Contextually Plausible and Diverse 3D Human Motion Prediction (Supplementary Material)

Sadegh Aliakbarian* Microsoft
Fatemeh Saleh ACRV, ANU
Lars Petersson Data61, CSIRO
Stephen Gould ACRV, ANU
Mathieu Salzmann CVLab, EPFL

1. LCP-VAE Architecture

Our motion prediction model follows the architecture depicted in Fig. 3 of the main paper. Below, we describe the architecture of each component in our model. Note that human poses, consisting of 32 joints in case of the Human3.6M dataset, are represented in 4D quaternion space. Thus, each pose at each time-step is represented with a vector of size 1×128. All the tensor sizes described below ignore the mini-batch dimension for simplicity.

The **observed motion encoder**, or `CS-VAE` motion encoder, is a single layer GRU [1] network with 1024 hidden units. If the observation sequence has length T_{obs}, the observed motion encoder maps $T_{obs} \times 128$ into a single hidden representation of size 1×1024, i.e., the hidden state of the last time-step. This hidden state, h_t, acts as the condition to the `LCP-VAE` encoder and the direct input to the `CS-VAE` encoder.

`CS-VAE`, similarly to any variational autoencoder, has an encoder and a decoder. The `CS-VAE` encoder is a fully-connected network with ReLU non-linearities, mapping the hidden state of the motion encoder, i.e., h_t, to an embedding of size 1×512. Then, to generate the mean and standard deviation vectors, we use two fully connected branches. They map the embedding of size 1×512 to a mean vector of size 1×128 and a standard deviation vector of size 1×128, where 128 is the length of the latent variable. Note that we apply a ReLU non-linearity to the vector of standard deviations to ensure that it is non-negative. We then use the reparameterization trick [7] to sample a latent variable of size 1×128. The `CS-VAE` decoder consists of multiple fully-connected layers, mapping the latent variable to a variable of size 1×1024, acting as the initial hidden state of the observed motion decoder. Note that we apply a Tanh non-linearity to the generated hidden state to mimic the properties of a GRU hidden state.

The **observed motion decoder**, or `CS-VAE` motion decoder, is similar to its motion encoder, except for the fact that it reconstructs the motion autoregressively. Additionally, it is initialized with the reconstructed hidden state, i.e., the output of the `CS-VAE` decoder. The output of each GRU cell at each time-step is then fed to a fully-connected layer, mapping the GRU output to a vector of size 1×128, which represents a human pose with 32 joints in 4D quaternion space. To decode the motions, we use a teacher forcing technique [10] during training. At each time-step, the network chooses with probability P_{tf} whether to use its own output at the previous time-step or the ground-truth pose as input. We initialize $P_{tf} = 1$, and decrease it linearly at each training epoch such that, after a certain number of epochs, the model becomes completely autoregressive, i.e., uses only its own output as input to the next time-step. Note that, at test time, the motions are generated completely autoregressively, i.e., with $P_{tf} = 0$.

Note that the future motion encoder and decoder have exactly the same architectures as the observed motion ones. The only difference is their input, where the future motion is represented by poses from T_{obs} to T_{end} in a sequence. In the following, we describe the architecture of `LCP-VAE` for motion prediction.

`LCP-VAE` is a conditional variational encoder. Its encoder's input is a representation of future motion, i.e., the last hidden state of the future motion encoder, h_T, conditioned on h_t. The conditioning is done by concatenation, thus the input to the encoder is a representation of size 1×2048. The `LCP-VAE` encoder, similarly to the `CS-VAE` encoder, maps its input representation to an embedding of size 1×512. Then, to generate the mean and standard deviation vectors, we use two fully connected branches, mapping the embedding of size 1×512 to a mean vector of size 1×128 and a standard deviation vector of size 1×128, where 128 is the length of the latent variable. Note that we apply a ReLU non-linearity to the vector of standard deviations to ensure that it is non-negative. To sample the latent variable, we use our extended reparameterization trick, explained in the main paper. This unifies the conditioning and sampling of the latent variable. Then, similarly to `CS-VAE`, the latent variable is fed to the `LCP-VAE` decoder, which is a fully connected network that maps the latent representation of size 1×128 to a reconstructed hidden state of size 1×1024 for future motion prediction. Note that we apply

*Work done while at the Australian National University.

a Tanh non-linearity to the generated hidden state to mimic the properties of a GRU hidden state.

To train our model, we use Adam optimizer with learning rate of 0.005 and mini-batch size of 128. We train our model for 100 epochs on a single NVIDIA GTX 2080Ti.

2. Pseudo-code for LCP-VAE

Here, we provide the forward pass pseudo-code for both `CS-VAE` and `LCP-VAE`.

Algorithm 1 A forward pass of `CS-VAE`

procedure `CS-VAE`($condition$) ▷ Human motion up to time t or source text
  h_t = `EncodeCondition`(x_t) ▷ Observed motion/source text encoder
  μ_c, σ_c = `CS-VAE.Encode`(h_t)
  `Sample` $\epsilon \sim \mathcal{N}(0, I)$ ▷ Sample from standard Gaussian
  $z_c = \mu_c + \sigma_c \odot \epsilon$ ▷ Reparameterization
  $\hat{h}_t$ = `CS-VAE.Decode`(z_c)
  $\hat{x}_t$ = `DecodeCondition`($\hat{h}_t$, `seed`)
  return $\hat{x}_t, \mu_c, \sigma_c, h_t, z_c$

Algorithm 2 A forward pass of `LCP-VAE`

procedure `LCP-VAE`(x_T, z_c, h_t) ▷ Human motion from t to T or target text
  if `isTraining` **then**
    h_T = `EncodeData`(x_T) ▷ Future motion/target sentence encoder
    h_{Tt} = `Concatenate`(h_T, h_t)
    μ, σ = `LCP-VAE.Encode`(h_{Tt})
    $z = \mu + \sigma \odot z_c$ ▷ Our extended reparameterization
  else
    $z = z_c$
  $\hat{h}_T$ = `LCP-VAE.Decode`(z)
  $\hat{x}_T$ = `DecodeData`($\hat{h}_T$, `seed`)
  return $\hat{x}_T, \mu, \sigma$

3. Derivation of LCP-VAE's KL Divergence Loss

In our approach, the model encourages the posterior of `LCP-VAE` to be close to the one of `CS-VAE`. In general, the KL divergence between two distributions P_1 and P_2 is defined as

$$\mathcal{D}_{KL}(P_1||P_2) = \mathbb{E}_{P_1}\left[\log \frac{P_1}{P_2}\right]. \tag{1}$$

Let us now consider the case where the distributions are multivariate Gaussians $\mathcal{N}(\mu, \Sigma)$ in $\mathbb{R}^d$, where $\Sigma = \text{diag}(\sigma^2)$, with σ and μ are d-dimensional vectors predicted by the encoder network of the VAE. The density function of such a distribution is

$$p(x) = \frac{1}{(2\pi)^{\frac{d}{2}} det(\Sigma)^{\frac{1}{2}}} exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big). \tag{2}$$

Thus, the KL divergence between two multivariate Gaussians is computed as

$$\begin{aligned}
&\mathcal{D}_{KL}(P_1||P_2) \\
&= \frac{1}{2}\mathbb{E}_{P_1}\Big[-\log\det\Sigma_1 - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \\
&\log\det\Sigma_2 + (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\Big] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \frac{1}{2}\mathbb{E}_{P_1}\Big[-(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \\
&(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\Big] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \frac{1}{2}\mathbb{E}_{P_1}\Big[-tr\{\Sigma_1^{-1}(x-\mu_1)(x-\mu_1)^T\} + \\
&tr\{\Sigma_2^{-1}(x-\mu_2)(x-\mu_2)^T\}\Big] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \frac{1}{2}\mathbb{E}_{P_1}\Big[-tr\{\Sigma_1^{-1}\Sigma_1\} + \\
&tr\{\Sigma_2^{-1}(xx^T - 2x\mu_2^T + \mu_2\mu_2^T)\}\Big] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} - \frac{1}{2}d + \frac{1}{2}tr\{\Sigma_2^{-1}(\Sigma_1 + \mu_1\mu_1^T - \\
&2\mu_2\mu_1^T + \mu_2\mu_2^T)\} \\
&= \frac{1}{2}\Big[\log\frac{\det\Sigma_2}{\det\Sigma_1} - d + tr\{\Sigma_2^{-1}\Sigma_1\} + tr\{\mu_1^T\Sigma_2^{-1}\mu_1 - \\
&2\mu_1^T\Sigma_2^{-1}\mu_2 + \mu_2^T\Sigma_2^{-1}\mu_2\}\Big] \\
&= \frac{1}{2}\Big[\log\frac{|\Sigma_2|}{|\Sigma_1|} - d + tr\{\Sigma_2^{-1}\Sigma_1\} + \\
&(\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1)\Big].
\end{aligned} \tag{3}$$

where $tr\{\cdot\}$ denotes the trace operator. In Eq. 3, the covariance matrix Σ_1 and mean μ_1 correspond to distribution P_1 and the covariance matrix Σ_2 and mean μ_2 correspond to distribution P_2.

Given this result, we can then compute the KL divergence of the `LCP-VAE` and the posterior distribution with mean $\mu + \sigma \odot \mu_c$ and covariance matrix $\text{diag}((\sigma \odot \sigma_c)^2)$. Let $\Sigma = \text{diag}(\sigma^2)$, $\Sigma_c = \text{diag}(\sigma_c^2)$, and d be the dimensionality of the latent space. The loss in Eq. 7 of the main paper can then be written as

$$\begin{aligned}
\mathcal{L}_{prior}^{\texttt{LCP-VAE}} = -\frac{1}{2}\Big[&\log\frac{|\Sigma_c|}{|\Sigma_c||\Sigma|} - d + tr\{\Sigma_c^{-1}\Sigma_c\Sigma\} + \\
&(\mu_c - (\mu + \Sigma\mu_c))^T\Sigma_c^{-1}(\mu_c - (\mu + \Sigma\mu_c))\Big].
\end{aligned} \tag{4}$$

Since $\Sigma_c^{-1}\Sigma_c = I$, $|\Sigma_c|$ will be cancelled out in the log term, which yields

$$\mathcal{L}_{prior}^{\texttt{LCP-VAE}} = -\frac{1}{2}\Big[\log\frac{1}{|\Sigma|} - d + tr\{\Sigma\} + (\mu_c - (\mu + \Sigma\mu_c))^T\Sigma_c^{-1}(\mu_c - (\mu + \Sigma\mu_c))\Big]. \quad (5)$$

4. Results on the Penn Action Dataset

As a complementary experiment, we evaluate our approach on the Penn Action dataset, which contains 2326 sequences of 15 different actions, where for each person, 13 joints are annotated in 2D space. Most sequences have less than 50 frames and the task is to generate the next 35 frames given the first 15. Results are provided in Table 1, where we compare our approach with the deterministic autoregressive (AR) counterpart. Note that the upper bound for the Context metric is 0.74, i.e., the classification performance given the Penn Action ground-truth motions.

Table 1. Quantitative evaluation on the Penn Action dataset. Note that a diversity of 1.21 is reasonably high for normalized 2D joint positions, i.e., values between 0 and 1, normalized with the width and the height of the image.

Method	Test MSE (KL) (Reconstructed)	Diversity (Sampled)	Quality (Sampled)	Context (Sampled)
LCP-VAE	0.034 (6.07)	1.21	0.46	0.70
AR Counterpart	0.048 (N/A)	0.00	0.46	0.51

5. Evaluating Sampling Quality

In Table 2, we compare our approach with the state-of-the-art deterministic motion prediction models [9, 6, 5, 2, 4] using the MAE metric in Euler space. To have a fair comparison, we generate one motion per observation by setting the latent variable to the distribution mode, i.e., $z = \mu_c$. This allows us to generate a plausible motion without having access to the ground truth. To compare against the deterministic baselines, we follow the standard setting, and thus use 50 frames (i.e., 2sec) as observation to generate the next 25 frames (i.e., 1sec). Surprisingly, despite having a very simple motion decoder architecture (one-layer GRU network) with a very simple reconstruction loss function (MSE), this motion-from-mode strategy yields results that are competitive with those of the baselines that use sophisticated architectures and advanced loss functions. We argue that learning a good, context-preserving latent representation of human motion is the contributing factor to the success of our approach. This, however, could be used in conjunction with sophisticated motion decoders and reconstruction losses, which we leave for future research.

Table 2. Comparison with the state-of-the-art deterministic models for 4 actions of Human3.6M. Note that in our approach, we use $z = \mu_c$ to generate a single motion.

	Walking						Eating					
Method	80	160	320	400	560	1000	80	160	320	400	560	1000
Zero Velocity	0.39	0.86	0.99	1.15	1.35	1.32	0.27	0.48	0.73	0.86	1.04	1.38
LSTM-3LR [2]	1.18	1.50	1.67	1.76	1.81	2.20	1.36	1.79	2.29	2.42	2.49	2.82
SRNN [6]	1.08	1.34	1.60	1.80	1.90	2.13	1.35	1.71	2.12	2.21	2.28	2.58
DAE-LSTM [3]	1.00	1.11	1.39	1.48	1.55	1.39	1.31	1.49	1.86	1.89	1.76	2.01
GRU [9]	0.28	0.49	0.72	0.81	0.93	1.03	0.23	0.39	0.62	0.76	0.95	1.08
AGED [4]	0.22	0.36	0.55	0.67	0.78	0.91	0.17	0.28	0.51	0.64	0.86	0.93
DCT-GCN [8]	**0.18**	**0.31**	0.49	0.56	0.65	**0.67**	**0.16**	0.29	0.50	0.62	0.76	1.12
LCP-VAE	0.20	0.34	**0.48**	**0.53**	**0.57**	0.71	0.20	**0.26**	**0.44**	**0.52**	**0.61**	**0.92**
	Smoking						Discussion					
Method	80	160	320	400	560	1000	80	160	320	400	560	1000
Zero Velocity	0.26	0.48	0.97	0.95	1.02	1.69	0.31	0.67	0.94	1.04	1.41	1.96
LSTM-3LR [2]	2.05	2.34	3.10	3.18	3.24	3.42	2.25	2.33	2.45	2.46	2.48	2.93
SRNN [6]	1.90	2.30	2.90	3.10	3.21	3.23	1.67	2.03	2.20	2.31	2.39	2.43
DAE-LSTM [3]	0.92	1.03	1.15	1.25	1.38	1.77	1.11	1.20	1.38	1.42	1.53	1.73
GRU [9]	0.33	0.61	1.05	1.15	1.25	1.50	0.31	0.68	1.01	1.09	1.43	1.69
AGED [4]	0.27	0.43	0.82	0.84	1.06	1.21	0.27	0.56	**0.76**	0.83	1.25	1.30
DCT-GCN [8]	0.22	**0.41**	0.86	0.80	0.87	1.57	**0.20**	**0.51**	0.77	0.85	1.33	1.70
LCP-VAE	**0.21**	0.43	**0.79**	**0.79**	**0.77**	**1.15**	0.22	0.55	0.79	**0.81**	**1.05**	**1.28**

6. Additional Qualitative Results

Here, we provide qualitative results on diverse human motion prediction on the Human3.6M dataset. As can be seen in Figures 1 to 6, the motions generated by our approach are diverse and natural, and mostly within the context of the observed motion. We also provide qualitative results as a video in a separate file.

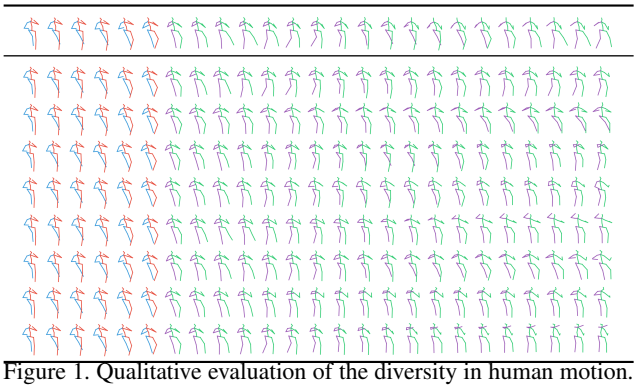

Figure 1. Qualitative evaluation of the diversity in human motion. The first row illustrates the ground-truth motion. The first six poses of each row depict the observation (the condition) and the rest are sampled from our model. Each row is a randomly sampled motion (not cherry picked). As can be seen, all sampled motions are natural, with a smooth transition from the observed to the generated ones. The diversity increases as we increase the sequence length.

References

[1] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 1

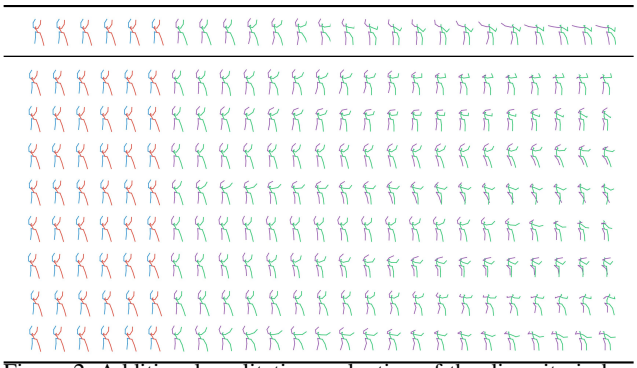

Figure 2. Additional qualitative evaluation of the diversity in human motion.

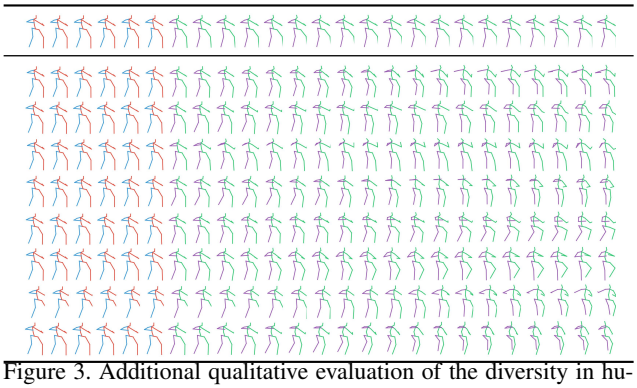

Figure 3. Additional qualitative evaluation of the diversity in human motion.

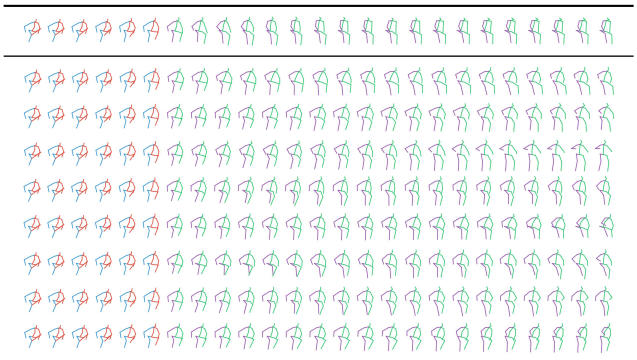

Figure 4. Additional qualitative evaluation of the diversity in human motion.

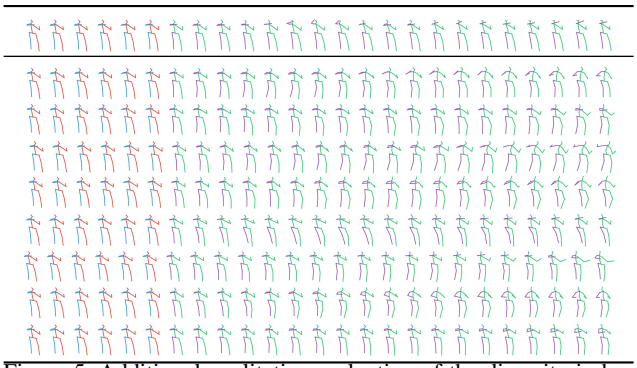

Figure 5. Additional qualitative evaluation of the diversity in human motion.

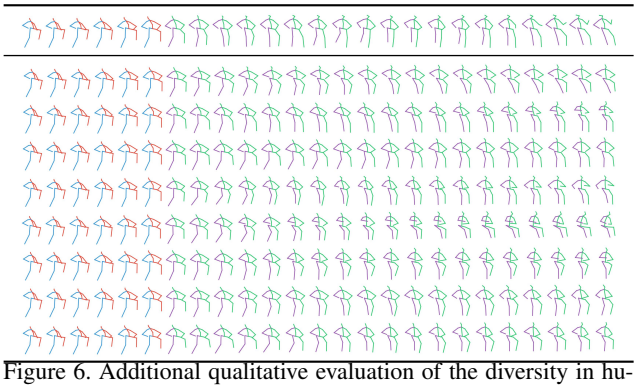

Figure 6. Additional qualitative evaluation of the diversity in human motion.

[2] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015. 3

[3] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. Learning human motion models for long-term predictions. In *2017 International Conference on 3D Vision (3DV)*, pages 458–466. IEEE, 2017. 3

[4] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José MF Moura. Adversarial geometry-aware human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 786–803, 2018. 3

[5] Liang-Yan Gui, Yu-Xiong Wang, Deva Ramanan, and José MF Moura. Few-shot human motion prediction via meta-learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 432–450, 2018. 3

[6] 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. 3

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[8] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *ICCV*, 2019. 3

[9] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4674–4683. IEEE, 2017. 3

[10] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 1