DOI: 10.1111/cgf.14116
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2020
J. Bender and T. Popa
(Guest Editors)

Volume 39 (2020), Number 8

Probabilistic Character Motion Synthesis using a Hierarchical Deep Latent Variable Model

S. Ghorbani[1], C. Wloka[1], A. Etemad[2], M. A. Brubaker[1], and N. F. Troje[1]

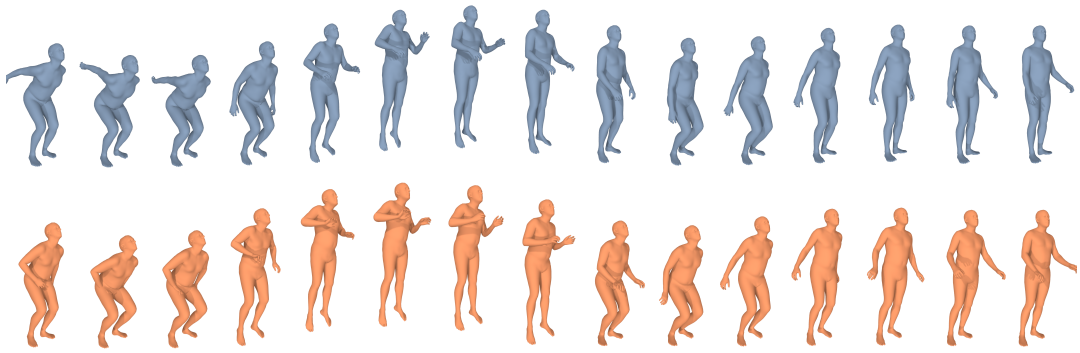


Figure 1: *Samples of a real motion sequence (blue) and synthesized motion sequence generated by our model (orange)*

Abstract

We present a probabilistic framework to generate character animations based on weak control signals, such that the synthesized motions are realistic while retaining the stochastic nature of human movement. The proposed architecture, which is designed as a hierarchical recurrent model, maps each sub-sequence of motions into a stochastic latent code using a variational autoencoder extended over the temporal domain. We also propose an objective function which respects the impact of each joint on the pose and compares the joint angles based on angular distance. We use two novel quantitative protocols and human qualitative assessment to demonstrate the ability of our model to generate convincing and diverse periodic and non-periodic motion sequences without the need for strong control signals.

CCS Concepts

• ***Computing methodologies*** → ***Animation; Machine learning approaches;***

1 Introduction

An active research area in computer animation is the automatic generation of realistic character animations given a set of control parameters. This can reduce the workload of key-framing, which is a laborious and time-consuming task done by skilled animators. Recent advances in motion capture technology and deep learning methods have increased interest in data-driven and learnable frameworks for modelling human motion. Most approaches model the motion sequences as a deterministic process, meaning that for a given set of control parameters only a single, fixed sequence is generated. On the other hand, human motion is stochastic in nature - given the same intention and target the joints always travel different paths. Hence, deterministic models fail to reflect such diversity which is an essential requirement for generating convincing and realistic character animation. Another challenge in designing a motion generative model is to enforce desirable motion sequences constrained by weak control signals such as action type. This is due to the fact that deterministic models usually regress to the mean pose in the long run as no strong control signal can be provided, especially for non-periodic movements, to steer the motion and reduce the motion uncertainty over time. Most recent controllable approaches are proposed only for periodic movements with strong control signals such as trajectory characteristics [HKS17, HSK16, PFAG19], or are limited to short-term predictions for non-periodic movements [MBR17, FLFM15, JZSS16]. Our work addresses these open issues by developing a model for animation synthesis which can be modulated by weak control signals while retaining the desired stochastic characteristics of human

motion across both temporal and spatial dimensions. Weak control signals are particularly useful for tasks in which large numbers of sequences are required, such as crowd simulation or providing data for other frameworks such as motion matching. In these situations requiring strong control signals such as trajectory characteristics would be unnecessarily labor intensive, whereas our framework can continuously generate novel motion clips with minimal user oversight.

Our proposed model for character animation synthesis is based on a deep recurrent neural network. We train our recurrent model on a large database of motion capture data such that it can generate novel, convincing motion samples that imitate the high-level stochastic nature of real data. This semi-supervised framework does not require any manual data preparation such as time-warping or motion clipping which minimizes the amount of manual work in the training and synthesizing processes.

Our framework is designed as a hierarchical recurrent latent variable network which models the spatiotemporal motion data with a two-level hierarchy. The hierarchical structure of the network architecture allows not only for motion sequences to be represented at multiple levels of abstraction but also for a higher level of desired variability in the generative process. The inner layer of the proposed architecture is designed as an extension of a variational recurrent neural network [CKD*15] which is conditioned on control signals and recursively processes high-level feature vectors (derived from motion subsequences) along with a stochastic latent variable. Defining this latent variable at a high level of abstraction enables the network to model the stochasticity observed in human movement. The inner layer is wrapped by encoder and decoder layers which encode the motion subsequences into feature vectors and decode the generated feature vectors back to motion subsequences.

We also propose a new objective function based on the geodesic distance between the ground-truth and reconstructed joint angles which has the following principal advantages: *i*) The geodesic distance better represents the deviation from the desired output than l_p norm losses. *ii*) The influence of different joints in the kinematic tree can be represented by assigning different weights to each joint. *iii*) High level and semantic information are integrated into the learning process by comparing the ground-truth and reconstructed sequences in the feature space of pre-trained classifiers.

We validated the performance of our model both qualitatively, via human scoring, and quantitatively through a novel evaluation protocol based on the *Inception Score (IS)* [SGZ*16] and *Fréchet Inception Distance Score (FID)* [HRU*17]. These metrics were first used for evaluation in image synthesis, and provide a measure not only of the quality of the generated output but also the diversity of output provided. Given the importance of movement variety for character animation synthesis, we have therefore adapted these metrics to provide a more complete evaluation than previously used metrics. The results show that our model effectively learns human motion dynamics and is capable of generating realistic and diverse character animation sequences coherent with control parameters, outperforming all other state-of-the-art models tested.

Our contributions can be summarized as: *i*) we propose a novel hierarchical generative recurrent architecture which effectively learns human motion dynamics and generates realistic character animation sequences coherent with control parameters, *ii*) we present a new objective function based on angular distances and the influence of different components in the kinematic tree which better represents network error and leads to improved learning, *iii*) we provide a new benchmark and evaluation protocol for character animation synthesis to measure both the quality and variability of generated sequences.

2 Related Work

Traditional Data-Driven Approaches: Data-driven approaches to character animation synthesis have been a popular area of research for nearly two decades. These approaches rely on motion capture data [MGT*19, SB06, GMT*20] which are provided as a sequence of poses represented by 3D joint locations or 3D joint angles of the skeleton at each time frame. With the advent of such motion capture datasets, many traditional approaches such as Motion Graphs [AF02, KG04], PCA-based models [SHP04], Kernel-based models [MK05, Muk11], and Hidden Markov models (HMMs) [TH00] were proposed for the task of motion synthesis. However, these approaches fail to model the complex nonlinear dynamics of human movements, especially for long-term multi-modal motion synthesis. For instance, HMMs require a hidden state size which is exponential in the number of components and therefore suffers from having a simple hidden state.

Early Deep-Learning-based Approaches: One of the earliest attempts to overcome the limitations of the above approaches, Taylor et al. [THR07, TH09, THR11] approached human motion modelling using variations of conditional Restricted Boltzmann Machines (cRBM) as an undirected energy-based model. They modelled the temporal dependency by adding poses from previous time steps as additional inputs. More recently, the impressive results achieved by deep generative models in other areas such as image and speech synthesis has encouraged researchers to model human movement using these models. The bulk of these works make use of recurrent neural networks (RNNs) [FLFM15, JZSS16, MBR17, AKH19, WHSZ19, WCX19, LLL18] as they have a high representational capacity in their internal state.

RNN-based Approaches: A notable approach which applies the recurrence step to a learned representation was introduced by Fragkiadaki et al. [FLFM15], who proposed an Encoder-Recurrent-Decoder (ERD) architecture which encodes each pose into a feature vector where it is recursively processed through a two-layer LSTM network. During motion synthesis, the prediction is fed back to the model in the following time step which causes the accumulation of small errors at each time step (called exposure bias). To address this problem, they corrupted the input by Gaussian noise with progressively increasing standard deviation as a type of curriculum learning. While it is hard to tune the amount of noise, this strategy was used in some of the subsequent proposed approaches as well [JZSS16]. To tackle the problem of exposure bias, Martinez et al [MBR17] exposed the model to its own prediction during training using a seq-2-seq architecture. They also enforced the model to learn velocities instead of absolute values via residual connections to address the problem of discontinuity in the seq-2-seq models.

Approaches Exploiting a Kinematic Tree: Both [FLFM15]

and [MBR17] modelled motion without an explicit representation of the kinematics tree, but this structural information is a potentially very useful model component. By explicitly modelling spatiotemporal interactions between joints, Jain et al. [JZSS16] combined the explicit representational power of spatiotemporal graphs with the implicit sequential learning of RNNs. Aksan et al. [AKH19] demonstrated an alternative method of incorporating structural information, proposing a Structured Prediction Layer (SPL) where the prediction of each joint at time t is conditioned on the joint's previous state and the current state of the parent joint. Therefore, at each time step, the joint angles are predicted starting from the root to the leaf nodes in the kinematic tree. They integrated the proposed layer in a variety of baseline architectures and showed improvements for the task of motion prediction.

Approaches based on Strong Control Signals: Although RNN-based methods overall show impressive results in short-term motion prediction, they fail in long-term generation due to their deterministic state assumption which fails to capture the intrinsic variability in human motion which compounds through time. Additionally, deterministic models assume a single future output, which causes them to converge over the long-term to a mean pose (referred to as *mean collapse*). Some of the proposed methods addressed this problem by adding additional information to the model to disambiguate the generative process. Holden et al [HKS17, HSK16] proposed providing foot contact and phase information as strong control parameters for locomotion movements to decrease the model uncertainty. Pavllo et al [PFAG19] augmented the generative network with a pre-trained *pace network* which provides the foot-step frequency, local speed, and facing direction given the trajectory. Martinez et al [MBR17] showed concatenating weak control parameters such as action type to the input sequence alleviates the mean collapse problem to some extent.

Probabilistic Approaches: Another way of avoiding converging to the mean pose is to model the intrinsic uncertainty of motion using probabilistic schemes. Many approaches were proposed based on Gaussian Process Latent Variable Models (GPLVM) [LWH*12, WHF06]. However, these models are limited due to their high memory cost for large data. More recently, adversarial learning has also been investigated for non-deterministic human motion modelling. Barsoum et al [BKL18] proposed a probabilistic motion prediction approach via GANs. Their model architecture is designed based on a seq-2-seq model and predicts multiple possible sequences from the same input. However, GANs are oftentimes hard to train, and their method was not designed to be steered by control signals. Using an alternative method also originally derived from image synthesis, Henter et al. [HAB19] proposed an autoregressive model based on normalizing flows (NFs) [RM15, DKB14]. They extended a variant of NFs, *GLOW* [KD18], to bipedal and quadrupedal motion sequences. However, their model needs strong trajectory control signals and is limited to locomotion synthesis. Similar to our proposed model, Habibie et al. [HHS*17] use a variational autoencoder (VAE) to model the spatial relationships. However, they extended their model to operate in an autoregressive fashion by setting the cell state of the LSTM components equal to the corresponding latent variable at each time step during training. While this approach successfully couples the LSTM representation to the posterior distribution, by collapsing the latent variable and internal state to one variable it limits the representational power of the model's internal state. Additionally, the balance their architecture strikes between control signals and previous cell state during generation limits model performance for non-periodic complex movements. We attempt to mitigate these drawbacks in our proposed model by formulating the internal state and latent code in two separate channels and conditioning the latent code to the previous internal state to model the temporal dependencies during test time. Recently, [LZCVDP20] et al. proposed an interesting model based on VAEs where the motion is controlled by setting the latent code as the output of a deep reinforcement learning module. Unlike our method, they modelled the motion by a Markovian assumption, meaning that each pose only depends on the previous pose and the autoregressive model is memoryless. They also modelled the VAE decoder as a Mixture of Experts (MoE) network.

Mixture-of-Experts Approaches: Another strategy exploited in [SZKS19, SZKZ20, LZCVDP20] to address the problem of mean collapse in multi-modal motion data is to use a Mixture-of-Experts (MoE) network where each expert is responsible for one mode in the training data. Though effective at mitigating mean collapse, the number of parameters in these networks increases with the number of experts.

Style Transfer Approaches: Motivated by recent advances in style transfer in images and videos, style transfer techniques were exploited to transfer the style of one animation clip to another [SCNW19, AWL*20]. While this method generates natural motion sequences with the desired style, these approaches cannot be directed by other control signals.

3 System Overview

A visual diagram of our model architecture is given in Fig 2. We provide a framework which encapsulates both the hierarchical and the stochastic nature of human motion within a deep hierarchical recurrent architecture. Our model generates motion sequences via a two-level hierarchy. In particular, we model the human motion as a sequence of high-level feature vectors called *Motion Words* where each Motion Word, $\mathbf{w}_i$, is computed as a function of a sub-sequence of poses. The recurrent processing of motion sequences is thereby applied at word-level rather than at pose-level.

We leverage an extension of a variational recurrent neural network [CKD*15] which contains a variational autoencoder at each time-step conditioned on the control signals. We call the recurrent processing unit a *Motion Cell* (green blocks in Fig 2) which attaches a stochastic latent variable to the observed Motion Words at each time-step. Stochasticity at the Word level enables variability to be represented at a higher level of abstraction (see section 3.2 for details), thereby producing more internally consistent motion sequences. The mapping between Motion Words and the corresponding sub-sequence of poses is performed by f_{enc} and f_{dec} (yellow blocks in Fig 2). At each time step, we condition the Motion Cell on the control signals to modulate the motion characteristics and decrease uncertainty due to the multi-modality nature of the motion generation process. In general, any motion-related attribute, static or dynamic, such as style, action type, or motion trajectory could be used as a control signal. However, in this work, we used a set of holistic attributes consisting of action type and gender.

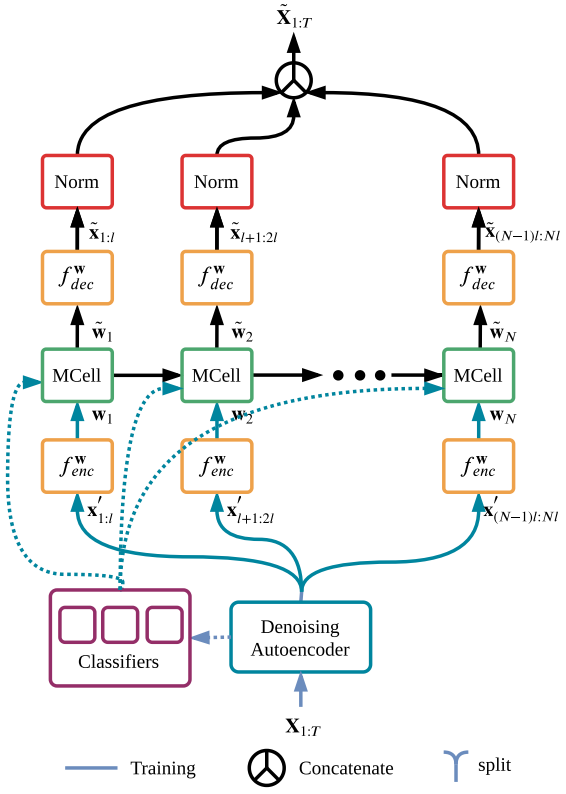


Figure 2: *An overview of our recurrent model. During training, denoised frames form temporal windows of equal size where each window $\mathbf{x}'_{(i-1)l+1:il}$ is projected into a high level feature vector $\mathbf{w}_i$ (called a Motion Word) via f^w_{enc}. Motion Cells operate on Motion Words in a latent space to combine information from the preceding sequence with stochastic variability to output the next step in the sequence. This output is converted to a joint angle representation via f^w_{dec}. A set of classifiers provide control signals for unlabelled input and normalization ensures that the representations fall within valid ranges.*

During training, we integrated individual pre-trained classifiers to the model for each attribute type to infer attributes from the unlabelled input sequence and also to provide additional higher-level learning signals to the objective function. This constrains the model to generate animations which fulfil the semantics defined by the attribute codes (see section 3.4 for details).

Our model uses a joint angle representation to define each pose. We tested the model with three different joint angle representations: axis-angle vectors, quaternions, and rotation matrices. To ensure that the model produces valid rotation for each joint, the estimated rotations in the output of the hierarchical recurrent neural network are normalized into valid rotations (red blocks in Fig 2). Regardless of the specific joint angle representation used, our model otherwise operates identically from one representation to the next. To have valid rotations represented by quaternions, the magnitude of the quaternions should be one. Therefore, we simply divided each quaternion by its magnitude. When instead using rotation matrices to represent joint angles, we applied the Gram–Schmidt orthonormalisation process on the output matrices. No normalization step was applied to the axis-angle vector representation.

3.1 Data Preprocessing

The local joint angle representation is augmented with processed root joint information which encodes the global transformation while keeping the final representation invariant to ground-plane (x-y) translation and rotation about the gravity direction. The augmented data includes forward direction velocity, sideways direction velocity, global root height, angular velocity around the gravitational axis, and the pitch and roll relative to the direction where the subject is facing. During motion synthesis, global translation and orientation can be recovered by integrating velocities over time while we assume the initial facing direction is in the direction of the x-axis in the global coordinate system. The final pose representation consists of a $D_p = 21 \times k + 6$ dimensional vector, where k is 3, 4 or 9 for axis-angle vectors, quaternions, and rotation matrices, respectively. We sub-sampled the motion sequences into 30 frames per second (they were originally recorded in 120 frames per second) and used all four offsets for training.

Our model can be trained by variable-length sequences of inputs. However, to speed up the training process by parallel computing we set the size of input sequences to a fixed size by clipping the longer sequences and padding zeros to the shorter ones. In our work, we set the length of input sequences to 200 frames (around 6.6 seconds). For synthesis the length of a generated sequence does not have to be equal to the length of the training sequences, rather our model can generate sequences with arbitrary length.

Before feeding to the main model, we apply a pre-trained denoising network to the training sequences to correct possible errors in the training data such as high-frequency noise resulting from marker occlusions or mislabelled markers in the motion capture process. The denoising network is implemented as a one-dimensional convolutional denoising autoencoder pretrained on a different subset of data than the one we used for training the main model. Details of the denoising network structure and its training process are given in Section 4.2.

3.2 Hierarchical Probabilistic Recurrent Network

Our hierarchical recurrent network models a motion sequence of length T with a two-level hierarchy. In the pose-level, we have a sequence of poses and in the word-level, we have a sequence of Motion Words. Each Motion Word, $\mathbf{w}_n \in \mathbb{R}^{D_w}$, summarizes a subsequence of poses using an encoding function

$$\mathbf{w}_n = f^{\mathbf{w}}_{\text{enc}}(\mathbf{X}_{nl:(n+1)l}), \tag{1}$$

where $\mathbf{X}_{1:T}$ is the sequence of poses ($\mathbf{X}_t \in \mathbb{R}^{D_p}$), $f^{\mathbf{w}}_{\text{enc}}$ is a non-linear complex encoder such as a fully-connected neural network, and l is the length of each sub-sequence. We define the sequence of Motion Words as an autoregressive model as follows:

$$p(\mathbf{w}_1, \ldots, \mathbf{x}_N) = p(\mathbf{w}_1) \prod_{t=2}^{N} p(\mathbf{w}_n | \mathbf{w}_{<n}), \tag{2}$$

where $N = \lfloor T/l \rfloor$ is the number of Motion Words in the sequence. l is considered as a hyperparameter where the best results were achieved for $l = 3$. The dimension of the Motion Word D_w was

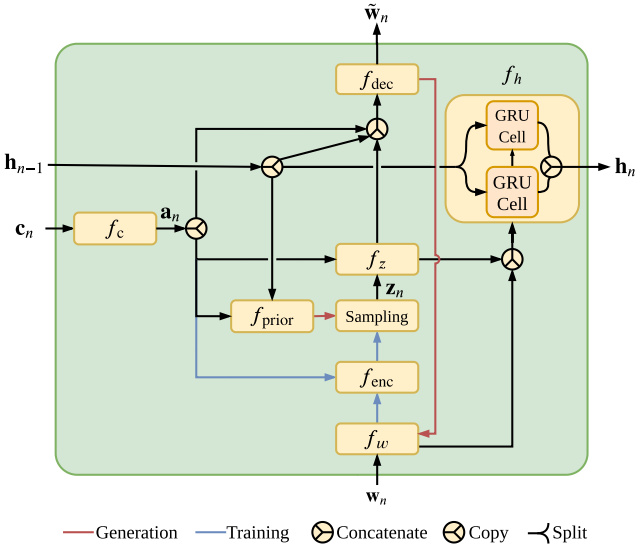


Figure 3: *Internal structure of Motion Cell. A Motion Cell can be viewed as a recurrent unit conditioned on control signals.*

set to $32 * 3 = 96$ which is equal to the approximate degrees of freedom in each pose ([LZCVDP20, PALvdP18]) times the length of each subsequence. To model the recursive structure of Motion Words we used a variational recurrent neural network [CKD*15] extended to condition on the control parameters. The proposed recursive model can be formulated as a recurrent neural network built upon a probabilistic recurrent cell which is structured as a conditional VAE at each time-step. We call these recurrent cells *Motion Cells* (green *MCell* blocks in Fig 2, see section 3.3 for more details). The combination of a hierarchical structure and probabilistic recurrence allows the model to define a stochastic latent variable at the word-level. Hence, the stochastic behaviour of the generation process is modelled at a deep level using highly abstracted features, allowing variation to be more easily sampled in an internally consistent manner from the learned feature space. We use another fully connected layer to convert the generated Motion Words back to the sub-sequence of poses

$$\tilde{\mathbf{X}}_{nl:(n+1)l} = f^{\mathbf{w}}_{\text{dec}}(\tilde{\mathbf{w}}_n), \tag{3}$$

where $\tilde{\mathbf{w}}_n$ is the output of the Motion Cell at time-step n (also called the reconstructed Motion Word) and $\tilde{\mathbf{X}}_{nl:(n+1)l}$ is the corresponding reconstructed sub-sequence. The details of the Motion Word encoder ($f^{\mathbf{w}}_{\text{dec}}$) and decoder ($f^{\mathbf{w}}_{\text{dec}}$) are given in Table 1, and the next section describes the internal structure of a Motion Cell.

3.3 Motion Cell

Our recurrent model is constructed by a probabilistic recurrent unit called a Motion Cell. The design of a Motion Cell is based on an entangled conditional VAE and a transition block. The VAE models the spatial dependencies and is additionally conditioned on control parameters and previous information. The transition block models the temporal dependencies and is a function of not only the input variable and previous internal state, but also the current latent variable. By conditioning both spatial and temporal paths on the latent variable, we introduce variability across both dimensions. The

Algorithm 1 This algorithm represents the FORWARD process of a Motion Cell for a single time-step during **training**. It takes as input a motion word $\mathbf{w}_n$, control signal $\mathbf{c}_n$, and previous internal state $\mathbf{h}_{n-1}$, and outputs a motion word $\tilde{\mathbf{w}}_n$ and updates the internal state to $\mathbf{h}_n$

function FORWARD($\mathbf{w}_n$, $\mathbf{c}_n$, $\mathbf{h}_{n-1}$)
  $\mathbf{a}_n = f_c(\mathbf{c}_n)$
  Compute Posterior distribution
  $\mu_{q,n} = f^{\mu}_{\text{enc}}\left(f_{\mathbf{w}}\left(\mathbf{w}_n\right), \mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  $\sigma_{q,n} = f^{\sigma}_{\text{enc}}\left(f_{\mathbf{w}}\left(\mathbf{w}_n\right), \mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  Sample latent variable from Posterior distribution
  (using reparameterization trick)
  $\mathbf{z}_n \sim \mathcal{N}(\mathbf{z}_n; \mu_{q,n}, \text{diag}\,\sigma^2_{q,n})$
  Compute Prior distribution
  $\mu_{p,n} = f^{\mu}_{\text{prior}}\left(\mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  $\sigma_{p,n} = f^{\sigma}_{\text{prior}}\left(\mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  Update internal state
  $\mathbf{h}_n = f_{\mathbf{h}}\left(f_{\mathbf{w}}\left(\mathbf{w}_n\right), f_{\mathbf{z}}\left(\mathbf{z}_n\right), \mathbf{h}_{n-1}\right)$
  Compute cell output
  $\tilde{\mathbf{w}}_n = f_{\text{dec}}\left(f_{\mathbf{z}}\left(\mathbf{z}_n\right), \mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  return ($\tilde{\mathbf{w}}_n$, $\mathbf{h}_n$, $\mu_{q,n}$, $\sigma_{q,n}$, $\mu_{p,n}$, $\sigma_{p,n}$)
end function

structure of a Motion Cell is illustrated in Fig 3. In the following, we describe in more detail how Motion Cells operate during training and generation phases.

3.3.1 Training Phase

Algorithm 1 provides the processing steps of a Motion Cell for a single time-step during training (the FORWARD function). Unlike a standard VAE, the posterior is not only conditioned on the input (observation) but also on the previous internal state and control parameters. A computationally inexpensive and common choice for the latent code distribution is a factorized Gaussian distribution

$$\begin{aligned} q(\mathbf{z}_n|\mathbf{w}_n, \mathbf{h}_{n-1}, \mathbf{a}_n) &= q(\mathbf{z}_n|\mathbf{w}_{\leq n}, \mathbf{z}_{<n}, \mathbf{a}_{\leq n}) \\ &= \mathcal{N}(\mathbf{z}_n; \mu_{q,n}, \text{diag}(\sigma^2_{q,n})), \end{aligned} \tag{4}$$

where $\mathbf{z}_n \in \mathbb{R}^{D_z}$ is the latent variable, $\mathbf{h}_n \in \mathbb{R}^{D_h}$ is the internal state of the Motion Cell which summarizes all the past information up to step n, and $\mathbf{a}_n = f_c(\mathbf{c}_n) \in \mathbb{R}^{D_a}$ is the feature vector extracted from control signals. In our model we only used weak attributes such as action type or style, either included as a component of sample labelling or inferred by integrated classifiers if the sample is unlabelled. However, the same methods can be straightforwardly extended to include other attributes, including dynamic parameters of the motion such as locomotion trajectory. We set $D_z = 96$, $D_h = 1024$, and $D_a = 8$.

The mean, $\mu_{q,n}$, and covariance parameters, $diag(\sigma_{q,n})$, are computed as:

$$\begin{aligned} \mu_{q,n} &= f^{\mu}_{\text{enc}}\left(f_{\mathbf{w}}\left(\mathbf{w}_n\right), \mathbf{h}_{n-1}, \mathbf{a}_n\right), \\ \sigma_{q,n} &= f^{\sigma}_{\text{enc}}\left(f_{\mathbf{w}}\left(\mathbf{w}_n\right), \mathbf{h}_{n-1}, \mathbf{a}_n\right), \end{aligned} \tag{5}$$

where f^{μ}_{enc} and f^{σ}_{enc} are non-linear complex functions such as multilayer perceptrons (MLP). $f_{\mathbf{w}}$ is also implemented as an MLP for

extracting Motion Word features, which is an essential requirement for learning complex motions. During training the latent variable is sampled from the posterior distribution using the reparameterization trick [KW14].

$$\mathbf{z}_n \sim \mathcal{N}\left(\mathbf{z}_n; \mu_{q,n}, \operatorname{diag}\sigma^2_{q,n}\right) \tag{6}$$

Similar to the posterior distribution, the prior distribution is also conditioned on the previous internal state and attribute vectors

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{h}_{n-1}, \mathbf{a}_n) &= p(\mathbf{z}_n|\mathbf{w}_{<n}, \mathbf{z}_{<n}, \mathbf{a}_{\leq n}) \\ &= \mathcal{N}(\mathbf{z}_n; \mu_{p,n}, \operatorname{diag}\sigma^2_{p,n}), \end{aligned} \tag{7}$$

where:

$$\begin{aligned} \mu_{p,n} &= f^{\mu}_{\text{prior}}\left(\mathbf{h}_{n-1}, \mathbf{a}_n\right), \\ \sigma_{p,n} &= f^{\sigma}_{\text{prior}}\left(\mathbf{h}_{n-1}, \mathbf{a}_n\right). \end{aligned} \tag{8}$$

where f^{μ}_{prior} and $f^{\sigma}_{\text{prior}}$ are implemented as MLPs. Conditioning the prior and posterior distributions on past information increases the temporal representational power of the model. Additionally, conditioning them on control parameters helps the model find distinct modes within the latent space.

In contrast to standard RNNs in which the output distribution is only conditioned on the previous internal state, the output distribution in the Motion Cell is also conditioned on the latent variable and control signals.

$$p(\mathbf{w}_n|\mathbf{z}_n, \mathbf{h}_{n-1}, \mathbf{a}_n) = p(\mathbf{w}_n|\mathbf{w}_{<n}, \mathbf{z}_{\leq n}, \mathbf{a}_{\leq n}). \tag{9}$$

In this work, we formulate the VAE decoder function deterministically, such that the reconstructed output, $\tilde{\mathbf{w}}_n$, is computed by an MLP:

$$\tilde{\mathbf{w}}_n = f_{\text{dec}}\left(f_{\mathbf{z}}\left(\mathbf{z}_n\right), \mathbf{h}_{n-1}, \mathbf{a}_n\right), \tag{10}$$

where $f_{\mathbf{z}}$ is a feature extraction MLP applied on the latent variable.

The internal state of the Motion Cell is updated by a transition function given the current input, previous internal state, and current latent variable:

$$\mathbf{h}_n = f_{\mathbf{h}}\left(f_{\mathbf{w}}\left(\mathbf{w}_n\right), f_{\mathbf{z}}\left(\mathbf{z}_n\right), \mathbf{h}_{n-1}\right). \tag{11}$$

Conditioning the internal state on the latent variable makes the temporal transition probabilistic and also helps the model address the mean collapse problem. Similar to [PFAG19], we used two stacked Gated Recurrent Units (GRU) with an internal state of size 512 for the transition function where the Motion Cell internal state is formed by concatenating the internal state of the two GRU cells. All of the components of the Motion Cell are learned by optimizing the objective function explained in section 3.5.

3.3.2 Generation Phase

Algorithm 2 provides the processing steps for a single time-step of a Motion Cell during motion synthesis (the SAMPLE function). At each time step during generation the latent variable is sampled from a prior distribution, computed in the same manner as the posterior distribution sampling done in the training phase (Eq. 7)

$$\mathbf{z}_n \sim \mathcal{N}\left(\mathbf{z}_n; \mu_{p,n}, \operatorname{diag}\sigma^2_{p,n}\right). \tag{12}$$

Algorithm 2 This algorithm represents the SAMPLE process of a Motion Cell for a single time-step during **generation**. It takes control signal $\mathbf{c}_n$ and previous internal state $\mathbf{h}_{n-1}$, and generates motion word $\tilde{\mathbf{w}}_n$ and current internal state $\mathbf{h}_n$

function SAMPLE($\mathbf{c}_n$, $\mathbf{h}_{n-1}$)
  $\mathbf{a}_n = f_c(\mathbf{c}_n)$
  $\mu_{p,n} = f^{\mu}_{\text{prior}}\left(\mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  $\sigma_{p,n} = f^{\sigma}_{\text{prior}}\left(\mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  Sample latent variable from Prior distribution
  (using reparameterization trick)
  $\mathbf{z}_n \sim \mathcal{N}(\mathbf{z}_n; \mu_{p,n}, \operatorname{diag}\sigma^2_{p,n})$
  Compute cell output
  $\tilde{\mathbf{w}}_n = f_{\text{dec}}\left(f_{\mathbf{z}}\left(\mathbf{z}_n\right), \mathbf{h}_{n-1}, \mathbf{a}_n\right)$
  Update internal state
  $\mathbf{h}_n = f_{\mathbf{h}}\left(f_{\mathbf{w}}\left(\tilde{\mathbf{w}}_n\right), f_{\mathbf{z}}\left(\mathbf{z}_n\right), \mathbf{h}_{n-1}\right)$
  return ($\tilde{\mathbf{w}}_n$, $\mathbf{h}_n$)
end function

The latent variable is then used with the previous internal state and control signals to generate the reconstructed Motion Word $\tilde{\mathbf{w}}_n$ (Eq. 10). Finally, the internal state is updated using the previous internal state, current latent vector, and the reconstructed Motion Word.

$$\mathbf{h}_n = f_{\mathbf{h}}\left(f_{\mathbf{w}}\left(\tilde{\mathbf{w}}_n\right), f_{\mathbf{z}}\left(\mathbf{z}_n\right), \mathbf{h}_{n-1}\right). \tag{13}$$

3.4 Attributes Classifiers

For each attribute type, we integrate a separate pre-trained classifier into the generative model. Integrating classifiers into the hierarchical probabilistic recurrent network serves three purposes. First, they provide control parameters to the generative model for unlabelled data, allowing our system to operate in a semi-supervised manner (dashed arrows in Fig 2). Second, during training, the classifiers provide additional high-level signals (both from their intermediate layers as well as the output class inferred by the classifier) to the objective function. This constrains the generative model to generate motions semantically coherent with the motion attributes (see Section 3.5.2). Third, the classifiers can be used for the evaluation of our generative model (see Section 5).

We implemented all the classifiers using one-dimensional convolutional neural networks and trained them on 50% of the training data. We observed that this amount of training data is sufficient to label the rest of the data with a high accuracy. Further details of classifier implementation are given in Section 4.

3.5 Objective Function

We formulate model training as an optimization problem by minimizing the objective function

$$\mathcal{L} = \mathcal{L}_{RVAE} + \lambda_{CL}\mathcal{L}_{CL} + \lambda_{Ang}\mathcal{L}_{Ang}, \tag{14}$$

where $\mathcal{L}_{RVAE}$ is the recurrent VAE loss equal to the sum of the negative step-wise variational lower bound over the whole sequence. We define a new hierarchical geodesic loss for reconstruction part of $\mathcal{L}_{RVAE}$ which is more accurate than the l_p norm loss and takes into account the relative impact of each joint in the kinematic tree on the final loss. $\mathcal{L}_{CL}$ is the complementary loss provided by the

classifiers, found by evaluating the ground-truth and reconstructed samples in the intermediate and last layer of each classifier. $\mathcal{L}_{Ang}$ is the sum of constraints encouraging the model to produce valid joint representation. We will describe each term in more detail below.

3.5.1 RVAE Objective

The first term in our objective function, $\mathcal{L}_{RVAE}$, is defined as a variational autoencoder objective summed over all sequence steps as follows

$$\begin{aligned}\mathcal{L}_{RVAE} &= \mathbf{E}_{q(\mathbf{z}_{\leq N}|\mathbf{w}_{\leq N},\mathbf{a}_{\leq N})}\Bigg[\sum_{n=1}^{N=T/l} -\log p\left(\mathbf{w}_n|\mathbf{z}_{\leq n},\mathbf{w}_{<n},\mathbf{a}_{\leq n}\right) \\ &\quad + \lambda_{KL}\mathrm{KL}(q(\mathbf{z}_n|\mathbf{w}_{\leq n},\mathbf{z}_{<n},\mathbf{a}_{\leq n})\|p(\mathbf{z}_n|\mathbf{w}_{<n},\mathbf{z}_{<n},\mathbf{a}_{\leq n}))\Bigg]. \\ &= \mathcal{L}_{rec} + \lambda_{KL}\mathcal{L}_{KL}\end{aligned} \tag{15}$$

The first term in the above loss is the expected log-likelihood or reconstruction loss which is usually defined as the distance between observations and the reconstructed values. We define our reconstruction term as a custom loss over joint angles rather than Motion Words to simultaneously train the Motion Word encoder $f^{\mathbf{w}}_{\text{enc}}$ and decoder $f^{\mathbf{w}}_{\text{dec}}$. The second term in the summation is the KL-divergence between the posterior and the prior at time-step n weighted by λ_{KL}. To prevent optimization process from getting stuck in an undesirable stable equilibrium we used an annealing scheduler for λ_{KL} where the optimization is performed for a few epochs with $\lambda_{KL} = 0$ (warm-up phase), then λ_{KL} is slowly increased from 0 to 1 (annealing phase), and then for the last few epochs we set $\lambda_{KL} = 1$ (cool-down phase) [BVV*15]. In the following we describe how the reconstruction loss is formulated.

Geodesic Distance of Joint Angles: Assuming a deterministic prediction in joint angles, the first term can be defined as a reconstruction loss. Often, metrics in the Euclidean space such as l_1 and l_2 norms are used as the reconstruction loss. However, these metrics do not represent the geodesic distance of two rotations which confuses the training process especially for large angular distances and at the beginning of the optimization process. To address the above-mentioned problems in Euclidean distances, we define more relevant distance functions which respect the intrinsic structure of 3D rotations both for quaternions and rotation matrices. The angular distance between two unit quaternions $\mathbf{q}$ and $\tilde{\mathbf{q}}$ is defined as

$$d(\mathbf{q},\tilde{\mathbf{q}}) = \mathbf{q}\tilde{\mathbf{q}}^{-1} = 2\arccos(\mathbf{q}\cdot\tilde{\mathbf{q}}). \tag{16}$$

Since the quaternions double-cover the space of rotations meaning that quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation we can take into account this ambiguity by modifying the above function as

$$d_1(\mathbf{q},\tilde{\mathbf{q}}) = 2\arccos(|\mathbf{q}\cdot\tilde{\mathbf{q}}|) \tag{17}$$

Since arccos is a monotonically decreasing function we can define an approximate but computationally less expensive distance as

$$d_2(\mathbf{q},\tilde{\mathbf{q}}) = 1 - |\mathbf{q}\cdot\tilde{\mathbf{q}}|. \tag{18}$$

which only needs 4 multiplication and 1 comparison for each pair of quaternions [Huy09]. It can be proven that the square of the l_2 norm of two unit quaternions is equivalent to Eq.18 for small angular distances

$$\begin{aligned}\|\mathbf{q}-\tilde{\mathbf{q}}\|^2 &= \|\mathbf{q}\|^2 + \|\tilde{\mathbf{q}}\|^2 - 2(\mathbf{q}\cdot\tilde{\mathbf{q}}) \\ &= 2(1-\mathbf{q}\cdot\tilde{\mathbf{q}})\end{aligned} \tag{19}$$

Similarly, we can modify the above measure to disambiguate the quaternions representations as follows:

$$d_3(\mathbf{q},\tilde{\mathbf{q}}) = \min\left\{\|\mathbf{q}-\tilde{\mathbf{q}}\|^2, \|\mathbf{q}+\tilde{\mathbf{q}}\|^2\right\} \tag{20}$$

All distance measures d_1, d_2, and d_3 address the double-coverage problem, however, the last two are approximations and do not measure the exact geodesic distance.

Similarly, for the scenarios where the joint angles are represented by rotation matrices, we can use the Geodesic distance between a pair of rotation matrices using logarithm map in $SO(3)$ as follows

$$d(\mathbf{R},\tilde{\mathbf{R}}) = \|\log(\tilde{\mathbf{R}}\mathbf{R}^\top)\|, \tag{21}$$

where $\|\log(\tilde{\mathbf{R}}\mathbf{R}^\top)\|$ is a skew-symmetric matrix containing the rotation axis-angle components and therefore $\|\log(\mathbf{R})\|$ is the magnitude of the angular distance multiplied by a constant.

Hierarchical Loss: Proposed approaches in human motion modelling represent each human pose either by 3D joint locations in a global or body's local coordinate system, or 3D joint angles where, given the limbs' length, the final position and orientation of the body parts are calculated by forward kinematics. Models which use 3D joints locations usually normalize the skeleton size of the training samples and define the loss as an l_p norm over joint locations [HSK16, HKS17]. The main problem in such approaches is that during training and generation they are not exploiting the constraints imposed by parameterized skeleton and limbs rigidity. Therefore, the generation phase should be followed by a corrective re-projection onto a valid character skeleton.

Modelling poses by joint angles inherently follows the constraints imposed by parameterized skeleton [THR07, MBR17, FLFM15, JZSS16]. However, defining loss over joint angles ignore the amount of influence that each joint contributes to the learning process and gives all joints equal weights. On the other hand, an error in a parent joint has more impact on the final pose than the same amount of error in its child joints. This is due to the fact that an error in the parent joints propagates through all of its children down to the leaf nodes in the kinematic tree during forward kinematics. Recently, [PFAG19] proposed using joint angles to represent body pose but defined the loss over joint locations by applying a differentiable forward kinematics on ground-truth and predicted joint angles. However, applying forward kinematics at each pose is computationally expensive especially for long sequences and when the number of joints is high.

In this work, we propose a hierarchical loss over joint angles which weights each joint's error based on its impact on the reconstructed pose as follows

$$\mathcal{L}_{rec}(t) = \sum_{k=1}^{K} \alpha_k d(\mathbf{X}^k_t, \tilde{\mathbf{X}}^k_t), \tag{22}$$

where $\mathbf{X}^k_t$ and $\tilde{\mathbf{X}}^k_t$ are the ground-truth and the reconstructed joint angles for joint k at time t and $d(.)$ is one of the distance functions defined in Eq.17, 18, 20, or 21. α_k is the impact factor which

weights the impact of the corresponding joint angle on the pose reconstruction. A rule of thumb for choosing α_ks is that the child joint should be weighted with a lower impact factor compared to its parent joint ($\alpha_k < \alpha_{parent(k)}$) in the kinematic tree. In this work, we set α_k as the maximum path length from joint k down to all of the connected end-effectors in an average body skeleton. We can define α_k recursively as follows

$$\alpha_k = \max_j(\alpha_j + l_{k-j}), j \in SC_k, \tag{23}$$

where SC_k is the set of all children of joint k, and l_{k-j} is the length of the bone connecting joints l and j. As suggested by [PFAG19] we also evaluated the results by applying forward kinematics and computed the positional loss. In practice, the results were very close, while the latter took around 35% longer for training.

3.5.2 Classifiers Loss

The classifiers are trained to infer the motion attributes and incorporate a complementary loss from the output of intermediate and final layers. This complementary loss can be defined as

$$\mathcal{L}_{CL}(t) = \sum_{c=1}^{C} \sum_{l \in L_c} \beta_{(c,l)} d(l(\mathbf{X}_t), l(\tilde{\mathbf{X}}_t)), \tag{24}$$

where C is the number of classifiers and L_c is the set of layers in cth classifier. $d(l(\mathbf{X}), l(\tilde{\mathbf{X}}))$ computes the loss for the output of layer l given ground-truth and reconstructed samples. $\beta_{(c,l)}$ is a predefined weight assigned for each layer. We compute the loss by using the l_2 norm for intermediate layers and cross-entropy loss for the attribute labels. Details of the classifiers' architecture are given in Table 1.

3.5.3 Intrinsic Rotation Representation Constraints

In order to encourage the model to produce valid rotations, we add some constraint terms to the final objective function based on the representation we use for the joint angles. This helps to better ensure convergence at the beginning of the training process and smooths the optimization landscape. Although we normalize the output of $f^{\mathbf{w}}_{\text{dec}}$, better performance is achieved when these outputs are very close to valid rotations leaving the role of normalizers as only a final correction on very small errors.

For rotation matrices we define two constraints: orthogonality and unit determinant, formulated as follows:

$$\begin{aligned}\mathcal{L}_{\text{ang}}(t) &= c_1 \mathcal{L}_{\text{orth}}(t) + c_2 \mathcal{L}_{\text{det}}(t) \\ &= \sum_{k=1}^{K} \left(c_1 \|\tilde{R}^k_t (\tilde{R}^k_t)^\top - I\|^2_2 + c_2 |\det(\tilde{R}^k_t) - 1| \right)\end{aligned} \tag{25}$$

where the first term encourages the orthogonality of the output matrices and the second term enforces the model to produce matrices with a unit determinant. We also added Sigmoid activation to the output of $f^{\mathbf{w}}_{\text{dec}}$ to ensure that the elements of the output matrices are in the range of $[0,1]$.

For quaternions we only need to set the unit length constraint

$$\mathcal{L}_{\text{ang}} = \mathcal{L}_{q-norm} = \sum_{k=1}^{K} |\|\tilde{\mathbf{q}}^k_t\|^2_2 - 1| \tag{26}$$

For axis-angle rotation representation we did not set any constraint as they represent the three degrees of freedom by only three scalars.

Function	Architecture
f_w, f_z	$2 \times [\text{FC}(128) + \text{ELU}] + \text{FC}(96) + \text{ELU}$
f_{dec}	$2 \times [\text{FC}(128) + \text{ELU}] + \text{FC}(96) + \text{ELU}$
f^{μ}_{enc}, f^{μ}_{prior}	$4 \times [\text{FC}(128) + \text{ELU}] + \text{FC}(96)$
f^{σ}_{enc}, f^{σ}_{prior}	$4 \times [\text{FC}(128) + \text{ELU}] + \text{FC}(96) + \text{Softplus}$
f_h	$2 \times \text{GRUCell}(512)$
$f^{\mathbf{w}}_{\text{enc}}$	$2 \times [\text{FC}(128) + \text{ELU}] + \text{FC}(96) + \text{ELU}$
$f^{\mathbf{w}}_{\text{dec}}$	$2 \times [\text{FC}(128) + \text{ELU}] + \text{FC}(3 \times D_P)$
Classifiers	$3 \times [\text{Conv1D} + \text{ReLU}] + \text{AdaptiveAvgPool1D} + \text{FC}(N_C)$
Denoising Autoencoder	$2 \times [\text{Conv1D} + \text{ReLU}] + \text{ConvTranspose1D} + \text{ReLU} + \text{ConvTranspose1D}$

Table 1: *The architecture of model components.* FC(n) *is the abbreviation for Fully Connected linear layer with* n *nodes.* Conv1D *and* ConvTranspose1D *are one-dimensional convolution and transposed convolution layers.* AdaptiveAvgPool1D *is one-dimensional adaptive average pooling layer.*

4 Implementation and Training

4.1 Dataset

We trained and evaluated our model on a subset of AMASS [MGT*19], a very large database of human motion which unifies different marker-based motion capture datasets by representing them in a common framework. The kinematic tree is represented by 21 joints and the root (pelvis). We used the MoVi [GMT*20] and RuB [Tro02] datasets from AMASS for training and evaluating the main module and the rest of the AMASS data for training the denoising autoencoder.

The control parameters in our model are action type and gender. We used a subset of actions: walking, jogging, jumping, and lifting. The data were split into 150, 25, and 25 subjects for the purpose of training, validation, and testing, respectively. All splits contained male and female subjects in equal proportion.

4.2 Training Process

The details of the model architecture are given in Table 1. All model components were implemented using the PyTorch library.

For training the hierarchical model (Motion Cell, $f^{\mathbf{w}}_{\text{enc}}$, and $f^{\mathbf{w}}_{\text{dec}}$), we optimized the objective function in Eq.14 with the joint angle distances computed by Eq .17 using Adam optimizer [KB14] with a learning rate of 0.001, no weight decay, and a batch size of 64. We also set the gradient norm clipping to 0.1 to avoid any exploding gradients. All weights of the model were initialized using Kaiming initialization [HZRS15]. We trained our network for 1600 epochs which took around 2 hours on a GeForce RTX 2080 Ti GPU. The scheduling of different loss component coefficients during training can be found in the supplementary material.

For each combination of attributes, the initial internal states of the GRU cells is learned as a Gaussian distribution. Then each se-

quence is initialized by sampling the initial state from the distribution which corresponds to the required attribute.

We train our recurrent model in a teacher forcing scheme (i.e. the ground-truth input is provided to the Motion Cell at each time-step during training). Although this is an effective and fast approach for training, the model is prone to exposure bias and risks overfitting to the training data. To address this problem we experimented with three different mitigating strategies: (i) progressively corrupting input by adding Gaussian noise [FLFM15,JZSS16], (ii) progressively dropping motion words and exposing the model to its own previous output [MBR17, PFAG19], and (iii) adding joint-wise dropout on the input poses [GSAH17]. We achieved the best results when we used the second strategy with a scheduled drop rate which helps with the problem of foot skating as well.

We trained all classifiers with similar architecture (Table 1) on 50% of training data using Adam optimizer with a learning rate of 0.005 for 30 epochs. We used Adaptive Average Pooling before the last fully connected layer to adapt the classifier models to different input lengths.

We trained the denoising autoencoder separately and on the rest of the AMASS data. During training 10% of the input dimensions were chosen randomly and corrupted by Gaussian noise with zero mean and standard deviation of 0.5. We trained this model for 300 epochs and using the Adam optimizer with a learning rate of $1e-4$ with an exponential decay of 0.99 per 10 epoch.

5 Experiments and Evaluation

5.1 Models and Ablations

For the purpose of comparison, we compared our model with Pavllo et al.'s Quaternet [PFAG19] and Fragkiadaki et al.'s Encoder-Recurrent-Decoder (ERD) model [FLFM15]. These two models were trained on the same training data with the same training hyperparameter optimization techniques as our model. The initial internal state was learned in the same way to our model. Sampling the initial state is the only source of stochasticity in these two models.

For all models, we used a common generation scheme. Each walking or jogging sequence was generated with 140 frames, and the first 20 frames were discarded during evaluation (resulting in 4 seconds of motion). For the non-periodic actions (jumping and lifting), we terminated the generated sequence when they collapse to the mean pose.

In order to evaluate the influence of model components, we trained three additional ablated configurations of the our model. In the first ablated configuration, "Proposed(SL)" (for "Single Layer"), we removed the hierarchical encoder $f^{\mathbf{w}}_{\text{enc}}$ and decoder $f^{\mathbf{w}}_{\text{dec}}$, and fed the individual poses directly to the recurrent model. For the second ablation configuration, "Proposed(NL)" (for "Normal Loss"), we disabled the influence of hierarchical loss by setting all α_k coefficients in the Eq. 22 to 1. The last ablation, "Proposed(NC)" (for "No Classifier"), disabled the influence of classifiers on the final loss by setting λ_{CL} to zero.

For all evaluated models, we achieved comparable quantitative results between quaternion and rotation matrix representations, both of which outperformed axis-angle representations. Therefore, for the rest of the paper, we only report the results for quaternions.

5.2 Quantitative Evaluation

In this work, we evaluate models based on two main criteria: quality and diversity. We expect the generated samples to be realistic and coherent with the attributes which are set as control parameters (quality). In addition, we expect the model to generate motions with high diversity and natural stochasticity while still following the manifold of realistic motions (diversity). To codify both criteria in our quantitative evaluation we use the *Inception Score (IS)* [SGZ*16] and *Fréchet Inception Distance Score (FID)* [HRU*17] metrics which were originally proposed for image generative models. Both evaluation metrics have been shown to correlate well with human evaluation on generated images.

IS is formulated based on two criteria, diversity and quality, defined as follows:

$$\text{IS} = \exp\left(\mathbf{E}_{\tilde{X}\sim p_g} D_{KL}(p(\mathbf{a}|\tilde{\mathbf{X}})\|p(\mathbf{a}))\right) \tag{27}$$

where $\tilde{\mathbf{X}}$ is a synthetic sample generated by a generative model, $p(\mathbf{a}|\tilde{\mathbf{X}})$ is the conditional attribute distribution of a classifier which is pre-trained on separate training data, and $p(\mathbf{a}) = \int_{\tilde{\mathbf{X}}} p(\mathbf{a}|\tilde{\mathbf{X}}) p_{\text{model}}(\tilde{\mathbf{X}})$ is the marginal attribute distribution. Equation 27 can be also written as $\text{IS} = \exp\left(H(\mathbf{a}) - H(\mathbf{a}|\tilde{\mathbf{X}})\right)$, where $H(\mathbf{a})$ and $H(\mathbf{a}|\tilde{\mathbf{X}})$ are the attribute entropy and the conditional attribute entropy, respectively. Generated animations which fulfil the semantics defined by the attributes should have a conditional attribute distribution $p(\mathbf{a}|\tilde{\mathbf{X}})$ with low entropy. In other words, the classifier should be very confident about the attribute associated with the generated animation. On the other hand, we expect our model to generate a high variety of motions for each attribute class, therefore, $p(\mathbf{a})$ should have a high entropy. An estimator of IS as follows

$$\text{IS} \approx \exp\left(\frac{1}{M}\sum_{i=1}^{M} D_{KL}\left(p(\mathbf{a}|\tilde{\mathbf{X}}^{(i)})\|\hat{p}(\mathbf{a})\right)\right), \tag{28}$$

where $\tilde{\mathbf{X}}^{(i)}$ is a generated motion sample and $\hat{p}(\mathbf{a}) = \frac{1}{M}\sum_{i=1}^{M} p\left(\mathbf{a}|\tilde{\mathbf{X}}^{(i)}\right)$ is the empirical conditional distribution.

FID captures the similarity between the generated and the real motion samples. It evaluates the model by comparing the statistics of a set of generated samples to a set of real motion sequences from the dataset. Similar to IS, we use a classifier trained on a separate dataset. Then, the activations of the last feature extraction layer (the last layer prior to the last fully connected layer) are summarized as a multivariate Gaussian distribution for synthetic and real data. The distance between the two distributions is then computed with Fréchet Distance as follows:

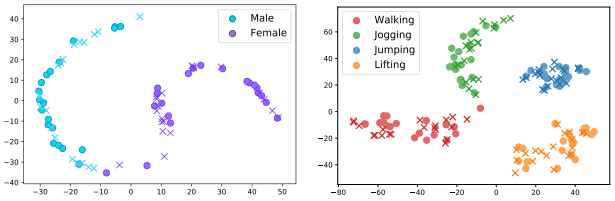
$$\text{FID} = \|\mu_g - \mu_d\|_2^2 + \text{Tr}\left(\Sigma_g + \Sigma_d - 2\left(\Sigma_g\Sigma_d\right)^{\frac{1}{2}}\right), \tag{29}$$

where $\mathcal{N}(\mu_g, \Sigma_g)$ and $\mathcal{N}(\mu_d, \Sigma_d)$ are the distributions of the activations in the last feature extraction layer for synthetic and real data, respectively.

Results: The results of quantitative evaluations are shown in Table 2. Using each model, we generated 1000 samples for each combination of attributes ($M = 8000$ in total). Quaternet and ERD generated convincing walking and jogging samples. However, since the only source of stochasticity is the initial hidden state, these models fail to generate a diverse set of sequences (ERD's performance was

Model	IS ↑	FID ↓
Quaternet [PFAG19]	5.12	92.31
ERD [FLFM15]	5.91	86.42
Proposed(SL)	6.45	31.41
Proposed(NL)	7.11	17.3
Proposed(NC)	7.43	11.92
Proposed	7.52	10.45
Real data	7.64	0

Table 2: *Results from quantitative evaluations using IS (higher score is better) and FID (lower score is better).*



(a) *Samples for different genders* **(b)** *Samples for different actions*

Figure 4: *Visualizing the activations of the last layer of action classifier projected onto two dimensions using t-sne for real data (circles) and data synthesized data (crosses) by our model. As can be seen, synthesized samples strongly coincide with the corresponding clusters formed by real data.*

slightly better due to its hierarchical structure yielding higher diversity at the beginning of the motions). In addition, they usually failed to generate a complete sequence for non-periodic motion such as lifting and were regressed to the mean pose after $70-80$ frames. Among ablation configurations, Proposed(SL) had the lowest performance showing the significant influence of hierarchical structure in generating diverse motions (higher $H(\mathbf{a})$). The lower scores for the other two ablation configurations (Proposed(NL) and Proposed(NC)) indicate the impact of our hierarchical loss structure and the effect of integrating classifiers into the loss, respectively, on the conditional attribute entropy and higher motion quality.

5.3 Qualitative Evaluation

To qualitatively evaluate how realistic and natural the synthetic animations are we also performed an experiment for subjective evaluation. All six evaluated models in the previous section were used in this experiment as well. We sampled five sequences for each of the four action types from the synthesized sequences of each model resulting in $6*4*5=120$ synthesized samples which were added to 20 motion sequences from real data. 20 human observers rated each motion sample from 1 (completely unrealistic) to 10 (completely realistic). The motion clips were displayed to the raters in a randomized order. Raters were asked to rate each animation after it was displayed completely, and no information about the aim of the experiment was given to the raters. We used a few motion clips for the purpose of training the raters before each experiment.

Results: The results of our qualitative evaluation are illustrated in Table 3. The qualitative results correlate well with the quantitative results (Pearson correlations of 0.86 and -0.95 with IS and FID, respectively). Quaternet and ERD achieved the lowest ratings for non-periodic actions (jumping and lifting) since they usually fail to complete these motions and instead regressed to the mean pose in the last frames. Among ablation configurations, Proposed(NL) achieved the lowest mean rating, which shows the impact of hierarchical loss on having more realistic motions. Although the main goal of our hierarchical structure is to improve the diversity of motions, the lower ratings achieved by Proposed(SL) compared to the main model nevertheless demonstrates the effectiveness of our hierarchical architecture even on short sequences. Similar to the quantitative evaluation, disabling the hierarchical structure of the loss function (Proposed(NL)) resulted in decreased ratings, suggesting the importance of classifiers to better learn action modes on the motion manifold.

Figure 4 shows the visualization of motion sequences in a two-dimensional space. We sampled 20 sequence for each action type and gender from real data (circles) and the sequences generated by our model (crosses), extracting the activations from the last layer of classifiers. We then applied *t-sne* [MH08] dimensionality reduction to project the activations onto two dimensions. As seen, our model generates sequences with similar diversity to real data while still accurately separating the modes for each action type and gender.

5.4 Discussion

In this work, we propose a motion generative model with a focus on preserving the stochastic nature of human motion while generating convincing and natural spatiotemporal motion sequences. The proposed model uses a deep hierarchical recurrent framework which can further be tuned via weak control signals such as action type. Each sequence is generated using a probabilistic recurrent structure which models the underlying stochasticity by injecting noise in an abstract level. We also propose a novel hierarchical geodesic loss which incorporates the structural information of the kinematic tree and compares joint angles based on angular distances, yielding a better representation of error and more accurate learning.

Limitations and Future Work. The proposed architecture was implemented for four different action types in addition to the gender attribute. Extending the model to include more actions is not straightforward and is prone to mean collapse. Different strategies can possibly be exploited to increase the capacity of the architecture for more action types or other additional semantics. One possible solution is to increase the network capacity, though we expect this may make training more difficult. Another solution could be to train a separate network for each subset of attributes or actions, which may serve as a feasible solution since our network is relatively small (around 30MB). Finally, providing additional strong control signals such as body contact with the environment could serve to decrease the uncertainty in the motion generation phase and prevent it from collapsing to the mean pose. These control signals could be provided manually by the animator or by a separate network which is trained on the data [PFAG19, HKS17].

In our recurrent model, each cell is represented as a VAE conditioned on the previous internal state. However, VAEs are based on maximizing a log-likelihood lower bound which might give a suboptimal solution for the true log-likelihood. We also made a

Model	Actions				Average
	Walking	Jogging	Jumping	Lifting	
Quaternet [PFAG19]	9.31±0.25	9.45±0.22	6.27±0.31	5.87±0.28	7.73±0.26
ERD [FLFM15]	8.97±0.33	8.32±0.30	6.31±0.28	5.71±0.23	7.31±0.28
Proposed(SL)	9.15±0.24	9.26±0.22	8.43±0.37	8.62±0.18	8.87±0.25
Proposed(NL)	9.03±0.29	8.97±0.21	8.25±0.19	8.30±0.32	8.64±0.26
Proposed(NC)	9.32±0.27	9.42±0.30	8.95±0.31	8.92±0.34	9.15±0.27
Proposed	9.38±0.13	9.35±0.12	9.13±0.28	9.27±0.27	9.28±0.21
Real data	9.64±0.13	9.81±0.24	9.62±0.1	9.53±0.21	9.65±0.17

Table 3: *Results for qualitative evaluation. The results show the average scores assigned to a set of motions sampled from each action set and from each model, where 1 corresponds to completely unrealistic and 10 corresponds to completely realistic.*

strong assumption about the posterior distribution by modelling it as a standard isotropic Gaussian, which could increase the error in the posterior approximation. It is possible that normalizing flows [KPB19, PNR*19] could be exploited to improve this aspect of the model. The approximate posterior distribution could be parameterized by a network of normalizing flows which is applied to the output of the VAE decoder; this has been shown to provide a tighter lower bound for other applications [RM15]. Alternatively, the VAE module in our Motion Cell could potentially be replaced entirely with a conditional normalizing flow network in which the temporal dependencies are modelled by conditioning the prior on the previous internal state.

References

[ACC13] OSU ACCAD. ACCAD. .https://accad.osu.edu/research/motion-lab/system-data, May 2013. 14

[AF02] ARIKAN O., FORSYTH D. A.: Interactive motion generation from examples. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 483–490. 2

[AKH19] AKSAN E., KAUFMANN M., HILLIGES O.: Structured prediction helps 3d human motion modelling. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 7144–7153. 2, 3

[AWL*20] ABERMAN K., WENG Y., LISCHINSKI D., COHEN-OR D., CHEN B.: Unpaired motion style transfer from video to animation. *arXiv preprint arXiv:2005.05751* (2020). 3

[BKL18] BARSOUM E., KENDER J., LIU Z.: Hp-gan: Probabilistic 3d human motion prediction via gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2018), pp. 1418–1427. 3

[BVV*15] BOWMAN S. R., VILNIS L., VINYALS O., DAI A. M., JOZEFOWICZ R., BENGIO S.: Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* (2015). 7

[CKD*15] CHUNG J., KASTNER K., DINH L., GOEL K., COURVILLE A. C., BENGIO Y.: A recurrent latent variable model for sequential data. In *Advances in neural information processing systems* (2015), pp. 2980–2988. 2, 3, 5

[CVMG*14] CHO K., VAN MERRIËNBOER B., GULCEHRE C., BAHDANAU D., BOUGARES F., SCHWENK H., BENGIO Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014). 13

[DKB14] DINH L., KRUEGER D., BENGIO Y.: Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014). 3

[FLFM15] FRAGKIADAKI K., LEVINE S., FELSEN P., MALIK J.: Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 4346–4354. 1, 2, 7, 9, 10, 11, 13

[GMT*20] GHORBANI S., MAHDAVIANI K., THALER A., KORDING K., COOK D. J., BLOHM G., TROJE N. F.: Movi: A large multipurpose motion and video dataset. *arXiv preprint arXiv:2003.01888* (2020). 2, 8

[GSAH17] GHOSH P., SONG J., AKSAN E., HILLIGES O.: Learning human motion models for long-term predictions. In *2017 International Conference on 3D Vision (3DV)* (2017), IEEE, pp. 458–466. 9

[HAB19] HENTER G. E., ALEXANDERSON S., BESKOW J.: Moglow: Probabilistic and controllable motion synthesis using normalising flows. *arXiv preprint arXiv:1905.06598* (2019). 3

[HHS*17] HABIBIE I., HOLDEN D., SCHWARZ J., YEARSLEY J., KOMURA T., SAITO J., KUSAJIMA I., ZHAO X., CHOI M.-G., HU R., ET AL.: A recurrent variational autoencoder for human motion synthesis. In *BMVC* (2017). 3

[HKS17] HOLDEN D., KOMURA T., SAITO J.: Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–13. 1, 3, 7, 10

[HRU*17] HEUSEL M., RAMSAUER H., UNTERTHINER T., NESSLER B., HOCHREITER S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems* (2017), pp. 6626–6637. 2, 9

[HS97] HOCHREITER S., SCHMIDHUBER J.: Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780. 13

[HSK16] HOLDEN D., SAITO J., KOMURA T.: A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–11. 1, 3, 7

[Huy09] HUYNH D. Q.: Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision 35*, 2 (2009), 155–164. 7

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034. 8

[JZSS16] JAIN A., ZAMIR A. R., SAVARESE S., SAXENA A.: Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition* (2016), pp. 5308–5317. 1, 2, 3, 7, 9, 13

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 8

[KD18] KINGMA D. P., DHARIWAL P.: Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems* (2018), pp. 10215–10224. 3

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (ToG) 23*, 3 (2004), 559–568. 2

[KPB19] KOBYZEV I., PRINCE S., BRUBAKER M.: Normalizing flows: An introduction and review of current methods. *arXiv preprint arXiv:1908.09257* (2019). 11

[KW14] KINGMA D. P., WELLING M.: Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (2014), Bengio Y., LeCun Y., (Eds.). URL: http://arxiv.org/abs/1312.6114. 6, 13

[KW*19] KINGMA D. P., WELLING M., ET AL.: An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning 12*, 4 (2019), 307–392. 13

[LLL18] LEE K., LEE S., LEE J.: Interactive character animation by learning multi-objective control. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–10. 2

[LMR*15] LOPER M., MAHMOOD N., ROMERO J., PONS-MOLL G., BLACK M. J.: SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 34*, 6 (Oct. 2015), 248:1–248:16. 14

[LWH*12] LEVINE S., WANG J. M., HARAUX A., POPOVIĆ Z., KOLTUN V.: Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–10. 3

[LZCVDP20] LING H. Y., ZINNO F., CHENG G., VAN DE PANNE M.: Character controllers using motion vaes. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 40–1. 3, 5

[MBR17] MARTINEZ J., BLACK M. J., ROMERO J.: On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2891–2900. 1, 2, 3, 7, 9, 13

[MGT*19] MAHMOOD N., GHORBANI N., TROJE N. F., PONS-MOLL G., BLACK M. J.: Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 5442–5451. 2, 8

[MH08] MAATEN L. V. D., HINTON G.: Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605. 10

[MK05] MUKAI T., KURIYAMA S.: Geostatistical motion interpolation. In *ACM SIGGRAPH 2005 Papers*. 2005, pp. 1062–1070. 2

[Muk11] MUKAI T.: Motion rings for interactive gait synthesis. In *Symposium on Interactive 3D Graphics and Games* (2011), pp. 125–132. 2

[PALvdP18] PENG X. B., ABBEEL P., LEVINE S., VAN DE PANNE M.: Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 5

[PFAG19] PAVLLO D., FEICHTENHOFER C., AULI M., GRANGIER D.: Modeling human motion with quaternion-based neural networks. *International Journal of Computer Vision* (2019), 1–18. 1, 3, 6, 7, 8, 9, 10, 11, 13

[PNR*19] PAPAMAKARIOS G., NALISNICK E., REZENDE D. J., MOHAMED S., LAKSHMINARAYANAN B.: Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762* (2019). 11

[RM15] REZENDE D. J., MOHAMED S.: Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770* (2015). 3, 11

[SB06] SIGAL L., BLACK M. J.: Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. *Brown Univertsity TR 120* (2006). 2

[SCNW19] SMITH H. J., CAO C., NEFF M., WANG Y.: Efficient neural networks for real-time motion style transfer. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 2*, 2 (2019), 1–17. 3

[SGZ*16] SALIMANS T., GOODFELLOW I., ZAREMBA W., CHEUNG V., RADFORD A., CHEN X.: Improved techniques for training gans. In *Advances in neural information processing systems* (2016), pp. 2234–2242. 2, 9

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (ToG) 23*, 3 (2004), 514–521. 2

[SLY15] SOHN K., LEE H., YAN X.: Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems* (2015), pp. 3483–3491. 13

[SZKS19] STARKE S., ZHANG H., KOMURA T., SAITO J.: Neural state machine for character-scene interactions. *ACM Trans. Graph. 38*, 6 (2019), 209–1. 3

[SZKZ20] STARKE S., ZHAO Y., KOMURA T., ZAMAN K.: Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics 39* (06 2020). doi:10.1145/3386569.3392450. 3

[TH00] TANCO L. M., HILTON A.: Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings Workshop on Human Motion* (2000), IEEE, pp. 137–142. 2

[TH09] TAYLOR G. W., HINTON G. E.: Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning* (2009), pp. 1025–1032. 2

[THR07] TAYLOR G. W., HINTON G. E., ROWEIS S. T.: Modeling human motion using binary latent variables. In *Advances in neural information processing systems* (2007), pp. 1345–1352. 2, 7

[THR11] TAYLOR G. W., HINTON G. E., ROWEIS S. T.: Two distributed-state models for generating high-dimensional time series. *Journal of Machine Learning Research 12*, Mar (2011), 1025–1068. 2

[Tro02] TROJE N. F.: Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of vision 2*, 5 (2002), 2–2. 8

[WCX19] WANG Z., CHAI J., XIA S.: Combining recurrent neural networks and adversarial training for human motion synthesis and control. *IEEE Transactions on Visualization and Computer Graphics* (2019). 2

[WHF06] WANG J., HERTZMANN A., FLEET D. J.: Gaussian process dynamical models. In *Advances in neural information processing systems* (2006), pp. 1441–1448. 3

[WHSZ19] WANG H., HO E. S., SHUM H. P., ZHU Z.: Spatio-temporal manifold learning for human motions via long-horizon modeling. *IEEE Transactions on Visualization and Computer Graphics* (2019). 2

A Supplementary Material

Recurrent Neural Networks

Most of proposed methods [PFAG19, JZSS16, FLFM15, MBR17] for modelling human motion are based on recurrent neural networks (RNN). An RNN [CVMG*14, HS97] models data recursively by decomposing the probability distribution of the sequence over time

$$P_\theta(\mathbf{x}_1,\ldots,\mathbf{x}_T) = \prod_{t=2}^{T} P_\theta(\mathbf{x}_t|\mathbf{x}_{<t})\, P_\theta(\mathbf{x}_1), \tag{30}$$

where θ is the set of model parameters and $P_\theta(\mathbf{x}_1,\ldots,\mathbf{x}_T)$ is the likelihood of sequence $\mathbf{x}_{1:T}$. Each time step includes two main operations: updating the hidden internal state of the model which summarizes the past information, and a mapping from the hidden state to the next element in the sequence. Therefore, at each timestep we have

$$\mathbf{h}_t = f_h(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{31}$$

$$\begin{aligned} P_\theta(\mathbf{x}_{t+1}|\mathbf{x}_{\leq t}) &= P_\theta(\mathbf{x}_{t+1}|\mathbf{h}_t) \\ &= f_o(\mathbf{h}_t), \end{aligned} \tag{32}$$

where $\mathbf{h}_t$ is the internal hidden state and f_h is the non-linear updating function parameterized by θ. f_o is the mapping from internal state to the output and usually characterized by another network.

Variational Autoencoder

The Variational Autoencoder (VAE) [KW14] is a class of deep generative models which optimize a deep latent-variable model (DLVM) jointly with the inference model using a gradient-based optimizer. DLVM is a class of models with a simple prior $p_\theta(\mathbf{z})$ and complex marginal $p_\theta(\mathbf{x})$ and likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ distributions, where z is a latent variable and x is the observed variable. VAEs approximate the intractable posterior inference $p_\theta(\mathbf{z}|\mathbf{x})$ by introducing a parametric inference function $q_\phi(\mathbf{z}|\mathbf{x})$. Variational parameters ϕ and the generative parameters θ can be optimized jointly by maximizing the *evidence lower bound* (ELBO) defined as follows:

$$\begin{aligned} \mathcal{E}_{\theta,\phi}(\mathbf{x}) &= \mathbf{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \\ &= \mathbf{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right). \end{aligned} \tag{33}$$

The first term is the expected log-likelihood or reconstruction term which is estimated by Monte Carlo estimator and reparameterization trick [KW14]. The second term, $D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)$, is the Kullback-Leibler divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z})$ which acts as a specific regularization term [KW*19]. The combination of inference q_ϕ and generative p_θ models form a kind of autoencoder where the first term in Eq.33 is called the reconstruction term and the second term is called the regularization term. Vanilla VAEs suffer when the space of outputs is multi-modal, resulting in blurry generated samples. Sohn et al. [SLY15] proposed conditional VAE (CVAE) as an extension of VAE for learning structured output predictions simply by conditioning the generative process on an control variable. CVAE not only helps to address the problem of one-to-many mapping but also allows the model to control the generated sample class and/or characteristics by the control variable. The ELBO for CVAE is formulated as follows:

$$\mathcal{E}_{\theta,\phi}(\mathbf{x},\mathbf{a}) = \mathbf{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x},\mathbf{a})} \log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{a}) - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{a})\|p_\theta(\mathbf{z}|\mathbf{a})\right), \tag{34}$$

where $\mathbf{a}$ is the input variable.

RVAE Objective Function

Here we explain how the RVAE objective function in Eq. 15 is derived. We can consider the whole sequence as a single sample and write the ELBO similar to Eq. 34 as follows:

$$\mathcal{E} = \mathbf{E}_{q_\phi(\mathbf{z}_{\leq N}|\mathbf{w}_{\leq N},\mathbf{a}_{\leq N})} \log \frac{p_\theta(\mathbf{w}_{\leq N},\mathbf{z}_{\leq N}|\mathbf{a}_{\leq N})}{q_\phi(\mathbf{z}_{\leq N}|\mathbf{w}_{\leq N}.\mathbf{a}_{\leq N})} \tag{35}$$

By factorizing p_θ and q_ϕ across time (similar to 30) we have

$$\begin{aligned} \mathcal{E} &= \mathbf{E}_{q_\phi(\mathbf{z}_{\leq N}|\mathbf{w}_{\leq N},\mathbf{a}_{\leq N})} \log \frac{\prod_{n=1}^{N} p_\theta(\mathbf{w}_n,\mathbf{z}_n|\mathbf{w}_{<n},\mathbf{z}_{<n},\mathbf{a}_{\leq n})}{\prod_{n=1}^{N} q_\phi(\mathbf{z}_n|\mathbf{z}_{<n}\mathbf{w}_{<n}.\mathbf{a}_{\leq n})} \\ &= \mathbf{E}_{q_\phi(\mathbf{z}_{\leq N}|\mathbf{w}_{\leq N},\mathbf{a}_{\leq N})} \log \prod_{n=1}^{N} \frac{p_\theta(\mathbf{w}_n|\mathbf{w}_{<n},\mathbf{z}_{\leq n},\mathbf{a}_{\leq n}) p_\theta(\mathbf{z}_n|\mathbf{z}_{<n},\mathbf{a}_{\leq n})}{q_\phi(\mathbf{z}_n|\mathbf{z}_{<n}\mathbf{w}_{<n}.\mathbf{a}_{\leq n})} \\ &= \mathbf{E}_{q_\phi(\mathbf{z}_{\leq N}|\mathbf{w}_{\leq N},\mathbf{a}_{\leq N})} \Bigg[\sum_{n=1}^{N} \log p_\theta\left(\mathbf{w}_n|\mathbf{w}_{<n},\mathbf{z}_{\leq n},\mathbf{a}_{\leq n}\right) \\ &\quad - \lambda_{KL}\mathrm{KL}(q_\phi(\mathbf{z}_n|\mathbf{w}_{\leq n},\mathbf{z}_{<n},\mathbf{a}_{\leq n})\|p_\theta(\mathbf{z}_n|\mathbf{w}_{<n},\mathbf{z}_{<n},\mathbf{a}_{\leq n}))\Bigg], \end{aligned} \tag{36}$$

where the RVAE objective is the negative value of the above ELBO.

More Training and Synthesis Details

Scheduling Loss Coefficients

Figure 1 shows how we set the schedulers for loss coefficients in Eq. 14. We set an annealing scheduler for λ_{KL} to address the KL vanishing problem. We also set λ_{CL} to zero for the beginning of the training and then gradually increase it. These two strategies allow the model to focus more on capturing useful information for reconstruction during the initial epochs.

Hardware and Software

We implemented the framework using PyTorch library. We also used a single GeForce RTX 2080 Ti GPU for parallel computing both in training and synthesis.

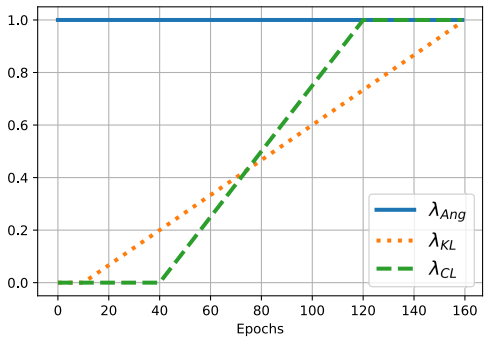


Figure 1: *Scheduling the coefficients for each term in Eq. 14 during training.*

Visualization and Sequence Samples

The visualization of motion sequences for conducting qualitative experiments was shown as stick figures. For the demo, we used Unity to visualize SMPL [LMR*15] model with average body shape for both females and males.

We present representative samples of the output of our model in Figure 2. You can see by comparing the generated samples (orange) to real samples (blue), our synthesized samples look very natural and from the same action cluster. For more samples please check out our video demo.

To show the ability of our framework in learning transitions between different actions, we also trained the network on ACCAD database [ACC13] which contains samples of transition between jogging and walking. The examples of generated samples are included in the video demo.

Figure 2: *Samples of real (blue) and synthetic (orange) motion sequences.*