Diffusion Based Human Motion Generation

Reilly John Oldham

***Abstract*—Efficiently generating realistic human motion presents a significant challenge across various domains, including animation and robotics. Traditional handcrafted motion sequences for animation are notoriously time-intensive and skill-demanding. On the other hand, motion capture technology, while being very effective for real-word data, often incurs high costs for the equipment and may produce noisy data requiring further work.**

This project focuses on the development of autoregressive conditional diffusion models tailored to human motion generation. We conduct a comprehensive examination of existing state-of-the-art motion models that utilize Diffusion and Normalising Flows while acknowledging other generative models. We identify limitations and opportunities for enhancement in these models and propose generalisable solutions. Our research contributes to the ongoing evolution of generative diffusion techniques, particularly in the area of autoregressive generative models.

Furthermore, we provide an additional tangible demonstration of autoregressive diffusion using a toy model showed in an intuitive way that does not require animated sequences. This will further illustrate the model's potential for time-series tasks and its ability to be applied to other domains while producing convincing results. By thoroughly evaluating our model and its capabilities, we aim to provide a valuable contribution to the field with explorations into important hyperparameters and model architectures. This work underscores the importance of understanding and addressing challenges in predictive time-series tasks, thereby advancing our collective knowledge in this area.

***Index Terms*—Diffusion, Normalising Flows, Human Motion, Score-based models, Multivariate, Time-series**

I. Introduction

A Significant problem in the world of game development, animation, robotics, and other related fields is the realistic generation of human like movement. Creating lifelike human motion through manual animation requires a significant amount of skill, time, and thought on how to create the movements [1]. Because of this, it could be beneficial for many creatives to be able to automatically, or semi-automatically create lifelike human motion with little to no effort. Existing methods for generation of human motion such as mocap suits are quite useful, but these suits are very expensive, require a human to act out the motion, and the data collected from the suits can be noisy which necessitates post capture clean-up of the data [1].

Currently there are models that exist that can create these sequences such as MoGlow [2]. However, as discussed in the MoDiff paper [3] and will be further discussed here, there are associated issues and drawbacks with current models. Therefore, to address these challenges we have developed a novel model-based approach through the use of an autoregressive diffusion model. This report will show that this is a viable solution with room for further refinement and improvement.

This project was supervised by Bastiaan Kleijn

A. Model Choice

Diffusion in recent years has come to the forefront of generative model research as it has many advantages over General Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Normalising Flows which were previously the main focus of research [4]. Diffusion transforms a given noise distribution into the desired data distribution such as an image, audio, or in the context of this project, a human motion sequence.

By iteratively transforming the noise distribution to the target distribution these models can more accurately estimate the data likelihood and produce better samples.

Comparatively, VAEs use the reconstruction loss, commonly Mean Squared Error (MSE), which can result in a failure to capture complex distributions resulting in poor samples. This is due to the tendency for MSE to capture the average or midpoint of the distributions, whereas a likelihood model would separate the distributions. Furthermore, VAEs can suffer from posterior collapse where the latent variables become independent of the generated samples thus being unable to capture the full complexity of the data [4].

GANs on the other hand can create high-quality samples equal to or better than diffusion models, however they can suffer from training instability where the critic and generator no longer work together causing mode collapse [4]. This means the creation of a good model will require fine tuning of the architecture and parameters.

The reliability of diffusion models in their ability to accurately learn and replicate the target distribution without significant issues has led to its current focus in research. With state-of-the-art performance, in different modalities diffusion would be the ideal model for human motion exploration.

Our model's design is influenced by prior research, particularly MoGlow, which demonstrates that using an autoregressive component is effective in generating realistic samples. Our contribution extends this work by integrating a similar architecture into the realm of diffusion, presenting a novel approach to this task.

B. Project Outcomes

This project has effectively addressed the issue of time-series generative modeling related to recalling prior actions. This was achieved by introducing an autoregressive component within the diffusion model. Employing an LSTM network as this autoregressive element, the model becomes capable of incorporating past actions as context for shaping future predictions. Consequently, the model can produce human motion sequences that are both realistic and coherent, as will be shown in the evaluation section.

The problem statement of generating sculptable motion sequences is addressed by the incorporation of path conditioning to the model in the form of Δx, Δy, and Δr tuples. These tuples specify the translation and rotation of the model at each frame, which the model has learned and can accurately predict future sequences conditioned on these. This allows anyone using this model the ability to specify a path to follow, and a realistic walking or running sequence will be generated.

To assess this project, we utilize a combination of quantitative and qualitative criteria. These criteria aim to demonstrate the model's alignment with previously unseen ground truth data. Through a user survey rating the realism of the sequences we found that the model achieved a rating of 3.71 out of 5 whereas the ground truth received a score of 3.95 Table I. This shows that subjectively the model performs very closely to the ground truth data. We have also pinpointed areas for model enhancement, which we will subsequently address. Additionally, we've developed a simplified model to illustrate the model's autoregressive capabilities without the use of animated sequences, thereby highlighting its potential applicability to various continuous domain time-series tasks.

C. Sustainability Goals

The main focus of this project is to create a predictive model for human motion generation, which has significant applications in game development, animations, and robotics. It's important to note that while environmental and sustainability concerns are a top priority, this project doesn't directly impact those issues.

However, if this model or a similar one were to be deployed in the real world, especially in gaming for real-time motion generation, the cumulative impact of numerous users could be substantial. This underscores the necessity to consider the model's efficiency in both training and inference. At present, real-time inference for this model is not possible, but through code optimisations and potential techniques like InstaFlow by Liu et al. [5], a diffusion technique which only requires one denoising iteration, we can further enhance the model's efficiency.

One notable improvement in this project is the speed of the training process, which is much faster than the original base model. This is achieved by using a different diffusive solver which will be discussed later. This enhancement has streamlined the training process, making it more efficient thus following Green Coding principles decreasing the environmental impact though the use of less computational resources.

D. Tools and Methodologies

Implementation of a generative model such as diffusion from scratch would likely have taken too long in a time-limited project like this. To this end we have employed multiple tools and methodologies that have helped reduced the time taken to produce this model.

The core of our model utilised TimeGrad's implementation [6] which is built in Python on top of the GluonTS library, a well-established library for probabilistic forecasting. Working with Python for this project was a great choice due to its familiarity, flexibility, and library support. GluonTS overall was a good library but the original TimeGrad implementation was broken and lack of documentation of GluonTS which contributed to stalled development at some stages of the proejct. Nonetheless in the end these libraries sped up the development of this project and are the tools that contributed the most to the success of this project.

Additionally, HuggingFace Diffusers [7] was extremely helpful as a resource both for papers and implementations for alternate diffusion solvers. This enables our design to implement any solver from this library which again greatly cut down on work an enabled us to address the core problem of the project directly.

For efficient training we were able to make use of the compute servers provided by ECS. Our model was trained and developed on the Gryphon server using an RTX A6000 for CUDA acceleration. This allowed us to have fast and easy access to compute resources facilitating this project heavily as it gave us the ability to quickly iterate over different ideas and solutions to problems we encountered.

The usage of Git and GitLab was not hugely significant but they still played a valuable role in this project. These tools allowed for version control enabling work to be done concurrently on both the model implementation and other sections such as visualization. This enabled our projects iterative methodology by allowing us to address different components simultaneously and transition to iterative development when required.

II. Background Theory

Prior to delving into the literature review, we will explore the essential foundational elements necessary for comprehending the subject matter. Through an examination of key equations, we aim to establish a mathematically grounded understanding that will facilitate the linkage between the "laymans" explanation and the practical implementation of these methodologies. This section will consist of explanations of Normalising Flows and Diffusion for generative modelling, followed by a section on Autoregression consisting of the different autoregressive solutions.

A. Normalising Flows

Normalising flows in simple terms is a method in which we can take data from a known distribution, such as a Gaussian, and transform it to a more complex target distribution like a human motion sequence. This transformation is accomplished by a series of transformations in which there is an associated inverse transformation. Training the functions to turn the target distribution into noise allows us to use the inverse of those functions to transform the noise distribution to the target distribution [8].

Formally, given a latent distribution Z we can transform this with a non-linear function f, which has a well-defined inverse and a one-to-one mapping, to a more complex distribution X [9]. This transformation is conducted by chaining multiple simpler functions to transform the latent distribution into the target distribution. This sequence is described in equations (1) and (2).

$$x = f_\theta(z) = f_n \circ f_{n-1} \cdots \circ f_1(z) \tag{1}$$

$$z = f_\theta(x)^{-1} = f_1^{-1} \circ \ldots f_{n-1}^{-1} \circ f_n(x)^{-1} \tag{2}$$

This method exploits the change of variables (3) formula which describes that we map x to the density of z under $p(z)$ multiplied by some scalar magnitude which is the inverse of the absolute magnitude of the Jacobian matrix [9]. Given we take the inverse poses one key issue, which is that we cannot invert a matrix which is not square, meaning our latent samples must be the same dimensionality as the target samples. This reduces interpretability of the model as our latent sample dimensionality is the same as our target dimensionality, though, as we will find with diffusion this is the same. However, this also does encourage diversity allowing for a wider range of samples compared to a VAE.

$$p_x(x) = p_z(z) \left| \det\left(\frac{\partial f^{-1}(z)}{\partial z}\right) \right|^{-1} \tag{3}$$

For the training of these functions we can, take the log of both sides (4) and this will give us the exact log-likelihood evaluation in a tractable way [9]. Given this we can optimise the model parameters with respect to the maximum log-likelihood. Additionally, this method allows us exact posterior inference of z through the inverse mapping on x.

$$\log p_x(x) = \log p_z(z) + \sum_{i=1}^{N} \log \left| \det\left(\frac{\partial f_i^{-1}(z)}{\partial z_i}\right) \right|^{-1} \tag{4}$$

B. *Diffusion*

Diffusion or Denoising Diffusion Probabilistic Models (DDPMs), similar to Normalising Flows, aims to transform a known distribution into a more complex target distribution. However, the means in which diffusion approaches this task is significantly different. The central concept revolves around systematically and gradually destroying the structure of the data through an iterative forward process, then a backwards process learns to reverse the noise iteratively to reconstruct the data's original structure. This iterative process results in a model that is both flexible and tractable.

There are two main theoretical frameworks for solving diffusion; diffusion probabilistic models and denoising score matching models. Our solution uses the Ordinary Differential Equation (ODE) approach which falls under the denoising score matching models. To give an intuitive explanation of both diffusion and the ODE approach we will start from the Markovian interpretation and show how to get to the ODE solution.

1) Markovian Interpretation: The Markovian interpretation of diffusion typically involves the use of two Markov chains. Markov chains are mathematical system for transitioning between states according to a probabilistic rule, in this case a transition kernel. One chain is for the forward process which is responsible for destroying the data's structure. The other is for the reverse chain which reconstructs the noisy and distorted data into a more structured form [10].

Formally, for the forward process we can find the joint distribution of all random variables from x_1 to x_T given the original sample x_0 (5) [10]. Through the iterative application of the transition kernel (6) we gradually inject noise into the sample destroying the structure of the data. The β from the transition kernel (6) describes the noise schedule which is reduced after each step.

$$q(x_1, \ldots, x_T | x_0) = \prod_{t=1}^{T} q(x_t | x_{t-1}) \tag{5}$$

$$q(x_t, | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I) \tag{6}$$

From this forward process we can derive an approximate inverse of the forward process with a reverse process that is also in the form of a Markov chain [10]. This is defined as (7) which takes the approximately Gaussian sample x_T and predicts the joint distribution of x_0 to x_T. This again has an associated transition kernel defined as (8) which is usually defined as a neural network with parameters θ to predict the μ and Σ to reverse the noise [10]. Though for the covariance matrix in practice this is set to a specific value such as the identity matrix I so we only need to predict μ.

$$q(x_0, \ldots, x_T) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} | x_t) \tag{7}$$

$$q(x_t, | x_{t-1}) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{8}$$

In the training process of Markov chain-based diffusion models, the primary objective is to optimize the Evidence Lower Bound (ELBO) [10]. The ELBO serves as the objective function for learning the model's parameters, specifically the transition kernel. By maximising the ELBO, the model learns to generate data that closely resembles the target distribution. Computing this however is very cumbersome and quite inefficient which is why score-based generative models are the preferred solution. Score-based models are both more tractable and require less diffusion steps resulting in an efficient computation of the diffusion problem.

2) Stochastic Differential Equations: The Stochastic Differential Equation (SDE) solution to diffusion can be interpreted and derived as a continuous case of the Markov implementation, though with some key changes and insights. Reducing the size of the time steps for the beta schedule so that it is effectively a continuous function of the time-step t we can fine the forward formula as follows (9) where we have some functions defined by (10) [11].

With dw being the standard Wiener process of infinitesimally small noise this effectively models a particle's (or "pixel" in the image case) evolution through space as it diffuses from high to low entropy through Brownian motion [11]. The SDE intuitively connects diffusion to the idea that we are solving the diffusive process we know from physics in reverse to arrive at a structured distribution.

$$dx = f(x,t)dt + g(t)dw \tag{9}$$

$$f(x,t) = -\frac{1}{2}\sqrt{\beta(t)}x \text{ and } g(t) = \sqrt{\beta(t)} \tag{10}$$

We can therefore derive the reverse time SDE as (11) which includes the score function $\nabla_x \log(p(x,t))$ [11]. This score function is the central concept of both the SDE and ODE approach.

$$dx = \big(f(x,t) - g^2(t)\nabla_x \log(p(x,t))\big)\, dt + g(t)dw \tag{11}$$

The score function function has an analytical solution which is implemented by a neural network to estimate the score at each time-step. The idea of the score function is to evaluate the log probability of the density function at each time-step which will allow us to converge on the target distribution [11]. Connecting this to Langevan equations through substitutions of $f(x,t)$ to λx and $p(x,t)$ to $p(x) \propto \exp\frac{-V(x)}{kT}$, with $V(x)$ as a potential, we can say that $\log p(x)$ is a potential and the score function $\nabla_x \log(p(x,t))$ is a force [11]. While these are a few new concepts to understand, this effectively says that our score function evaluates the potential of the current time-step and applies a force that moves it towards the target distribution.

3) Ordinary Differential Equations: While SDEs are faster, better, and more general than the Markov approach through a tractable likelihood estimation it would be better if we could reduce the number of diffusion steps further. The main issue here is that because SDEs are a stochastic process we do not always take the direct path to the target distribution, this is what ODEs aim to solve.

To arrive at the backward SDE equation (11) the Fokker-Plank equation was used which describes the time evolution for probability distributions. This was then written in a form that uses the time evolution of the probability distribution to describe the evolution of x. Using this method there is also an ODE solution for x that corresponds to the same Fokker-Plank equation for $p(x,t)$ which is defined as (12) [11].

$$\frac{dx}{dt} = f(x,t) - \frac{1}{2}g^2(t)\nabla_x \log(p(x,t)) \tag{12}$$

This equation removes the nondeterministic Wiener process which allows for a one-to-one mapping of noise to target. Without the nondeterministic component we have a direct mapping with is also invertible by reversing the minus sign in-front of the score function. Similar to Normalising Flows we can now have exact posterior inference of the latent z from our sample x. With this deterministic mapping we can now also take larger diffusion steps without affecting the quality again reducing the number of steps requried.

However, while ODEs are very efficient and have a lot of benefits it is worth noting that there are theoretical problems with them. One of which is that SDEs are self-healing whereas ODEs are not, which means SDEs are less sensitive to random fluctuations. Additionally, SDEs should produce better quality samples due to their stochastic nature adding random noise. Again though, these are only theoretical problems and are not usually encountered in practice.

C. Autoregression

Because this is a time-series task it is a given that we have an autoregressive component to allow past actions to influence future predictions. There are a number of autoregressive techniques which will be discussed here to give context to the choices made for this project and to allow a deeper understanding of the literature review. This section will consist of the basic Recurrent Neural Network (RNN) followed by Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Transformers.

1) Recurrent Neural Network: The RNN is the most basic autoregressive model which takes in the previous hidden state and the current input to create a new hidden state (13). This hidden state is kept track inside the neural network to be used for future predictions [12]. For the output representation we again run the hidden state through a neural network (14).

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{13}$$

$$y_t = W_{hy}h_y + b_y \tag{14}$$

This method is very simple, and because of that suffers in a few ways. The memory of past actions decays uniformly as we iteratively update the hidden state without any additional modifiers [12]. This means that for longer term time dependencies RNNs suffer. For our solution this would not be ideal as there are likely long term temporal relationships in human motion such as running speed. Additionally, this method can suffer from vanishing/exploding gradients due to the weight term in the output resulting in a model that might not converge and remain sub-optimal [12].

2) Long Short-Term Memory: LSTMs are more complex and address the issues of the uniform information decay by introducing a forgetting factor f_t and a new-contribution factor i_t. This allows importance to be placed on new information and control the rate at which this information is decayed. The calculation for the current contribution C_t, analogous to the hidden state from RNNs, is as follows (15) [12].

$$C_t = f_t C_{t-1} + i_t \bar{C}_t \tag{15}$$

Each of these newly introduced elements describing the decay factor and new contribution factor are modelled by neural networks. This allows for a dynamic value for each new input allowing LSTMs to be much more expressive in its output representation.

For the output we have h_t which is the same as the y_t output from the RNN and should not be confused by the hidden state. There is another factor introduced in the calculation (16) which is the factor of the state to the output which is similar to the RNN calculation but without an explicit weight term [12].

$$h_t = o_t \tanh(C_t) \tag{16}$$

LSTMs therefore do not encounter the problem of exploding or vanishing gradients because their output lacks the explicit weight term. This characteristic, along with their effectiveness in capturing long-term temporal dependencies and resilience against gradient issues, renders LSTMs a highly suitable choice for this model. Furthermore, implementing this would be a novel approach for diffusion based human motion generation which we can gain further insights on time-series modelling with.

3) Gated Recurrent Unit: GRUs, like LSTMs, are another simple feed-forward neural network approach to model time-series dependencies. Their approach can be thought of as a simpler LSTM with the use of "gates" and the hidden state is also the output state.

The formula for the output of a GRU is as follows (17) [12]. z_t represents the update gate which controls how much of the previous state should be retained and how much of the current state should contribute. This allows the model to dynamically decide whether to retain old information or incorporate new information. The new contribution factor $\bar{h_t}$ is calculated similar to C_t from the LSTM with one key change. This change is that there is also a reset gate r_t included in the calculation. This reset gate controls which elements from the previous hidden state are allowed to continue to the new contribution.

$$h_t = (1 - z_t)h_{t-1} + z_t\bar{h_t} \tag{17}$$

The use of gates rather than factors is the key difference but in practice neither LSTMs or GRUs are the definitively better memory unit. GRUs do solve the issue of exploding/vanish gradients as well due to the lack of a weight term in the output equation which means that for this project GRUs and LSTMs are on level ground. For this project LSTMs were selected with further reasoning to be discussed in the design section.

4) Transformers: Transformers are significantly different from all the of RNN based approaches mentioned previously. Introduced in the paper "Attention Is All You Need" by Vaswani et al. [13]. the transformer network have become the dominant architecture for natural language processing and time-series tasks. For current human motion models with diffusion the transformer models are the basis of their autoregressive abilities.

The main divergence from the RNN approach is the use of self-attention layers which allow the model to effectively "pay attention" to specific parts of the input sequence. Their non-sequential approach to modelling connections between the input stream allows for parallel evaluation.

The self-attention mechanism applies a linear transformation of the input data according to some query, key, and value matrices (Q, K, and V). The equation (18) describes this with a normalising constant d_k which is then passed through a softmax turning this into a probability based representation of how important each input is.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{18}$$

One issue with this approach is that there is no inherent sequence ordering which would result in poor performance for autoregressive tasks. The solution to this is to use positional encoding as part of the attention calculation, from the paper these were calculated as (19) and (20). Additionally, there is another modification for autoregression which includes masking vectors to prevent attention calculations on future vectors.

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{19}$$

$$\text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{20}$$

Overall, transformers produce state-of-the-art results in autoregressive tasks which is why they have been chosen for existing models. Combining these with multiple heads to effectively pay attention to multiple parts of the input sequence (21) allows further flexibility and performance for representing time-series dependencies. Though as we are looking at a novel method the transformer was not chosen for the autoregressive component.

The following sections will continue to discuss the key concepts described here and how they were implemented in specific papers for human motion generation and generative models. For further information and in depth explanations all sources for this section are within the references section.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{21}$$

III. Literature Review

In this section of the report we will delve into the existing literature surrounding human motion generation and diffusive models to give a comprehensive review of the methodologies and techniques involved in this project. Building off the previous tutorial style section we will examine the practical uses of these components for generative modelling of human motion sequences. By examining the existing literature we can motivate the design of our model by addressing weaknesses and find inspiration for evaluation techniques.

A. MoGlow

MoGlow by Henter et al. [2] introduces a method for human motion generation using the Normalising Flows framework extended with an autoregressive component to allow for temporal dependencies on previous frames. This autoregressive component is an LSTM which motivates our model as this paper has proven that you can get state-of-the-art results results without a transformer. Furthermore, this paper provides multiple insights into pitfalls of the training process and provides motivation for our model's design in regards to autoregression.

The overall architecture of the model consists of a Normalising Flows model with the LSTM for autoregression. Previous poses of $x_{t-\tau:t-1}$ are fed in to the LSTM and the output, h_{t-1} of which is concatenated with the control inputs $c_{t-\tau:t}$ [2]. This is then used as the conditioning information for the Normalising Flows model. The size of the LSTM MoGlow used was a two layer 512 node neural network which is what we have used in our model due to the success here.

One of the main insights from this paper was the notion of data dropout. It was found that the MoGlow model had poor adherence to the control signal causing foot sliding artefacts and the model ignoring motion controls [2]. This suggests that the model was relying overly on the autoregressive context. To fix this they applied a data drop on the poses during training. Higher than 50% dropout solved this issue with better results seen at 95% dropout [2]. Keeping this in mind we also included a dropout of 95%.

Choosing a good context window size for predictive models is imperative for good results. While MoGlow has not justified their choice in their context window size τ they have used the value of 10 frames. The motion data this model used to train on was an amalgamation of the HDM05 and CMU databases with a frame rate of 20fps [2]. This results in a context length of 0.5 seconds which as shown by the results of this paper appears to be sufficient. Other motion models we will see also use the same 0.5 second context window so for our model this will also be acceptable.

As one of our goals is sculptable human motion, the ability to condition the model is vital. MoGlow also introduces the Δx, Δy, and Δr tuples which were derived from the recorded motion. Our model will therefore use this data and conditioning information as collecting our own mocap data is out of the scope of this project. This dataset consists of 21 bones with the x, y, and z components resulting in a dimensionality of $\mathbb{R}^{63}$ for the motion sequences and $\mathbb{R}^{3}$ for the conditioning information resulting in a total input size of $\mathbb{R}^{66}$.

The final insights that were gained from this paper are the evaluation metrics which is critical for determining whether a model accurately represents the system we are modelling. To this extent, they employed both quantitative and qualitative metrics for their analysis. For the qualitative metric they conducted a user survey to rate the perceived naturalness of the animation on a scale from 1 to 5. This included animation of their model, ablation models, and ground truth data included to accurately assess the model subjectively. For quantitative metrics they analysed the foot sliding artefacts and bone length analysis to verify the consistency of the frames. Their implementation of these metrics were not described so we will make further comments on these in the evaluation section.

B. *TimeGrad*

TimeGrad by Rasul et al. [6] is the foundation of the produced model. This paper introduces a solution to autoregressive multivariate probabilistic time series forecasting with diffusion. The central concept of this model is to use an GRU with the past predictions and covariate information. The GRUs output is then used as conditioning information for the diffusion process which produces state-of-the-art performance in probabilistic forecasting.

A key component of diffusion models that was not discussed in the background theory was the use of a U-Net. Proposed in the paper “U-Net” Ronneberger et al. [14], U-Nets have become widely used in all areas of generative modelling. In the case of diffusion they are used to estimate the noise to be subtracted at each time-step of the denoising process. The overall architecture is a convolutional neural network (CNN) which contracts the data with an encoder to a bottleneck which is then expanded again using a decoder with skip connections. These skip connections allow the network to combine feature maps from the encoding allowing the model to maintain finer detail and overall context.

The U-Net architecture that TimeGrad uses has 8 residual blocks for the encoder end decoder with skip connections connecting them both [6]. Given this models effectiveness on data with a similar dimensionality to ours, we will opt to keep this architecture as it is a proven model.

For evaluation techniques TimeGrad proposes the Continuous Ranked Probabality Score (CPRS) which is a proper scoring function [6]. Their claim is that this is a better evaluation technique than likelihood for probabilistic forecasting as not all methods they compare against yield analytical forecast distributions or likelihoods are not meaningfully defined. Therefore, this will be used as one of our evaluation techniques and the theory behind it will be discussed further in that section.

One issue with this paper is that it uses the Markov diffusion solver compared to an ODE or SDE solver. This results in a model that takes significantly longer to train and can potentially be less performant. Therefore, as an improvement to this model we will substitute in the Diffusion Exponential Integrator Sampler (DEIS) Multistep Scheduler, a fast ODE solver proposed in the paper ”Fast Sampling of Diffusion Models with Exponential Integrator ” Zhang et al. [15]. Additionally, we will change the model to use an LSTM rather than the GRU due to the effectiveness of LSTMs seen in MoGlow.

C. *MotionDiffuse*

MotionDiffuse by Zhang et al. [16] was the first paper to our knowledge to use diffusion to attempt the problem of human motion generation. One distinct difference from MoGlow, and consequently our model, is the alternative approach of textual input for the conditioning rather than the Δx, Δy, and Δr tuples. This alternative approach allows for finer control of the specific walking style and actions of the generated figure, albeit at the cost of losing control of the exact path it should follow.

For incorporating textual conditioning and the autoregressive context they propose a Cross-Modality Linear Transformer. There are several components of this transformer which include the Text Encoder, Linear Self-attenuation with Cross-attenuation, and a Body Part-independent Controller. The Linear Self-attenuation block utilises a global context rather than a fixed context window to provide semantic meaning over time [16]. This allows for the model to follow the text prompts over an extended period of time. For the Text Encoder they used the pre-trained weights of OpenAI’s CLIP model [17] where the first 7 layers were frozen and the remaining were allowed to be trained. The output of this Cross-Modality

Linear Transformer is then used to condition the diffusion process resulting dynamic length part aware time-series human motion sequences.

This approach does produce convincing results, however, as observed in the accompanying demo video for this paper longer sequences have visible hitching artefacts [16]. This is possibly due to global context causing issues when producing the block based sequences. A better implementation of this would likely solve this issue such an explicit autoregressive component which our model implements.

MotionDiffuse cannot directly be compared to either MoGlow or our model due to the difference in conditioning information. However, this still serves as an establishing paper for diffusion based human motion and provides insights into alternate ways to condition motion models.

D. MoDiff

MoDiff by Yin et al. [3] is a transformer based diffusion model for human motion generation from same institute as MoGlow, though with different authors. This paper does mention TimeGrad as an alternative approach and they developed their own model from scratch for this. However, this paper has not released its code publicly or created a paper on the TimeGrad implementation so any implementation details have not been revealed. Instead this project will show the capabilities of a TimeGrad based approach.

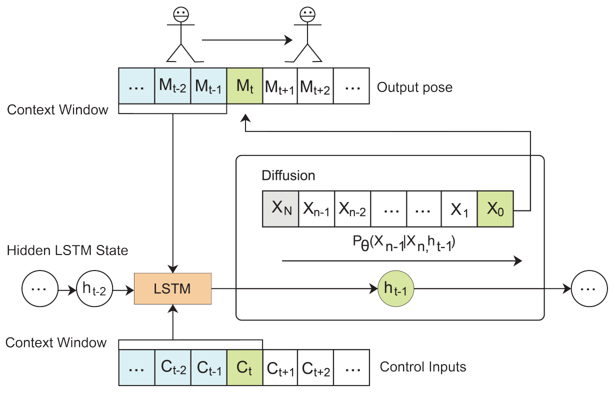
As previously mentioned, MoDiff uses a transformer and similar to MotionDiffuse they use a Cross-Modal Transformer with independent encoders for both the motion context and the control context. For the context window they use the same 10 frame window as MoGlow with the same data making our model and this model directly comparable. The encodings created by the Cross-Modal Transformer are then used as conditioning alongside the step embedding for the diffsion process [3].

Another change compared to MoGlow is a linear scheduler for the data dropout compared to a fixed rate. Other hyperparameters that were insightful were the number of diffusive steps which was set to 100, the same as MotionDiffuse.

The evaluation of the MoDiff paper closely resembles the MoGlow paper. Under this analysis both MoDiff and their TimeGrad implementation end up performing better than MoGlow in both quantitative and qualitative tests. This paper also claims to be better than TimeGrad in the temporal domain for longer sequences due to its transformer based autoregressive architecture. However, since no design or analysis for their custom implementation of TimeGrad has been documented it is not possible to confirm their results.

IV. Design

This section will discuss the general design of our model with references to how the different components connect to represent the system. Choices made for our model will be justified with references to the existing research that we have discussed previously. Specifics on model parameters and data will in the following implementation section.



Output pose
Context Window
Diffusion
Hidden LSTM State
LSTM
Context Window
Control Inputs
M_{t-2} M_{t-1} M_t M_{t+1} M_{t+2}
X_N X_{n-1} X_{n-2} X_1 X_0
$P_\theta(X_{n-1}|X_n, h_{t-1})$
h_{t-2} h_{t-1}
C_{t-2} C_{t-1} C_t C_{t+1} C_{t+2}

Fig. 1: Model Architecture

1) The Model: The architecture of our model can be described visually through Fig 1 which incorporates the flow of previous sequence information in to the LSTM to then be fed as conditioning information in the diffusion process.

Motivating the design of this model we take some inspiration from the MoGlow paper with regards to our selection of the autoregressive network. While a GRU would likely have given similar autoregressive performance we opted not to use this and use the proven approach of an LSTM. In our model implementation we have allowed for the autoregressive component to be a LSTM or GRU without additional work.

For the autoregressive conditioning we input both the previous sequences and their associated conditioning information with the current time-step's conditioning information. This reduces the representation to one hidden state rather than including the previous conditioning values as direct input to the model. This could theoretically allow for dynamic context window length which MoGlow is unable to do due to its fixed size window input.

Another key change from the TimeGrad model was the use of an alternative diffusion solver in the form of the DEIS Multistep Scheduler [15]. This is a fast ODE solver which claims to produce realistic samples in only 10 diffusive steps. Allowing for fast training and inference times would be greatly advantageous, especially in a time-limited project such as this. However, the quality of samples provided by this method evaluated using the Fréchet inception distance (FID), a metric for evaluating images, on the CIFAR10 dataset is is 3.37 [15] which is lower compared to some SDE solvers such as the VE SDE solver [11] with a score of 2.20. For a less time-dependant project it would be better to use a better performing solver. In our implementation we have not tightly coupled the solver with our implementation, this allows for solver substitutions opening this area for further work.

This faster diffusion solver also contributes to our sustainability goals by reducing the training and inference time for the model. Previously mentioned in the sustainability goals section, options to further improve this would be to use methods such as InstaFlow [5] for one shot inference. Issues with this approach similar to our fast ODE solver still include metrics showing worse performance. On a different dataset InstaFlow achieved an FID of 13.9 which shows further research improvements need to be made in this domain before it would be a viable substitution.

Our model design makes no assumptions on the data it will be trained on allowing it to be a generalisable model for all time-series tasks and not just for human motion generation. The next section will discuss the implementation details of our model at length and the simplicity of translating it to different domains.

V. Implementation

This section will describe the implementation of both the full motion model and the toy model for an additional comparison. This will include key discussion on the data and it's representation, model parameters, and finally how our motion inference works.

1) The Data: Our data for the motion synthesis comes from the MoGlow paper [2] which utilises a pooled dataset from the Edinburgh Locomotion MOCAP Database, CMU Motion Capture Database, and HDM05 datasets. With additional preprocessing implemented by MoGlow we have created additional Δx, Δy, and Δr tuples for conditioning of the model. With 21 bone represented by x, y, z coordinates we arrive at a 63 dimensional vector per time-step.

Our training dataset therefore consists of 13710 unique clips from our dataset and with each clip being 4 seconds at 20fps. Additionally, our test set consists of 31 unique clips that are 5 seconds in length allowing for validation to be done.

The toy model we have created uses synthetic data based on a triangular distribution. It is a sequence of 32 dimensional vectors representing the PDF of a triangular distribution with a total spread of 5 units. At each time-step a random translation of $\pm$ 2 along the vector is conducted on the triangular distribution. This data incorporates conditioning based on the step, and is able to be visualized efficiently as shown in Fig 2. This makes it an ideal candidate for showing how autoregression works visually while showing the generalisability of our model.

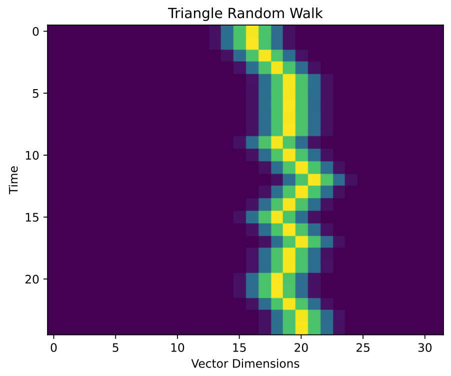


Fig. 2: Toy Model Example Sequence

The intuitive representation of a time-series sequence would be sequential lists of the vectors, in matrix form that is represented by (22).

$$\mathbf{X} = \begin{bmatrix} x_{1_1} & x_{1_2} & \cdots & x_{1_t} \\ x_{2_1} & x_{2_2} & \cdots & x_{2_t} \\ \vdots & \vdots & \ddots & \vdots \\ x_{66_1} & x_{66_2} & \cdots & x_{66_t} \end{bmatrix} \tag{22}$$

However, within GluonTS our data must be transposed before entering the model which would leave our motion data in the form of (23). This key distinction should not be overlooked as without transposition the model will not run and this information is not documented within GluonTS so it is very easy to miss.

$$\mathbf{X}^T = \begin{bmatrix} x_{1_1} & x_{2_1} & \cdots & x_{66_1} \\ x_{1_2} & x_{2_2} & \cdots & x_{66_2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1_t} & x_{2_t} & \cdots & x_{66_t} \end{bmatrix} \tag{23}$$

2) Motion Model: The motion model was trained over 9 hours using an RTX A6000 from the Gryphon compute server. This section will discuss the main hyperparemeters chosen in this model to produce this result and why we have chosen these values. See appendix for Juypter Notebook of the model.

Following the insights gained from the MoGlow paper, we elected to use a dropout rate of 95% to address the problem of autoregressive dominance. Furthermore, we employed the same 10 frame context window for predictions. Additionally, our LSTM neural network use the same architecture of a 2 layer 512 network. It is likely that we can use a smaller neural network but given that MoGlow has found this size to work well we elected to use this as well.

Implementation of the diffusion solver comes from HuggingFace's "diffusers" library [7]. Our implementation accepts one of their solvers as a hyperparameter which will be used to train the model. The use of this library allows our model to work with any diffusion solver HuggingFace have implemented which opens up our model for future investigation of alternative solvers. For this motion model though, we used the DEISMultistepScheduler implementation for training.

The number of diffusive steps we take is set to 150 to ensure a good quality sample is provided given the lower FID results indicated in the paper for the solver [15]. We have not made any comparisons with a smaller number of diffusive steps, but it is again likely to use a smaller value while ensuring high quality samples.

Our selection of batch size was based off prior experience of 64 providing adequate results and it being close to MoGlow's batch size of 100. Through this we worked backwards to find the number of batches per epoch to be 215 by taking our training dataset size // batch size so that we cover all training samples per epoch. Other motion models trained for multiple days, generally around 3, given our time-sensitive project we chose 2500 epochs to reach 9 hours of training.

Due to some generation bugs earlier in development we use a prediction length of 10. Those issues have since been resolved so we could step down to a prediction length of 1 without any differences in predictive power, but due to the time to re-train the model we opted to keep the 10 frame prediction window.

3) Toy Model: Our toy model follows a similar setup to the full motion model where it uses the same diffusive solver and diffusive steps, although it differs in the size of the LSTM and some other parameters which will be discussed. See appendix for Juypter Notebook of the model.

Given the simplicity of the data we have selected a considerably smaller networks size for our LSTM with a 2 layer 16 node neural network. Additionally, we only had a 2 frame context window as we only need to know where the previous distribution was for the next sample. The prediction length for this was set to 8 so we can tell with one generation whether it is following the conditioning value. Finally, the number of epochs used was 200 due to fast convergence. Training this on a consumer GTX 1070 took only 20 minutes which shows this models effectiveness with considerably shorter training times. The original implementation of TimeGrad would take around 1 minute per epoch, which would have made it infeasible for training our motion model in a reasonable amount of time.

4) Motion Inference: Motion inference was conducted on the 31 unique test samples, with the link to these in the appendix. Generation these samples was conducted in a few steps. This can be described by the following algorithm 1.

Algorithm 1 Generate Human Motion with Conditioning

```
PrevSeq ← First ContextWin frames from TestSeq
PrevCond ← First ContextWin frames from TestSeq
while More samples to predict do
    CondData ← Extend PrevCond with Future Cond
    MotionPoses ← Extend PrevSeq with NaN for PredLen
    InputData ← Concatenate(CondData, MotionPoses)
    PredictedMotion ← MotionModel(InputData, N_samples)
    AvgMotion ← Average(PredictedMotion, axis=0)
    PrevSeq ← AvgMotion
    PrevCond ← CondData
end while
```

We average the motion over a set number of samples, currently 100, as we found the model is quite jittery on a frame per frame basis. This means our model is now semi-deterministic, while it's approach is still probabilistic our averaging gives us $\mathbb{E}[X]$. During training we do not use multiple samples, so the raw output from the model is still deterministic and for comparison we have included the no averaging model in our analysis.

VI. Evaluation

For our evaluation we will present both qualitative and quantitative measures of the models performance in the form of user serveys and probabilistic rankings. We will also discuss alternative evaluation metrics such as quantitative bone length and footstep sliding anylsis implemented by MoGlow. Below we will include some select frames from a test sequence for demonstrate the model running and walking. A link to our full collection of motion sequences can be found in the appendix.

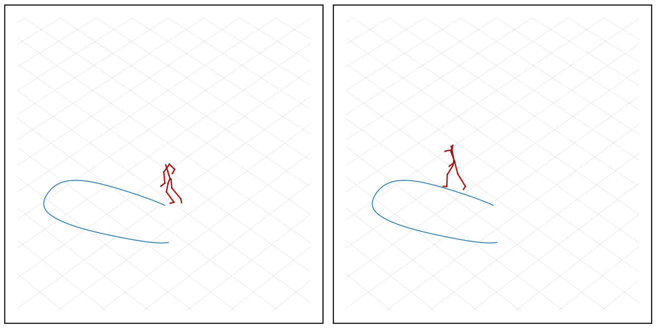


Fig. 3: Generated Sample (Frames 1 and 2)

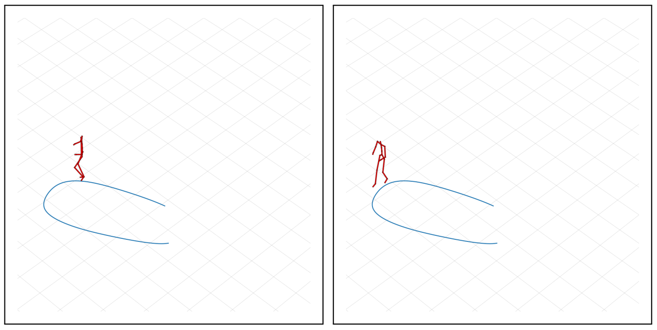


Fig. 4: Generated Sample (Frames 3 and 4)

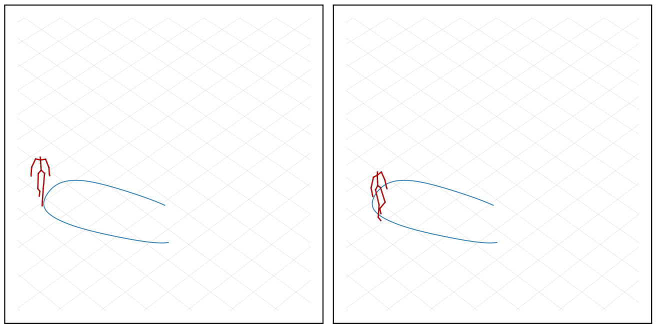


Fig. 5: Generated Sample (Frames 5 and 6)

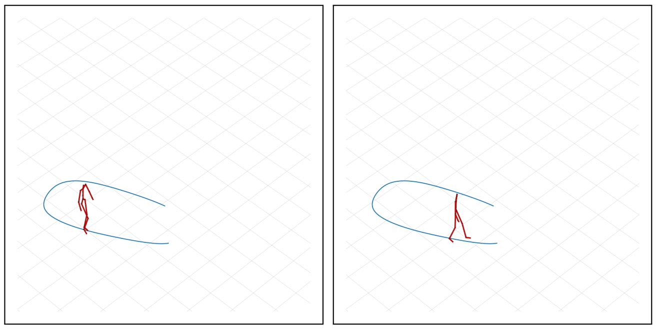


Fig. 6: Generated Sample (Frames 7 and 8)

A. Qualitative Evaluation

Our qualitative analysis was conducted from a user survey from 27 honours students. The given task was to rate the naturalness of the motion sequences provided from 1 to 5, with

1 being completely unnatural and 5 being that it is definitely real motion. We included the averaging, no averaging, and ground truth sequences so that we can get direct comparisons for the naturalness of the model.

TABLE I: Average Ratings of Naturalness of Generated Human Motion

Model	**Rating (1-5)**
Averaging	3.71
Ground Truth	3.95
No Averaging	2.90

Shown in Table I we can clearly observe that the averaged motion is rated significantly higher than the non averaged motion with a score of 3.71 compared to 2.90. Additionally, it appears that it was difficult for people to rate ground truth as the real data with a score of 3.95 out of 5. Comparing our averaged model to the ground truth we do get a very close result of 3.71 to 3.95 which suggests that our model is quite effective at generating motion similar to the ground truth. One set of ratings gave our model all 5's and the ground truth a 4, which confirms that it is quite difficult to tell the real motion apart. However, there is is still a 0.24 score difference which suggests that our model is not fully on par with real human motion.

Some raters gave feedback on the sequences they were shown, using this and personal observations of the animated sequences we can theorise on what could make this model more natural. During sharp corners the model did not lean in to corners as heavily as you'd expect for some sequences. A few sequences have the model running, slowing down for a corner, then continuing to walk the rest of the sequence causing foot sliding as it should have been running.

The samples in the survey were only a subset of the overall. Some of these had quite poor foot sliding artefacts and adherence to the control signals, mainly in regards to slow speed actions. Due to this they were not included and were instead noted as a topic for discussion.

Based on this feedback and additional analysis with the context of our architecture and the comment we made comparing it to MoGlow earlier, we can theorise on the main issue here. The previous poses in the autoregressive component is dominating the conditioning values. In the MoGlow model they separated our the conditioning values from the RNN whereas we do not, which is likely causing our issues. This is shown when previous poses are walking slowly and the conditioning information tells the model to speed up but it remains at the previous speed. The data dropout we applied to address this issue is not as effective as it should be due to it dropping out both conditioning and previous poses rather than only the previous poses. An improvement to our model would be the separation of the conditioning information from our LSTM and directly conditioning on the previous values.

Overall though, through subjective qualitative analysis we have found that our model performs very well compared to the ground truth data, though in the no averaging case it's performance is sub-par. Comparing our qualitative results to MoGlow's user survey we can find that their ground truth and model are rated higher but raters are still able to identify between the two with a similar margin to our own results. As an indicator of this methods validity though, these results show our model is very promising if given further development time.

B. Quantitative Evaluation

Our quantitative results will be analysis on our CPRS$_{\text{sum}}$ scores. This will be followed by comments on the bone length and foot sliding analysis done by MoGlow and why that has not been conducted here.

1) CRPS Evaluation: From the TimeGrad paper they define CPRS (Continuous Ranked Probability Score) as (24) which ranks the compatibility of x with the cumulative distribution function F. Their analysis uses the CPRS$_{\text{sum}}$ which sums across all dimensions of the model rather than examining a single vector's CPRS score.

$$\text{CRPS}(F, x) = \int_{-\infty}^{\infty} (F(y) - \mathbb{1}(y - x))^2 \, dy \tag{24}$$

$$\text{CRPS}_{\text{sum}} = \mathbb{E}_t \left[CPRS(\hat{F}_{sum}(t), \sum_i x_{i,t}^0) \right] \tag{25}$$

Recapping CRPS from the literature review, it is a proper scoring function to be used when not all methods present analytical forecast distributions or where likelihoods are not meaningfully defined. One issue with the use of this metric is that no other motion model have used this measure for evaluation, so we are unable to directly compare this. We can, however, compare it to experimental results from the TimeGrad paper.

Our underlying probabilistic model (no averaging) achieved a CPRS$_{\text{sum}}$ of 0.040, which is in the same ball-park range of other similarly dimensioned data presented in the TimeGrad paper. For a more meaningful comparison we can run our model on TimeGrad's Electricity dataset. With this we obtain a score of 0.018 compared to their 0.0206. Additionally, our score is better than a follow up paper to TimeGrad called ScoreGrad by Yan et al. [18] where they achieved a score of 0.0192. This quantitatively shows that our model is better than the original implementation and existing follow-up pieces.

2) Bone Length and Foot Sliding Analysis: Both bone length and foot sliding analysis are great metrics to measure both the consistency of the bone lengths over the and whether there is excessive sliding of the foot, as observed in our model, . However, MoGlow did not release their method for calculating these metrics, so any metrics produced would not be comparable. Preliminary findings for bone lengths also suggest that the motion data itself has large variances, possible due to the pooled dataset, which would cause validity issues of the variances produced. Therefore for our evaluation we have opted to exclude these metrics. Additionally, from the qualitative findings we have already established poor foot sliding artefacts in some conditions, so further analysis of this would not provide new insights.

C. Toy Model Examples

During the development of the full motion model we created a smaller toy model to demonstrate autoregression with diffusion and to test the effects of conditioning on the network. Shown in Fig 7 is the real vs predicted movement of the triangular distribution across the vector space. Intuitively, this shows how the autoregressive component takes previous actions, in this case the location of the triangular distribution, to produce continuous samples without discontinuities in time. Using the conditioning value of the step size $\pm$ 2 we can control the diffusion process to move the distribution either left or right. Abstracting this to our motion model, it takes in the previous frames and their conditioning then predicts future samples obeying our conditioned values.

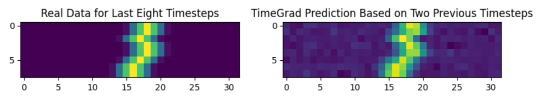


Fig. 7: Toy Model Predicted Sequence

VII. Future Work

Highlighted throughout this paper we have provided some insights on areas that can be improved and possible feature additions to this model. Here we will recap those methods providing justification as to why these should be investigated.

A. *Foot Sliding Artefacts*

Addressing this issue would require the modification of the architecture, mainly seperating the conditioning from the RNN and having model directly condition on the control inputs. Alternatively, there may be other ways to solve the autoregressive dominance of the previous poses which could be the focus of further research. MoGlow's implementation results in a fixed context window size despite the ability for LSTMs to represent longer range time dependencies, so a solution that did does not explicitly condition on the context window would be ideal.

B. *Alternative Diffusion Solvers*

While our results using the DEIS fast ODE solver are quite good, it is possible that we are sacrificing some level of detail in our system. So a further investigation into better solvers would be valuable. Real-time evaluation of the model is paramount, if a model like this were to be used in a game environment, so further research into methods like InstaFlow [5] would greatly benefit this model. However, at some point we may need to have some trade-off between execution speed and quality.

Additionally, a longer training time could possibly fix the jitters experienced in the non-averaged model which would allow us to have a purely probabilistic model.

C. *Textual Conditioning*

Textual conditioning is not a requirement but it would allow for more expressive control over the actions of the model. This would require a global context, so a transition to a transformer based network may be required for efficient computation across the entire sequence. From the same authors of MoGlow Listen, Denoise, Action! by Alexanderson et al [19] was developed which is an audio conditioned model for generating dance sequences could be adapted for this task with the addition of CLIP [17] conditioning.

VIII. Conclusion

In conclusion, we have presented our model which shows the potential of autoregressive conditional diffusion models for generating human motion sequences. Through a combination of qualitative and quantitative metrics we found that our model can generate motion that is difficult to subjectively differentiate from real human motion. With our user surveys indicating our averaged model achieved a naturalness score of 3.71 our of 5 compared to the ground truth data which achieved a score of 3.95. Additionally, quantitative metrics have shown our model produces better results in time-series forecasting than existed diffusion based generative models.

However, there are still key areas for improvement. Foot sliding artefacts remain an issue for this model when changing speeds and therefore need to be address. Different solvers and their effect on model performance and speed should be investigated.

Acknowledgments

I would like to thank my supervisor Bastiaan Kleijn for his support in this project by offering insightful discussions and advice for the direction of the project. Liam Crowley, a PhD student working on motion synthesis, was also a huge help with practical experience using MoGlow and TimeGrad. Kashif Rasul, the author of TimeGrad, was a great help by taking time out of his day to provide guidance for fixing the model and making it better. I'd also like to take the time to acknowledge my friends and family, especially Isaac Young, who have supported me during my time at university and by also providing feedback on this report. Finally, I'd like to thank those honours students who took part in my user survey for providing the results found in this report.

Appendix

Link to all generated sequences
Link to Motion Model
Link to Toy Model

References

[1] L. Mourot, L. Hoyet, F. L. Clerc, F. Schnitzler, and P. Hellier, "A survey on deep learning for skeleton-based human animation," *Computer Graphics Forum*, vol. 41, no. 1, pp. 122–157, nov 2021. [Online]. Available: https://doi.org/10.1111%2Fcgf.14426

[2] G. E. Henter, S. Alexanderson, and J. Beskow, "MoGlow," *ACM Transactions on Graphics*, vol. 39, no. 6, pp. 1–14, nov 2020. [Online]. Available: https://doi.org/10.1145%2F3414685.3417836

[3] W. Yin, R. Tu, H. Yin, D. Kragic, H. Kjellström, and M. Björkman, "Controllable motion synthesis and reconstruction with autoregressive diffusion models," 2023.

[4] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," 2023.

[5] X. Liu, X. Zhang, J. Ma, J. Peng, and Q. Liu, "Instaflow: One step is enough for high-quality diffusion-based text-to-image generation," 2023.

[6] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," 2021.

[7] P. von Platen, S. Patil, A. Lozhkov, P. Cuenca, N. Lambert, K. Rasul, M. Davaadorj, and T. Wolf, "Diffusers: State-of-the-art diffusion models," https://github.com/huggingface/diffusers, 2022.

[8] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," 2016.

[9] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," 2021.

[10] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2256–2265. [Online]. Available: https://proceedings.mlr.press/v37/sohl-dickstein15.html

[11] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2021.

[12] B. Kleijn, "Recurrent neural networks," September 2023, AIML425 2023 Lecture.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

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

[15] Q. Zhang and Y. Chen, "Fast sampling of diffusion models with exponential integrator," 2023.

[16] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu, "Motiondiffuse: Text-driven human motion generation with diffusion model," 2022.

[17] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[18] T. Yan, H. Zhang, T. Zhou, Y. Zhan, and Y. Xia, "Scoregrad: Multivariate probabilistic time series forecasting with continuous energy-based generative models," 2021.

[19] S. Alexanderson, R. Nagy, J. Beskow, and G. E. Henter, "Listen, denoise, action! audio-driven motion synthesis with diffusion models," *ACM Trans. Graph.*, vol. 42, no. 4, jul 2023. [Online]. Available: https://doi.org/10.1145/3592458