A Recurrent Variational Autoencoder for Human Motion Synthesis

Ikhsanul Habibie
abie.ikhsan@gmail.com
Daniel Holden
contact@theorangeduck.com
Jonathan Schwarz
schwarzjn@gmail.com
Joe Yearsley
josephelliotyearsley@gmail.com
Taku Komura
tkomura@inf.ed.ac.uk

The University of Edinburgh
School of Informatics
Edinburgh, UK

Abstract

We propose a novel generative model of human motion that can be trained using a large motion capture dataset, and allows users to produce animations from high-level control signals. As previous architectures struggle to predict motions far into the future due to the inherent ambiguity, we argue that a user-provided control signal is desirable for animators and greatly reduces the predictive error for long sequences. Thus, we formulate a framework which explicitly introduces an encoding of control signals into a variational inference framework trained to learn the manifold of human motion. As part of this framework, we formulate a prior on the latent space, which allows us to generate high-quality motion without providing frames from an existing sequence. We further model the sequential nature of the task by combining samples from a variational approximation to the intractable posterior with the control signal through a recurrent neural network (RNN) that synthesizes the motion. We show that our system can predict the movements of the human body over long horizons more accurately than state-of-the-art methods. Finally, the design of our system considers practical use cases and thus provides a competitive approach to motion synthesis.

1 Introduction

There is a growing demand for models capable of learning from human motion capture (MOCAP) data for applications including computer graphics, human computer/robot interactions, psychological analysis and surveillance. The existence of several publicly available MOCAP datasets has fueled interest in the application of machine learning methods to human motion modelling. Traditional machine learning methods such as Hidden Markov Models or Linear Dynamical Systems however, struggle to model the highly complex, non-linear dependencies of human motion that arise from a large variety of factors that are nearly impossible to model individually [17].

Following the track record of success in both discriminative and generative modelling, various researchers across the community are starting to apply deep learning techniques to problems such as human action recognition and synthesis. Such methods overcome the non-linear nature of simpler models and can learn complex dependencies due to the existence of large datasets. Existing approaches to motion synthesis are for instance based on conditional Restricted Boltzmann Machines [17], apply one-dimensional convolutions along the time line [9, 10] or consider the tree structure of the human body [3, 5]. Other methods rely on recurrent neural networks (RNNs) to implement time series models which have feedback structures [6, 11]. Such autoregressive models are attractive in terms of their nature of producing postures along the timeline as it can imitate biological models and fit well to real-time applications such as computer games.

However, all of the above methods have difficulty in predicting actions more than just a handful of frames ahead in the future, resulting in the motion sequence gradually converging to a static pose. This phenomenon occurs as such models tend to average over the possible future poses in the next state. For example, considering a sequence where the human body may either turn to the left or right, the autoregressor will average out the possible future movements and produce a floating motion.

The main issue resulting in this undesirable behaviour is the lack of consideration of control signals such as rotational and translational velocity. Provided such control signals are available at prediction time, uncertainty in possible future poses can be greatly reduced. Furthermore, the specification of control signals is of crucial importance in practical motion synthesis applications where an animator will be required to specify the exact trajectory, pace and actions the character is to take.

Furthermore, some previously proposed methods (eg. [6]) must be initialized with frames from an existing sequence and continue to complete the remaining time series thereafter. This is problematic in particular for methods that are fully deterministic, and will hence produce the same sequence given equal initialization. In addition, these methods do not consider the stochastic nature of human motion. As the human body is a biological model that varies behavior even if the state and the input signals are the same, a generative model is more appropriate for modelling its behavior. By formulating a prior over latent variables of such a model, motion can be generated from samples of a prior distribution at test time, eliminating the risk of too much similarity to example sequences. This can for instance be useful in crowd animation, where a large number of characters is generated at once.

We close this gap by proposing a novel approach to motion synthesis that has been designed to fulfill these desiderata. We achieve this through a combination of variational inference, the consideration of a control signal and several deep learning modules designed to produce high-quality samples. Our experiments show that our model significantly reduces the predictive error for long sequences and generates realistic human motion. The user only needs to provide high level instructions, i.e., the trajectory of the body. Gives these signals, the system first initializes the hidden unit values by sampling, and then synthesizes the full body motion in an autoregressive manner.

In summary, our main contributions are as follows:

- A novel generative architecture for human motion data that achieves state-of-the-art results in sequence prediction.

- The consideration of an additional control signal that allows animators to control the pace and direction of the generated sequence.

- A demonstration that a variational inference framework is well suited for time series predictions.
- The release of a new MOCAP dataset that provides both control signals and joint positions for more than 1800 sequences.

2 Related Work

Data-driven motion synthesis and classification using neural networks is attracting researchers in both the computer animation and machine learning communities thanks to its high scalability and runtime efficiency. These techniques can be categorized into those based on 1D convolutional filters [3, 9, 10] and autoregressive time series models [6, 14, 16, 18].

The methods based on temporal convolutional filters borrow ideas from the CNN (Convolutional Neural Network) structures developed for image classification and synthesis. Since the body has different topological structure from the images, initial attempts only conduct convolutions along the timeline. Holden *et al.* [9] apply convolutional autoencoders to the motion for denoising and retrieval purposes. They also enhance the method for motion synthesis, and provide one of the first approaches that considered the use of a control signals, by regressing such a window to the full body motion through 1D CNNs [10].

Bütepage *et al.* [3] propose a framework to regress the motion of the previous N frames to predict the future N frames. They also use a convolutional network along the body structure and demonstrate this method helps to reproduce the original motion sequence, although the improvement is subtle due to the limited locality of the body structure.

For the purpose of modelling human motion behavior, and its application for real-time motion synthesis, autoregressive neural network models are attractive choices. Taylor *et al.* [16, 18] apply conditional Restricted Boltzmann Machines (cRBM) for synthesizing gait animation. Mittelman *et al.* [14] use the spike-and-slab version of the recurrent temporal RBM to improve reconstructions. Due to the nature of sampling at every frame, the motions synthesized by RBMs are very noisy which can sometimes result in divergence. Fragkiadaki *et al.* [6] propose the Encoder-Recurrent-Decoder (ERD) network that applies a Long short-term memory (LSTM) model in the hidden space for predicting the next pose of the body. Due to the ambiguity issue that is mentioned in the introduction, autoregressive models often suffer from convergence to an average pose. We present in our work that we can avoid such convergence by using a window of motions encoded by 1D convolutional filters, as well as control signals for controlling the human body.

3 Methodology

We now define and motivate the structure of the proposed model that we call the VAE-LSTM model. We first describe how the canonical framework of the variational autoencoder [13] can be extended to time-series data such as human motion data. We further show how the model can be conditioned on additional deterministic variables in Section 3.1.

Next, we describe our VAE-LSTM architecture, whose overview is shown in Fig. 1. The system is composed of an encoder (Fig. 1, bottom) and the autoregressive decoder (Fig. 1, top). For the encoding, following [9, 10] we use one-dimensional convolutions along the temporal dimension to efficiently encode control signals $C_{1:T}$ and well as joint positions

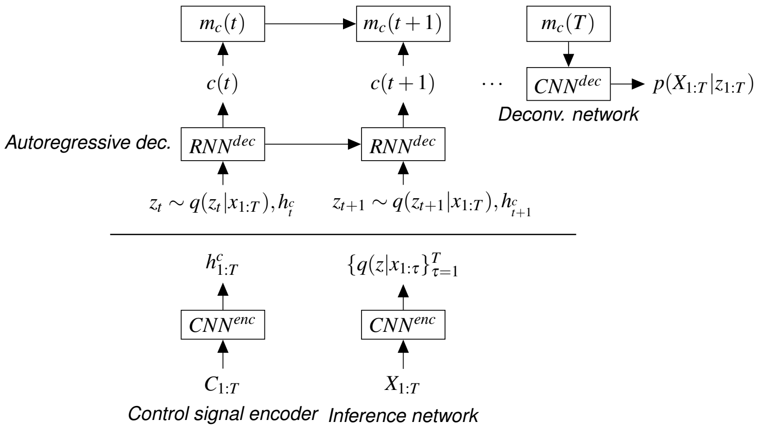

Figure 1: An overivew of the VAE-LSTM architecture. We assume a motion clip consists of $X_{1:T}$ joint positions and $C_{1:T}$ control signals.

$X_{1:T}$, using the latter encoding to paramterize a family of approximate posterior distributions. Importantly, we reduce the temporal dimensionality in the encoding process which increases computational efficiency both during training as well as in the final generative model. This process is described in Section 3.2. Given latent variables z_t and encoded control signals h^c_t, we adopt an autoregressive distribution through a Long short-term memory (LSTM) cell [8] to synthesize the motion at a reduced frame rate, resulting in a "motion canvas". A final convolution and upsampling module interpolates between the provided frames in the motion canvas resulting in high-quality motion $\hat{X}_{1:T}$. We elaborate these processes in section Section 3.3. Finally, we describe the stochastic motion generation process in section Section 3.4.

Throughout the paper, we denote sequences by $S_{1:T} = (s_1, \ldots, s_T)$ but may drop the subscript if this is clear. All lowercase symbols are vectors unless otherwise stated.

3.1 Variational Autoencoders

We now briefly review [13] which forms the core our proposed model. Consider a latent variable model in which we assume observed data x has underlying causal or latent factors z. While both prior $p(z)$ and likelihood $p(x|z)$ can be formulated exactly, the posterior $p(z|x) = \frac{p(x,z)}{\int p(x,z)dz}$ requires an intractable integral over the latent space. Instead, [13] consider a parametrized variational approximation $q_\phi(z|x)$ of the intractable posterior. We also assume the likelihood $p_\theta(x|z)$ is non-linear transformation of the latent variables parametrized by θ. This can be shown to result in a lower bound on the log marginal likelihood as derived through Jensen's inequality:

$$\log p(x) = log\mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right] \geq \mathbb{E}_{q_\phi(z|x)}\left[log\frac{p_\theta(x,z)}{q_\phi(z|x)}\right] = \mathcal{L}(x) \tag{1}$$

and can be equivalently written as:

$$\mathcal{L}(x) = -\mathcal{D}_{KL}(q_\phi(z|h)||p(z)) + \mathbb{E}_{q_\phi(z|h)}\left[logp(x|z)\right] \tag{2}$$

to intuitively understand this as trading off the Kullback-Leiber divergence $\mathcal{D}_{KL}$ with the data log-likelihood. For the rest of this paper we will assume a unit Gaussian prior and a diagonal Gaussian approximation of the posterior: $q(z|x) = \mathcal{N}(\mu(x;\phi), diag(\sigma^2(x;\phi)))$ where $\mu(x;\phi), \sigma^2(x;\phi)$ are deterministic transformations of the input. The introduced *reparametrization trick* allows the reformulation of the sampling process as the transformation of an auxiliary variable ε [13, 15]:

$$z = \sigma(x;\phi)\varepsilon + \mu(x;\phi); \varepsilon \sim \mathcal{N}(0, I) \tag{3}$$

which has the advantage that both parameters of the approximate posterior and likelihood ϕ, θ can be jointly optimized.

Extending this general framework to sequential data we draw a latent variable at each time step $z_t \sim q(z_t|x_{1:T})$ and assume a autoregressive generative model $p(x_t|x_{<t}, z_{<t})$ implemented by a recurrent neural network RNN^{dec} similar to [2]. As argued above, we also assume the existence of an encoding of a control signal h_t^c at each time step, so that the generative model takes the form $p(x_t|x_{<t}, z_{<t}, h_{<t}^c)$.

Instead of directly parameterizing this distribution by the output of RNN^{dec}, we instead take inspiration from [7] and subsequently concatenate these values to a motion canvas $m_c(t)$. The final motion sequence is then obtained by a final transformation of the canvas through a deconv. neural network $CNN^{dec}(m_c(T))$. We will motivate this choice below.

As we formulate a prior over the latent variables for each element in the sequence, the KL-divergence in Eq. (2) becomes a sum of divergences [7]:

$$\mathcal{L}_z = \sum_{t=1}^{T} \mathcal{D}_{KL}(q(z_t|x)||p(z_t)) \tag{4}$$

and $\mathcal{L}_x = \log p(X_{1:T}|z)$ is simply the negative log probability under a Gaussian distribution, the natural choice for real-valued data. We thus optimize:

$$\mathcal{L}(x) = \mathbb{E}_q[\mathcal{L}_z + \mathcal{L}_x] \tag{5}$$

through a single sample z_t each time step.

3.2 Encoding Networks

As a first step in the training procedure, we encode both joint positions and the provided control signals with two separate encoders. While we will assume very similar components, the encoding of the joint positions will result in parameters μ, σ^2 of the approximate posterior at every time step. The control signals on the contrary will be considered additional deterministic variables the decoder is conditioned on. We will thus refer to the modules "inference network" and "control signal encoder" respectively.

Inspired by [9, 10] we adopt one-dimensional convolutions to encode both joint positions and control signals. It is important to note that such non-recurrent modules in the inference network can only be used as we assume that the approximate posterior factorizes: $q_\phi(z_{1:T}|x) = \prod_{t=1}^{T} q_\phi(z_t|x)$. We make this choice as this results in a significant improvement in training efficiency. We will give further justification for this assumption in Section 3.3.

In particular, we denote the application of a convolutional encoder by CNN^{enc}, resulting in hidden representations $h_{1:T}^c$ of the control signal and parameters of the approximate posterior $(\{\mu(z_t;\phi), \sigma^2(z_t;\phi))\}_{t=1}^{T}$ respectively.

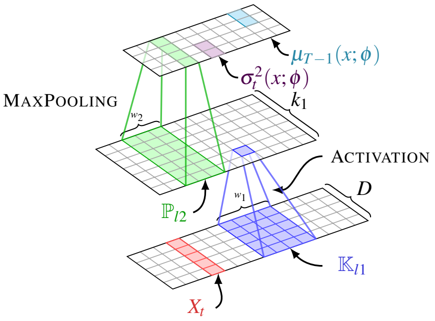

Figure 2: The convolutional encoder. After sliding k_1 1-D conv. kernels of size w_1 ($\mathbb{K}_{l1}$ is the receptive field) over the input motion, a maxpooling layer summarizes w_2 frames (resulting in a receptive field $\mathbb{P}_{l2}$) into a representation used to parameterize the posterior distribution. Figure inspired by [1].

Another important aspect of the convolutional encoder is the use of maxpooling layers, likewise along the temporal dimension. This reduces the effective temporal dimensionality of both joint positions and control signal. In addition to increased efficiency of the encoder modules, this also means that the recurrent decoder must only be run for as many time steps as there are left after the application of several pooling layers. Moreover, by summarizing several frames into a feature representation, we assume that every latent variable covers a longer time horizon which helps countering noise effects that may be introduced by the factorization assumption above. Note that we will not adopt our notation to take temporal pooling into consideration and continue denoting the time series as $1:T$.

Specifically, denoting the parameters of a two layer convolutional network by W_0, W_1, b_0 and b_1, and writing $\circledast, \alpha(\cdot), \Psi(\cdot)$ for convolution operation, a non-linear activation function and the max-pooling operator we have

$$CNN^{enc}(X) = \Psi(\alpha(\Psi(\alpha(X \circledast W_0 + b_0)) \circledast W_1 + b_1)) \tag{6}$$

where this can be easily generalized to multiple layers. Given the parametrization of $q_\phi(z|x)$ we then draw samples according to Eq. (3).

Fig. 2 shows a schematic illustration of the inference network using a single combination of convolution, activation function and maxpooling. The encoder of the control signals follows the same structure.

3.3 Autoregressive Decoder

Given samples $z_{1,\dots,T}$ and the encoding of the control signal $h^c_{1:T}$ we implement the autoregressive generative model RNN^{dec} through an LSTM layer. This recurrent models combines latent codes and the control signal encoding to produce the motion canvas $m_c(t)$ which will be used to synthesize the final sequence. In practice, we simply concatenate the LSTM's cell state up to time t to obtain $m_c(t)$.

While this module could be simple in principle, we found that a vanilla recurrent decoder had difficulty using information encoded in the latent variables. We argue that this may happen since previous frames and the control signal are strongly predictive of the next frame, so that the network may learn to ignore the latent code in the decoding process, confirming similar observations when modelling other forms of sequential data [19].

In order to overcome this issue, we instead set the cell state of the LSTM at every time step equal to the corresponding latent code. Note that this only happens during training. When novel motion is synthesized, we instead sample a single latent variable z_1 regardless of the desired length of the motion and use it to initialize the first cell state. This procedure has the following effects: Firstly, as the LSTM's hidden space is set equal to the latent code during training, it learns to operate in a similar space. Assuming the inference networks learns that the latent code corresponds to the manifold of valid human motion, the LSTM will move on this manifold during test, even though only an initial sample is provided. Secondly, the inference network is pressured to encode meaningful temporal dependencies in the latent code, further countering the effects of the conditional independence assumption. While we could also consider sampling a latent variable z_t for each frame at test time, we found both approaches to work equally well.

Finally, we use a modified version of the LSTM cell in [8]. In particular, we directly use the LSTM cell state in subsequent layers and make no use of an additional output regulated by a gate. Moreover, we make a small modification to the calculation of the cell state. We found this version to work well for our purposes. The modified LSTM equations become:

$$i(t) = s((W_{ic}[c_{t-1}] + b_{ic}) + (W_{ih}[h_t^c] + b_{ih})) \tag{7}$$

$$f(t) = s((W_{fc}[c_{t-1}] + b_{fc}) + (W_{fh}[h_t^c] + b_{fh})) \tag{8}$$

$$c(t) = \begin{cases} \alpha(f(t) \odot W_c[z_t] + i(t) \odot W_h[h_t^c]), & \text{during training} \\ \alpha(f(t) \odot W_c[c_{t-1}] + i(t) \odot W_h[h_t^c]), & \text{at test time} \end{cases} \tag{9}$$

where i, f, c are input- and forget gate and cell state of the LSTM respectively, $s(x) = \frac{1}{1+e^{-x}}$ denotes the sigmoid function and $\odot$ element-wise multiplication.

As a result of the maxpooling operation in the encoder, we can think of the resulting sequence as a motion sequence at a lower frame rate, hence the name motion canvas. In order to retrieve the original length of the sequence, we therefore require an additional upsampling module. This can simply be achieved through a deconvolutional network that corresponds to the inverse of the encoding network. Thus, this deconvolution module interpolates between the frames, therefore smoothing transitions in the motion.

3.4 Stochastic Motion Generation

Given a trained model and the control signal $C_{1:T}$ for T time steps, a new motion sequence $\hat{X}$ can be generated by sampling from the prior $z \sim \mathcal{N}(0, I)$ and executing the autoregressive decoder RNN^{dec} for each time step in the sequence, resulting in LSTM cell states $c(t)$. Finally, the motion canvas m_c, obtained be concatenating the LSTM cell states at each time step, is transformed into the final sequence. This procedure is summarized in Algorithm 1. We denote column-wise concatenation of matrices A, B by $[A||B]$.

4 Experiments

4.1 Data Preprocessing

The data used for this experiment consists of over 1800 data points of 60Hz human locomotion data, including walking, jogging and running that we captured internally due to the lack of suitable a MOCAP data set that provides a control signal. We make this data publicly

Algorithm 1: Stochastic Motion Generation with VAE-LSTM

```
Input  : Control signal $C_{1:T}$ for the entire sequence
Output: A generated MOCAP sequence $\hat{X}_{1:T}$
$h^c_{1:T} = CNN^{enc}(C_{1:T})$ // Control signal encoder
$z_1 \sim \mathcal{N}(0,I)$ // Prior sample
$c(1) = RNN^{dec}(z_1, h^c_1)$
$m_c(1) = c(1)$ // Motion canvas
for $t \leftarrow 2$ to $T$ do
    $c(t) = RNN^{dec}(c(t-1), h^c_t)$
    $m_c(t) = [m_c(t-1)||c_t]$ // Concatenate canvas
end
$\hat{X} = CNN^{dec}(m_c(T))$// Deconvolutional network
```

Model	Initial frames	Control signal	Predictive error at Frame 70	90	120	160	210
ERD [6]	60	×	0.66	2.10	6.13	13.53	19.49
LSTM-3LR [6]	60	×	**0.55**	3.18	9.47	14.14	17.98
cRBM, CD-1 [17]	60	×	1.41	3.51	7.96	11.51	15.77
cRBM, CD-15 [17]	60	×	1.41	3.71	8.28	10.54	15.79
VAE-LSTM	4	✓	1.45	**1.75**	**2.20**	**3.25**	**4.12**
VAE-LSTM (prior sample)	0	✓	1.97	2.37	2.91	3.97	4.91

Table 1: Average motion prediction error on the test set comparing ground truth joint positions to generated motions. While competing approaches are trained to predict future control signals, we provide the ground truth control signals to our method. Note that providing 4 initial frames to our method allows us to sample the initial latent variable from the approximate posterior, while we resign to sampling from the prior $p(z)$ when no initial frame is provided. Our method performs significantly better in both cases.

available to encourage future research [1]. Every frame of a sequence is represented by the 21 joint positions in the local Cartesian space defined with its origin at the hip of the character. We consider the turning speed, forward velocity and sideways velocity of the body as additional scalar control signals that our model is designed to encode. Thus, a single frame has 66 degrees of freedom.

During training, we represent a single example as a clip of 240 frames (= four seconds). As the recorded data includes sequences of varying length, we split longer sequences into sub sequences and pad individual sequences with the first and final pose if necessary.

4.2 Model specification and training

We now give details about the training procedure and specify all design choices of the architecture. Both encoders consist of two blocks of conv., activation and maxpooling respectively. The conv. filter sizes are (3, 25)*64, (64, 25)*128 for the control signal encoder and (63, 25)*64, (64, 25)*256 for the inference network. Thus the receptive field of each kernel

[1]https://bitbucket.org/jonathan-schwarz/edinburgh_locomotion_mocap_dataset

Model	Initial frames	Control signal	Pose error at Frame 70	90	120	160	210
ERD [6]	4	✓	2.31	2.62	2.54	2.99	2.63
LSTM-3LR [6]	4	✓	2.11	2.15	2.36	2.34	2.17
cRBM, CD-1 [17]	4	✓	2.76	2.81	3.13	2.82	3.01
cRBM, CD-15 [17]	4	✓	2.64	2.71	3.08	2.85	3.21
VAE-LSTM	4	✓	**0.82**	**0.85**	**0.82**	**0.83**	**0.87**
VAE-LSTM (prior sample)	0	✓	1.23	1.27	1.24	1.24	1.29

Table 2: Average pose prediction error on the test set. We provide ground truth control signals to all methods. Note that in contrast to Table 1 the values shown measure the error relative to the root of the character, ignoring both translation and rotation.

spans roughly 0.4 seconds of motion. Both pooling window and stride are of size 1, thus reducing the dimensionality from 240 to 60 frames. In addition, we add dropout on both the joint positions and control signals to reduce overfitting. Throughout the model we adopt the non-linear exponential linear unit (ELU) as activation function [4].

We use 128 units in the LSTM layer which results in a motion canvas of size (128, 60). The deconvolution layers consist of (128, 25)*128 and (66, 25)*128 convolutions, activation and upsampling operations.

We train our model for 5000 epochs using the Adam optimizer [12] with a learning rate of 5×10^{-4}. We optimize the loss in Eq. (5), therefore maximizing the variational lower bound in Eq. (1). Finally, we also add $\ell_1 = 0.1$ regularization to decrease overfitting.

4.3 Motion generation

We now provide a quantitative evaluation of the models by measuring the mean squared error between generated joint positions and ground truth motion on the held-out test set consisting of 20 sequences.

As many proposed methods do not explicitly consider a control signal, we adopt these models to jointly predict the future pose along with the control signal and consider their performance using either provided control signals or model's predictions at test time.

Note that we distinguish between motion and pose prediction error to provide a fair comparison in both scenarios. In particular, we adopt the latter metric when control signals are provided to all methods, as sequence predictions will follow a given trajectory. Thus, we ignore the provided translation and rotation of the character and calculate the error relative to the root of the character in the test set. On the contrary, we simply define motion prediction error as the mean-squared error between the joint positions of the characters taking into account above metrics. This allows us to demonstrate both the importance of considering control signals as well as the suitability of our method to make optimal use of these values.

We test our method in two scenarios: Firstly, by providing a sufficed number of initial frames to calculate the parameters of the approximate posterior at time $t = 1$. As we use two max pooling layers in the encoder, this corresponds to 4 initial frames. In addition, we demonstrate that our method continues to produce high quality motion when no frames are provided (corresponding to the procedure in Algorithm 1). In both cases, the decoding process is conditioned on a single latent variable. In addition, our method uses the ground-truth control signal throughout. This is a realistic assumption as an animator will typically

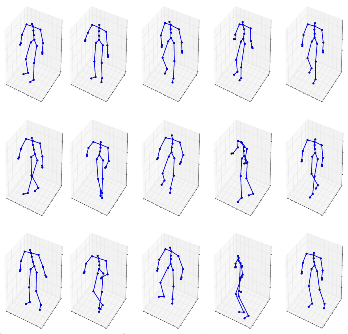
Figure 3: (left to right) Ground truth sequence and the generated motions produced by LSTM-3LR [6], ERD [6], CRBM [17] and VAE-LSTM at (top to bottom) frame 80, 100 and 120. The Figure correspond to the results in Table 1.

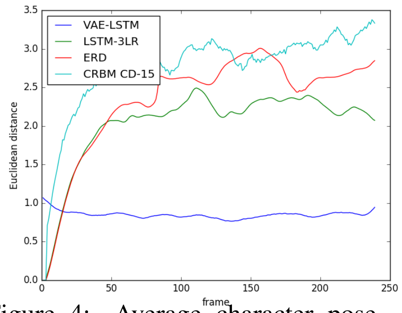

Figure 4: Average character pose prediction error on the test set. For each frame predicted by LSTM-3LR, ERD and CRBM, we replace predicted control signals with the ground truth. The Figure correspond to the results in Table 2.

provide a control signal for the entire motion to the generative model.

Table 1 shows the motion prediction error for all models. As no control signal is provided to LSTM-3LR, ERD and CRBM, we provide a large number of initial frames to make the comparison fairer. We see that our method significantly reduces the predictive error for long sequences. This demonstrates the importance of considering control signals in motion generation. We also observe that our models retains predictive accuracy when no initial frames are provided, justifying the variational inference approach.

We show typical behavior of the models in an example sequence in Fig. 3. We observe that character predicted by our method produces a realistic pose while other methods quickly tend to change style or direction.

Fig. 4 and Table 2 show the pose error when ground truth control signals are provided to all methods. The results shows that our method continues to outperform other approaches. This justifies the consideration these values throughout the design of our method.

5 Discussion

In this paper, we propose a novel approach to human motion synthesis that greatly reduces the predictive error for long sequences, thanks to the consideration of a control signal. In addition, the formulation within a variational inference framework allows us to generate novel motion without having to provide initial frames from an existing sequence. So far, we have shown that we achieve state-of-the-art results in motion synthesis when considering locomotive motions such as walking and running. In cases where an animated character is to perform additional actions that can not be straight-forwardly encoded in the presented framework (e.g. jumping, boxing), it might be necessary to consider further additions to the model. We believe this will be an interesting direction for future research.

References

[1] Miltiadis Allamanis, Hao Peng, and Charles Sutton. A convolutional attention network for extreme summarization of source code. *arXiv preprint arXiv:1602.03001*, 2016.

[2] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[3] Judith Bütepage, Michael Black, Danica Kragic, and Hedvig Kjellström. Deep representation learning for human motion prediction and classification. *arXiv preprint arXiv:1702.07486*, 2017.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[5] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proc. IEEE CVPR*, 2015.

[6] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proc. ICCV*, pages 4346–4354, 2015.

[7] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, pages 18:1–18:4, 2015. ISBN 978-1-4503-3930-8.

[10] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Trans on Graph*, 35(4), 2016.

[11] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[13] Diederik Kingma and Max Welling. Auto-encoding variational bayes. In *The 2nd International Conference on Learning Representations (ICLR)*, 2013.

[14] Roni Mittelman, Benjamin Kuipers, Silvio Savarese, and Honglak Lee. Structured recurrent temporal restricted boltzmann machines. In *Proc. ICML*, pages 1647–1655, 2014.

[15] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[16] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proc. ICML*, pages 1025–1032. ACM, 2009.

[17] Graham W. Taylor, Geoffrey E. Hinton, and Sam Roweis. Modeling human motion using binary latent variables. In *Proc. NIPS*, page 2007. MIT Press, 2006.

[18] Graham W Taylor, Geoffrey E Hinton, and Sam Roweis. Two distributed-state models for generating high-dimensional time series. *The Journal of Machine Learning Research*, 12:1025–1068, 2011.

[19] Weidi Xi, Haoze Sun, Chao Deng, and Ying Tan. Semi-supervised variational autoencoders for sequence classification. *arXiv preprint arXiv:1509.00519*, 2016.