}{\cos^2\left(\frac{\pi}{2}\frac{s}{1+s}\right)}$$

in eq. (2) for noising process, where s is a very small offset (typically 10^{-4}) to prevent singularities, that is (ignoring the denominator in $\bar{\alpha}_i$ as it is nearly 1 for small s),

$$x_i = \sqrt{\bar{\alpha}_i}\, x_0 + \sqrt{1-\bar{\alpha}_i}\, z_i, \quad \bar{\alpha}_i \approx \cos^2\left(\frac{\pi}{2} \cdot \frac{i/N+s}{1+s}\right), \quad z_i \sim \mathcal{N}(0, I)\,.$$

This is a DDPM perturbation scheme. To generalise to continuous time, we do exactly the same as in section 2.2.2: define function $\bar{\alpha}\left(\frac{i}{N}\right) \coloneqq \bar{\alpha}_i$, stepsize $\Delta t \coloneqq 1/N$ and time as $t = i\Delta t \in \{0, \Delta t, ..., N\Delta t = 1\}$. This yields the continuous time schedule

$$x(t) = \sqrt{\bar{\alpha}(t)}\, x(0) + \sqrt{1-\bar{\alpha}(t)}\, z(t)\,, \quad \bar{\alpha}(t) = \cos^2\left(\frac{\pi}{2} \cdot \frac{t+s}{1+s}\right), \quad z(t) \sim \mathcal{N}(0, I)\,.$$

Match with previous derivation, eq. (7), we are left to solve

$$\bar{\alpha}(t) = e^{-\int_0^t \beta(s)ds}.$$

It follows that

$$\begin{aligned}\beta(t) &= -\frac{d}{dt}\log\cos^2\left(\frac{\pi}{2}\cdot\frac{t+s}{1+s}\right)\\ &= -\frac{1}{\cos^2\left(\frac{\pi}{2}\cdot\frac{t+s}{1+s}\right)}\left[2\cos\left(\frac{\pi}{2}\cdot\frac{t+s}{1+s}\right)\right]\left[-\sin\left(\frac{\pi}{2}\cdot\frac{t+s}{1+s}\right)\right]\frac{d}{dt}\left(\frac{\pi}{2}\cdot\frac{t+s}{1+s}\right)\\ &= \frac{\pi}{1+s}\tan\left(\frac{\pi}{2}\cdot\frac{t+s}{1+s}\right). \end{aligned} \tag{10}$$

Instead of linear growth, the cosine schedule ensures a smoother increase in noise variance, the noising strength changes much slower near $i = 1$ and $i = N$, as illustrated by fig. 1. This schedule maintains higher signal levels for longer, enabling the score network to learn more effectively across a wider range of noise intensities.

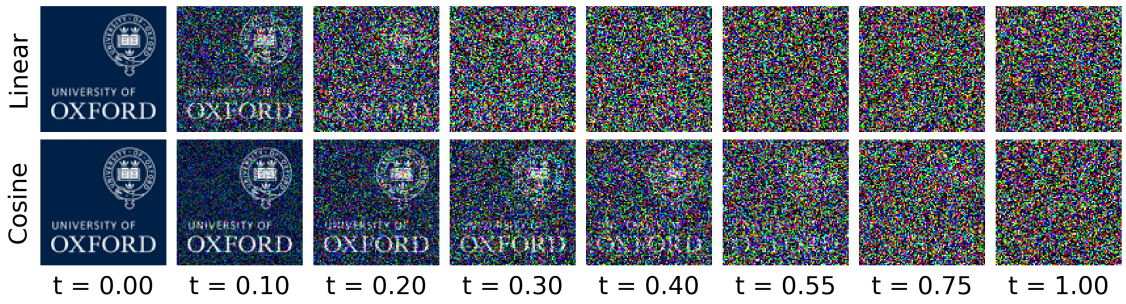

Figure 1: Linear schedule vs. cosine schedule.

Empirically, the cosine schedule improves sample quality and accelerates convergence compared to linear schedules. It balances the trade-off between injecting sufficient noise for diversity and preserving enough signal for stable training. Moreover, because it avoids excessively noisy late stages, fewer discretisation steps are needed during sampling, reducing computational cost.

In this dissertation, we adopt this cosine schedule as the forward diffusion process. This choice is motivated by its demonstrated effectiveness in image and structural generative tasks (Dhariwal and Nichol, 2021; Saharia et al., 2022; Watson et al., 2023), and by its compatibility with the variance-preserving SDE formulation used in our score model, eq. (10). All experiments in later chapters therefore use the cosine schedule as the default setting.

2.7 Summary

This chapter has established the theoretical foundations of score-based diffusion models, which form the core generative framework employed in this dissertation. We began by introducing the forward diffusion process, first in its discrete DDPM formulation and then in its continuous-time generalisation as a stochastic differential equation. We then derived the reverse-time SDE, showing that generative modelling reduces to the problem of score estimation. This led naturally to the denoising score matching objective, which enables training a neural network to approximate the score function across all noise levels. Finally, we discussed practical sampling algorithms, including reverse SDE integration

and predictor-corrector methods, and examined the role of noise schedules, with particular emphasis on the cosine variance-preserving schedule adopted in this work.

Together, these elements provide a principled probabilistic framework for deep generative modelling. Unlike GANs or VAEs, score-based diffusion models avoid common issues such as mode collapse or restrictive latent assumptions, while offering strong theoretical guarantees and flexible sampling procedures. The remainder of this dissertation builds upon these foundations to adapt score-based diffusion to the setting of protein backbone generation, where structural constraints and geometric representations introduce unique modelling challenges.

3 Protein Structure Modelling with Graph Neural Networks

3.1 Proteins and Structural Constraints

Proteins are linear polymers of amino acids that fold into three-dimensional structures. Their backbone is a repeating chain of atoms N-C_α-C, with side chains branching from C_α. In this work we adopt a C_α-only representation: each residue $i \in \{1, \ldots, L\}$ is associated with a 3D coordinate $x_i \in \mathbb{R}^3$. This representation is widely used for coarse-grained modelling because it captures the global geometry of the fold while keeping the state space compact.

Even at this coarse level, protein geometry is far from arbitrary. Neighbouring C_α atoms exhibit a characteristic virtual bond length

$$d_i \;=\; \|x_i - x_{i-1}\| \approx 3.8 \text{ Å},$$

and local shape can be described by the virtual bond angle

$$\theta_i \;=\; \angle(x_{i-1}, x_i, x_{i+1}),$$

and the virtual dihedral

$$\tau_i \;=\; \text{dihedral}(x_{i-2}, x_{i-1}, x_i, x_{i+1}),$$

whose empirical distributions are constrained by backbone sterics and peptide planarity. Beyond local geometry, valid structures avoid steric clashes (no two non-neighbour residues at unrealistically short distances) and typically exhibit compact, physically plausible folds.

Scope of this work. Our goal is backbone-level generative modelling using score-based diffusion. We do not impose hard geometric constraints during training or sampling (e.g. fixed bond lengths/angles); instead, we learn the distribution over C_α coordinate trajectories and evaluate fidelity against native backbones. Section 4 reports RMSD/TM results and discusses the limitation that high RMSD/low TM can still correspond to physically valid but non-native folds.

Paired evaluation vs. unconditional validity. Because each generated backbone is compared to its specific native target of the same length L, RMSD/TM measure reconstruction fidelity, not absolute physical validity. This is appropriate for our benchmarking setting (paired evaluation), but one should interpret low TM or high RMSD with caution: a sample can be a plausible alternative fold while scoring poorly against the chosen native reference.

3.2 Graph Representation of Backbones

A protein backbone of length L is represented as a graph $G = (V, E)$, where each residue corresponds to a node and edges capture both geometric proximity and sequential

connectivity. This formulation enables the use of graph neural networks as score models, allowing information to be propagated through local neighbourhoods while respecting the spatial structure of the backbone.

Nodes. Each node $i \in V$ corresponds to a residue with coordinate $x_i \in \mathbb{R}^3$ representing the position of its C_α atom. For simplicity, we do not include residue type or side-chain information in this work, focusing purely on the geometry of the backbone. The node features are initialised as positional embeddings derived from the residue index (Section 3.3), along with optional time-conditioning features from the diffusion process.

Edges. Edges are constructed using a radius graph: for each node i, we connect edges to all neighbours j within a cutoff distance r, subject to a maximum of k nearest neighbours. Formally,

$$(i, j) \in E \quad \Leftrightarrow \quad \|x_i - x_j\| \leq r \quad \text{and } j \text{ is among the } k \text{ closest to } i.$$

This design ensures that local spatial relationships are captured without requiring a fixed sequence length. In addition to geometric edges, self-loops are included to stabilise message passing and allow each node to retain its own features. Edges are directed: for each $(i, j) \in E$, messages are passed from i (source) to j (target). This convention is consistent with the implementation of `TransformerConv` in PyTorch Geometric, and we do not duplicate edges to create an undirected graph.

Edge features. For each edge $(i, j) \in E$, we compute a feature vector combining geometric and diffusion-time information:

- **Radial basis encoding:** The distance $\|x_i - x_j\|$ is expanded into a vector using Gaussian radial basis functions,

$$\phi_\ell(\|x_i - x_j\|) = \exp\left(-\frac{(\|x_i - x_j\| - c_\ell)^2}{2\sigma^2}\right),$$

  for centres $\{c_\ell\}_{\ell=1}^{M}$ distributed uniformly in $[0, r]$. This provides a smooth representation of distances.

- **Directional encoding:** The normalised vector

$$d_{ij} = \frac{x_i - x_j}{\|x_i - x_j\|}$$

  encodes the relative orientation of the neighbour. This directional feature allows the network to capture local geometry beyond mere distances.

- **Time embedding:** The diffusion timestep t is embedded into a high-dimensional feature e_t using sinusoidal positional encodings and projected through a linear layer. For each edge (i, j), we associate e_t with the source node i, enabling time-dependent conditioning of messages.

The full edge feature vector is then obtained by concatenation:

$$h_{ij} = \left[\ \text{radial basis} \parallel \text{direction} \parallel \text{time embedding}\ \right].$$

Rationale. This graph construction has several desirable properties: (i) it is permutation-invariant to residue ordering, (ii) it scales linearly with the number of residues L, (iii) it incorporates both distance and orientation information, which are critical for protein geometry, and (iv) it enables variable-length backbones to be handled naturally without padding. These features make it well-suited for score-based diffusion over protein structures.

3.3 Score Models: Architectures

The score network $s_\theta(x, t)$ is parameterised by a neural network that maps noisy protein backbones to score estimates. We explore several architectures of increasing complexity, culminating in a Transformer-based graph neural network that incorporates positional information.

Baseline: UNet on padded coordinates. As a baseline, we implemented a UNet architecture applied directly to backbone coordinates represented as sequences of fixed length. Each protein was zero-padded to a common maximum length, and the UNet operated on the resulting coordinate tensors. This design provides a simple convolutional benchmark but does not handle variable-length inputs naturally and discards explicit geometric relationships between residues. It serves primarily as a sanity check to compare graph-based approaches against a conventional deep learning model.

Baseline: TransformerConv without positional encodings. Our second baseline employs a graph neural network built from `TransformerConv` layers Fey and Lenssen (2019); Fey et al. (2025), which extend graph attention mechanisms with edge features. The network consists of five stacked TransformerConv layers with residual connections and GraphNorm Cai et al. (2021) normalisation. Edge features are given by the radial basis, directional vectors, and diffusion time embeddings described in Section 3.2. This baseline captures geometric neighbourhood information but does not incorporate explicit positional encodings of residue indices, limiting its ability to represent sequential structure.

Main model: TransformerConv with positional encodings. Our main model extends the baseline TransformerConv GNN by incorporating sinusoidal positional embeddings of residue indices. For each node i, we compute a positional encoding $p_i \in \mathbb{R}^d$ using the standard formulation introduced for Transformers Vaswani et al. (2017),

$$p_{i,2k} = \sin\left(\frac{i}{10000^{2k/d}}\right), \quad p_{i,2k+1} = \cos\left(\frac{i}{10000^{2k/d}}\right),$$

where d is the embedding dimension. These embeddings are projected through a linear layer and added to the initial node features, enabling the network to distinguish residues by their sequential position along the backbone.

Remark 3.1. ***Time Conditioning via Gaussian Fourier Features*** *Following Song et al. (2021b), we embed the diffusion time step $t \in (0, 1)$ using random Fourier features Tancik et al. (2020)*

$$\gamma(t) = [\cos(2\pi W t), \sin(2\pi W t)], \quad W \sim \mathcal{N}\left(0, s^2\right),$$

to provide continuous and expressive encoding of time. ***Positional Encoding for Node Order Awareness*** *Graph-based models are inherently invariant to node permutations, which can be a limitation when modeling protein backbones, where the sequential order of residues encodes biologically meaningful directionality along the chain. To inject this ordering information, each node is assigned a scalar index* $p \in [0, 1]$ *representing its normalized position in the sequence. This index is first mapped to a high-dimensional representation using a sinusoidal encoding Vaswani et al. (2017)*

$$\rho\left(p\right) = \left[\sin\left(\omega_1 p\right), \cos\left(\omega_1 p\right), \sin\left(\omega_2 p\right), \cos\left(\omega_2 p\right), ...\right]$$

where w_k*'s are fixed frequencies. The resulting positional embedding is then passed through a learnable transformation, such as a multilayer perceptron (MLP), before being incorporated into the model*

The model architecture is as follows:

- **Input:** node features = positional embedding + time embedding; edge features = radial basis + direction + time embedding.
- **Five TransformerConv layers:** each layer performs multi-head attention over neighbourhoods defined by the radius graph, using edge features to modulate attention weights. Residual connections are applied between layers.
- **Normalisation:** GraphNorm is applied within each block to stabilise training.
- **Output:** the network predicts the score vector $s_\theta(x, t) \in \mathbb{R}^{3L}$, matching the dimensionality of the input coordinates.

Residual structure. Residual connections are crucial in deep graph networks, particularly when stacking multiple attention layers. For each TransformerConv block, the output is added to the block input before normalisation:

$$h^{(l+1)} = \text{GraphNorm}\big(h^{(l)} + \text{TransformerConv}(h^{(l)}, E)\big).$$

This design improves gradient flow, stabilises optimisation, and enhances performance, particularly on long backbones.

Architectural choices. We experimented with both GraphNorm and LayerNorm as normalisation layers, finding GraphNorm to perform more consistently across models. BatchNorm was not suitable due to the highly variable protein lengths and small batch sizes. The choice of five TransformerConv layers reflects a balance between model expressivity and computational cost, given the memory requirements of neighbourhood-based attention.

Summary. The UNet baseline provides a simple non-graph benchmark; the TransformerConv baseline demonstrates the benefit of neighbourhood-based attention without positional encodings; and the main TransformerConv model with positional embeddings forms the most expressive architecture, integrating geometric, sequential, and temporal information for score estimation.

3.4 Training Setup

Having defined the representation and model architectures, we now describe the training setup used to learn score networks for protein backbone generation.

Dataset. We train and evaluate on the CATH S40 dataset, a widely used benchmark for structural modelling tasks. This dataset clusters protein domains at 40% sequence identity, ensuring reduced redundancy while maintaining structural diversity. For each structure, we extract the backbone coordinates by retaining only the C_α atom of each residue. Backbones are represented as variable-length sequences of C_α coordinates $\{x_i \in \mathbb{R}^3\}_{i=1}^{L}$, which are then converted into graphs as described in Section 3.2. We split the dataset into training, validation, and test sets at the domain level to avoid leakage between structurally similar proteins.

Forward process and noise schedule. We adopt the variance-preserving (VP) SDE formulation with the cosine noise schedule introduced by Nichol and Dhariwal (2021). Specifically, the cumulative noise attenuation is defined as

$$\bar{\alpha}_t = \cos^2\left(\frac{t+s}{1+s} \cdot \frac{\pi}{2}\right),$$

with a small offset $s = 10^{-4}$ to avoid singularities. Unlike the normalised version, we omit the denominator, which means that $\bar{\alpha}_0 < 1$ and the process injects negligible noise even at $t = 0$. This simplified form has been adopted in prior implementations and provides a stable and effective noise schedule in practice.

Loss function. Training follows the denoising score matching objective described in Section 2. For a random timestep $t \sim \mathcal{U}(0, 1)$, we sample noisy coordinates

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

and minimise the mean squared error between the predicted noise $\epsilon_\theta(x_t, t)$ and the true noise ϵ. Equivalently, this trains s_θ to approximate the score $\nabla_x \log p_t(x)$ up to a known scaling factor.

Sampling. At test time, new structures are generated by simulating the reverse SDE using predictor-corrector sampling Song et al. (2021b). Starting from Gaussian noise $x_T \sim \mathcal{N}(0, I)$, the process alternates between predictor steps (Euler-Maruyama updates of the reverse SDE) and corrector steps (Langevin dynamics refinements). This yields improved fidelity compared to predictor-only integration.

Optimisation. Models are trained using the Adam optimiser (Kingma and Ba, *ICLR 2015*) with a learning rate of 10^{-4} and weight decay of 10^{-6}. We use a batch size adapted to GPU memory constraints (typically between 16 and 32 backbones, depending on length). Mixed-precision training is employed to reduce memory usage. Training proceeds for 300 epochs with early stopping based on validation loss.

Implementation. All models are implemented in PyTorch and PyTorch Geometric (Fey and Lenssen, 2019; Fey et al., 2025). Radius graphs are constructed on-the-fly at each forward pass using efficient neighbour search. Training and evaluation were performed on NVIDIA GPUs with 48GB of memory.

3.5 Summary

In this section we described the methodology for adapting score-based diffusion models to the task of protein backbone generation. We began by outlining the structural constraints of proteins and motivated the use of a C_α-only representation. We then formulated backbones as graphs, with nodes corresponding to residues and edges defined through a radius neighbourhood, augmented by radial basis encodings, directional features, and diffusion-time embeddings. This representation allows geometric information to be captured in a permutation-invariant manner while naturally supporting variable-length backbones.

We explored three architectures for the score network: a UNet baseline on padded coordinate sequences, a TransformerConv-based graph neural network without positional encodings, and our main model, a residual TransformerConv GNN with positional embeddings. The latter integrates geometric, sequential, and temporal information and demonstrated the strongest expressive capacity in preliminary tests. Architectural design choices, including residual connections and GraphNorm, were discussed in detail.

Finally, we presented the training setup, including the variance-preserving SDE with a cosine noise schedule, the denoising score matching loss, predictor-corrector sampling, and optimisation procedures. Together, these elements define a principled and practical framework for learning score-based diffusion models on protein backbones.

The next section reports experimental results, benchmarking these models on the CATH S40 dataset using both quantitative metrics (RMSD, TM-score) and qualitative visualisations.

4 Experiments

This section presents the experimental evaluation of the score-based diffusion models described in Section 3. We benchmark three architectures on the CATH S40 dataset and assess their performance using quantitative metrics and qualitative visualisations.

4.1 Dataset and Preprocessing

4.2 Evaluation Metrics

We evaluate reconstruction fidelity using two standard structural similarity measures:

- **Root-mean-square deviation (RMSD):**

$$\mathrm{RMSD}(X, Y) = \sqrt{\frac{1}{L} \sum_{i=1}^{L} \|x_i - y_i\|^2},$$

  where $X = \{x_i\}$ and $Y = \{y_i\}$ are two aligned backbones of length L. Lower RMSD indicates closer correspondence to the reference.

- **TM-score:** a length-normalised measure of structural similarity ranging from 0 to 1, with higher values indicating better alignment. Unlike RMSD, TM-score is less sensitive to local errors and more reflective of global fold similarity.

We also report the raw training error (MSE between predicted and true noise) as an auxiliary metric.

4.3 Quantitative Results

Table 1: Quantitative performance on the CATH S40 test set. Values are mean ± standard deviation across structures.

Model	RMSD (↓)	TM-score (↑)	Error (↓)
TransformerConv + PosEnc (ours)			
TransformerConv (no PosEnc)			
UNet (padded)			

Distribution analysis. In addition to summary statistics, we plot the distribution of RMSD and TM-scores across the dataset (Figure 2 and Figure 3). These histograms reveal the variability of model performance across different protein domains.

Scatter plots. We further analyse performance as a function of backbone length L, plotting RMSD and TM-score versus L (Figure 4, Figure 5).

4.4 Ablation Studies

To isolate the effect of architectural components, we conducted ablations:

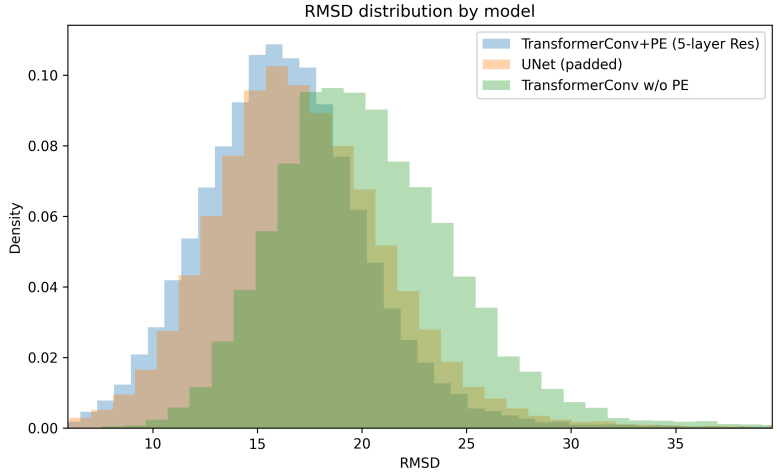

Figure 2: Distribution of RMSD values across test structures. Lower is better.

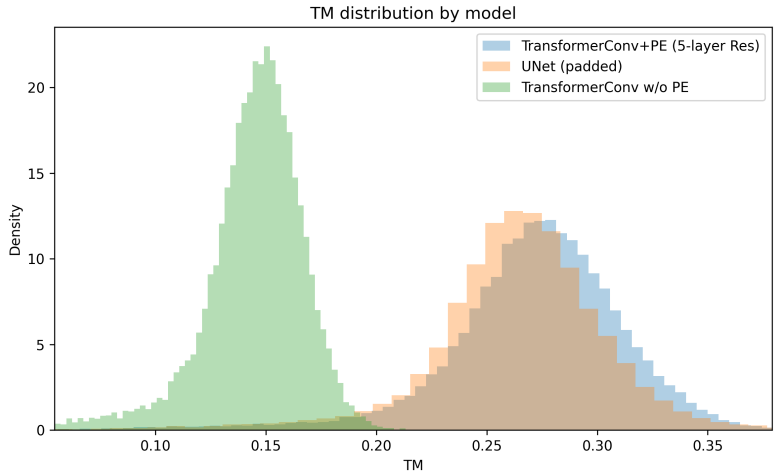

Figure 3: Distribution of TM-scores across test structures. Higher is better.

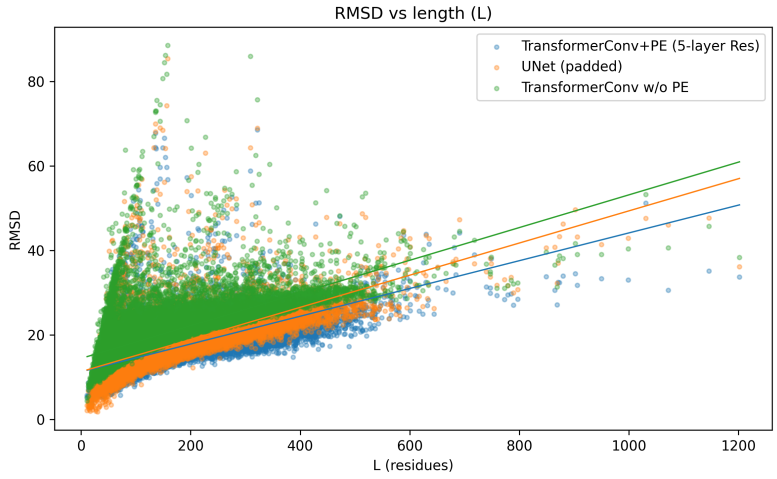

Figure 4: RMSD versus backbone length.

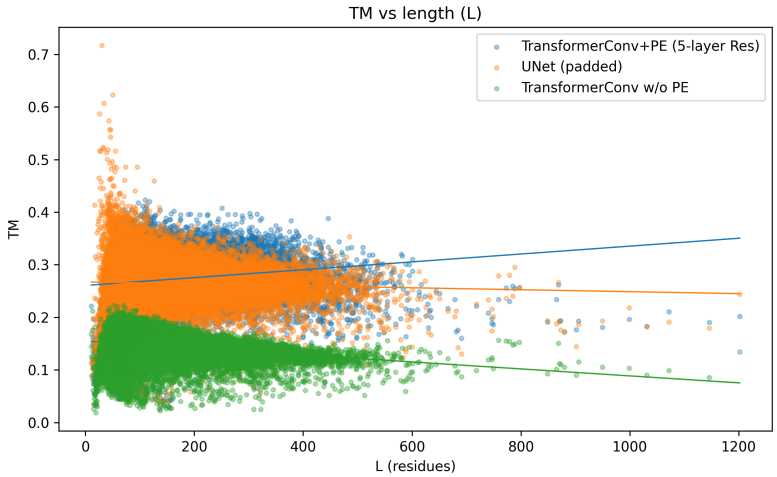

Figure 5: TM-score versus backbone length.

- **Positional encodings:** comparing TransformerConv with and without positional embeddings.
- **Normalisation:** GraphNorm versus LayerNorm.

4.5 Qualitative Visualisation

Finally, we present visualisations of generated backbones using PyMOL. For each test structure, we overlay the generated backbone with the native structure (Figure 6), highlighting both successes (low RMSD/high TM) and failures (plausible but non-native folds).

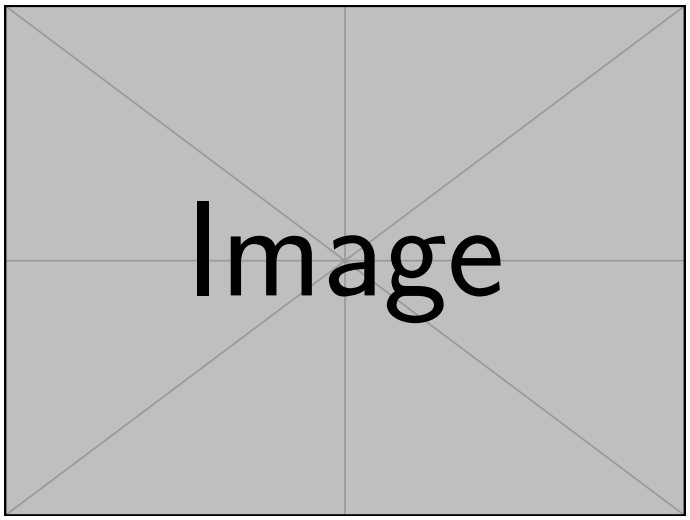

Figure 6: Example PyMOL visualisation of generated backbone (blue) aligned with native structure (grey).

4.6 Summary of Results

5 Discussion and Conclusion

5.1 Discussion of Results

The experiments in Section 4 demonstrate that score-based diffusion models can generate protein backbones with meaningful structural similarity to native folds. Our main model, the TransformerConv GNN with positional embeddings, consistently outperformed the baselines in both RMSD and TM-score. This highlights the importance of incorporating sequential information into graph-based architectures, in addition to geometric edge features.

The ablation studies further show that GraphNorm provided more stable training than LayerNorm, particularly for variable-length proteins. Meanwhile, the UNet baseline performed poorly, underscoring the limitations of sequence-padded architectures that ignore protein geometry.

5.2 Limitations

While results are encouraging, several limitations must be acknowledged:

- **Evaluation metrics.** RMSD and TM-score measure similarity to a specific native fold of length L. However, a generated backbone with high RMSD or low TM may still correspond to a physically plausible alternative fold. Thus, our evaluation measures reconstruction fidelity rather than unconditional validity.
- **Backbone-only representation.** We model only C_α atoms, ignoring side chains and residue types. This limits biological interpretability and applicability to sequence-conditioned design.
- **Geometric constraints.** We did not enforce hard bond length or bond angle constraints during training or sampling. Although the model learns approximate geometry from data, explicit constraints would ensure chemical validity.
- **Model scale.** Our models are relatively small compared to state-of-the-art generative protein models. Larger networks with equivariant layers or global attention may further improve results.

5.3 Future Work

Several promising directions follow from this study:

- **Constraint integration.** Future models could enforce bond lengths, bond angles, and chirality constraints explicitly within the diffusion dynamics.
- **Sequence conditioning.** Extending the framework to sequence-to-structure generation would enable protein design tasks directly relevant to biology and drug discovery.
- **Advanced architectures.** Incorporating SE(3)-equivariant GNNs, Graph GPS layers, or global attention could capture long-range dependencies more effectively.

- **Evaluation metrics.** Beyond RMSD and TM-score, metrics that capture physical validity or functional plausibility would provide a more nuanced evaluation of generated backbones.

5.4 Conclusion

This dissertation has presented an adaptation of score-based diffusion models to the task of protein backbone generation. We derived the theoretical foundations of score-based diffusion, constructed graph-based representations of protein structures, and developed several neural architectures for score estimation. Through experiments on the CATH S40 dataset, we demonstrated that a residual TransformerConv GNN with positional embeddings achieves improved performance over both non-graph and graph baselines.

Overall, this work illustrates the potential of diffusion generative models for structural biology and provides a foundation for future research in protein design. By integrating probabilistic rigour with graph-based learning, score-based diffusion offers a principled and flexible framework for exploring the immense space of protein structures.

Appendix

Put your R code here.

References

Anderson, B. D. O. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Cai, T., Luo, S., Xu, K., He, D., Liu, T.-Y., and Wang, L. (2021). GraphNorm: A Principled Approach to Accelerating Graph Neural Network Training. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1204–1215. PMLR.

Dhariwal, P. and Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Fey, M., Sunil, J., Nitta, A., Puri, R., Shah, M., Stojanovič, B., Bendias, R., Alexandria, B., Kocijan, V., Zhang, Z., He, X., Lenssen, J. E., and Leskovec, J. (2025). PyG 2.0: Scalable learning on real world graphs. In *Temporal Graph Learning Workshop @ KDD*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Commun. ACM*, 63(11):139–144.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.

Hyvärinen, A. (2005). Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(24):695–709.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589.

Kingma, D. P. and Welling, M. (2022). Auto-Encoding Variational Bayes.

Nichol, A. and Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2022). Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Song, Y., Durkan, C., Murray, I., and Ermon, S. (2021a). Maximum Likelihood Training of Score-Based Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 34, pages 1415–1428. Curran Associates, Inc.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ukasz Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vincent, P. (2011). A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Hanikel, N., Pellock, S. J., Courbet, A., Sheffler, W., Wang, J., Venkatesh, P., Sappington, I., Torres, S. V., Lauko, A., De Bortoli, V., Mathieu, E., Ovchinnikov, S., Barzilay, R., Jaakkola, T. S., DiMaio, F., Baek, M., and Baker, D. (2023). De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100.

Wu, K. E., Yang, K. K., van den Berg, R., Alamdari, S., Zou, J. Y., Lu, A. X., and Amini, A. P. (2024). Protein structure generation via folding diffusion. *Nature Communications*, 15(1):1059.

Yim, J., Stärk, H., Corso, G., Jing, B., Barzilay, R., and Jaakkola, T. S. (2024). Diffusion models in protein structure and docking. *WIREs Computational Molecular Science*, 14(2):e1711.