Pose-Centric Motion Synthesis Through Adaptive Instance Normalization

Oliver Hixon-Fisher[a], Jarek Francik[b] and Dimitrios Makris[c]
Kingston University, London, U.K.
{k2052803, jarek,d.makris}@kingston.ac.uk

Keywords: Motion Synthesis, Variational Auto-Encoder, Generative Adversarial Network, Adaptive Instance Normalization.

Abstract: In pose-centric motion synthesis, existing methods often depend heavily on architecture-specific mechanisms to comprehend temporal dependencies. This paper addresses this challenge by introducing the use of adaptive instance normalization layers to capture temporal coherence within pose-centric motion synthesis. We demonstrate the effectiveness of our contribution through state-of-the-art performance in terms of Fréchet Inception Distance (FID) and comparable diversity scores. Evaluations conducted on the CMU MoCap and the HumanAct12 datasets showcase our method's ability to generate plausible and high-quality motion sequences, underscoring its potential for diverse applications in motion synthesis.

1 INTRODUCTION

Human motion synthesis is the process of generating realistic motion for a humanoid skeleton. This capability is crucial for applications requiring realistic character animations, such as animated films, virtual avatars, or controlling non-player characters in computer games. To meet such applications human motion synthesis must meet several requirements, including the ability to synthesize poses in real-time, ensuring the animations are realistic and adapt to various guided or unguided scenarios.

Existing motion synthesis systems adopt widely different approaches (Mourot et al., 2021). To reflect how data is utilised, we propose a rough classification into two categories, each with distinct domains of application and specific challenges. Sequence-centric methods synthesize an entire sequence at once (Raab et al., 2024; Li et al., 2022; Tulyakov et al., 2017; Petrovich et al., 2021; Tevet et al., 2022; Zhang et al., 2023), whereas pose-centric methods build a sequence by iteratively adding one pose at a time (Guo et al., 2020; Mao et al., 2024; Xu et al., 2022; Luo et al., 2020; Hassan et al., 2021). Consequently, pose-centric and sequence-centric systems handle aspects of a motion sequence, such as temporal coherence, in different ways. In sequence-centric approaches, temporal coherence is explicitly modelled through learning the motion dynamics across the entire sequence. In contrast, pose-centric solutions address the same problem by presenting a method which allows future poses to be conditioned on the previous pose. This normally requires the application of an individual methodology for each solution, depending on its underlying architecture. On the other hand, the dependency on only two poses to maintain temporal coherence makes pose-centric solutions much more suitable for real-time applications, such as computer games, compared to sequence-based systems.

This paper concentrates on unconditional pose-centric solutions and proposes a novel approach for modelling temporal coherence based on the Adaptive Instance Normalisation (AdaIN), a method initially proposed in (Huang and Belongie, 2017) for neural style transfer between images and later used for motion style transfer (Wang et al., 2020). In our work, AdaIN ensures the transfer of temporal coherence from one pose to another, resulting in smooth motion synthesis. This approach addresses the limitations of existing pose-centric methods, which typically rely on architecture-specific techniques to maintain temporal coherence, leading to bias in the architectures used. We demonstrate that this approach can be applied to various architectures by applying it to both a VAE-based (Guo et al., 2020) and a normalizing flow (Wen et al., 2021) approaches.

[a] https://orcid.org/0009-0009-3157-4000
[b] https://orcid.org/0009-0003-8927-2802
[c] https://orcid.org/0000-0001-6170-0236

Hixon-Fisher, O., Francik, J. and Makris, D.
Pose-Centric Motion Synthesis Through Adaptive Instance Normalization.
DOI: 10.5220/0013102100003912
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025) - Volume 2: VISAPP*, pages 39-47
ISBN: 978-989-758-728-3; ISSN: 2184-4321
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

2 RELATED WORK

2.1 Human Motion Synthesis

Sequence-centric approaches adopt a holistic model of human motion, constructing a latent space that represents complete motion sequences. This approach captures both the primary characteristics of the motion and secondary features, such as temporal coherence. For example, studies like MoDi (Raab et al., 2024) demonstrate that while results from early training cycles exhibit poor temporal coherence, later examples show significant improvements. Architecturally, sequence-centric methods are compatible with most major generative AI techniques. State-of-the-art methods employ various architectures, including GANs like MoCoGAN (Tulyakov et al., 2017), diffusion models such as MoFusion (Dabral et al., 2023), and VAE-based models like ACTOR (Petrovich et al., 2021). The primary strength of sequence-centric systems lies in their holistic approach: addressing all aspects of a motion sequence leads to reduced complexity compared to pose-centric solutions. However, their drawback is their inability to function online in real time.

Pose-centric solutions, on the other side, synthesise a sequence pose-by-pose, using either a latent space of poses, such as Action2Motion(Guo et al., 2020), or a reward function such as CARL(Luo et al., 2020). They may be further classified depending on their predominant architectural approach: reinforcement learning (Mao et al., 2024), (Luo et al., 2020), dimensionality reduction-based methods such as variational auto-encoders (Guo et al., 2020) or GANs(Xu et al., 2022), and lastly transformer-based approaches (Kania et al., 2021). Whilst each category features a unique approach to condition future poses on the preceding pose or set of poses, they all must ensure that the synthesised individual poses can formulate a valid, temporally coherent motion sequence. To achieve this, they require an extra step, which depends on the architecture model applied. For example, in the case of transformer-based approaches such as (Xu et al., 2022), they employ the attention mechanism to learn this relationship between poses, whereas VAE-based approaches such as Action2Motion (Guo et al., 2020) or MotionNet (Hassan et al., 2021) employ the Kullback–Leibler divergence ubiquitous to their VAE-based architecture.

However, the requirement of this extra step severely constrains the choice of architectures that are employable in pose-centric solutions. On the positive side, their key benefit is that they are capable of online real-time motion synthesis, as shown by state-of-the-art real-time solutions, such as MotionNet in computer games (Hassan et al., 2021), and CARL in robotics (Luo et al., 2020).

2.2 Adaptive Instance Normalization

Instance normalization was proposed by (Ulyanov and Vedaldi, 2017) for neural style transfer, i.e. for transferring the style from one image to another. Specifically, they proposed a method of employing instance normalization instead of batch normalization within a network that can allow the system to learn how to apply a specific style to any supplied input image:

$$IN(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta \qquad (1)$$

The equation above highlights the form of normalization proposed in their original work, where x is the content image, $\mu(x)$ and $\sigma(x)$ are its mean and standard deviation, respectively, and finally, γ and β are learnable parameters.

Adaptive Instance Normalization (AdaIN), proposed by (Huang and Belongie, 2017), is an improved version of instance normalization with two significant advantages. Firstly, it is at least two orders of magnitude faster. Secondly, instead of retraining the system for each desired style, the user can specify a target style by providing a single example. AdaIN works by aligning the mean and variance of the content image with the mean and variance of the style image. This lack of learnable parameters allows the system to be used without any restriction on a variety of styles. The method is formulated as follows:

$$AdaIN(x,y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\mu(y) \qquad (2)$$

where, as previously, x refers to the content image, and $\mu(x)$ and $\sigma(x)$ are its mean and standard deviation, respectively. Additionally, y refers to the style image, and $\mu(y)$ and $\sigma(y)$ are its mean and standard deviation, respectively.

AdaIN has been successfully applied to style transfer across various domains. For example, Ruder et al. (Ruder et al., 2016) demonstrated its application in video stylization. Wang et al. (Wang et al., 2020), used AdaIN to transfer the pose of a 3D mesh to another with a different silhouette, in a process termed neural pose transfer, but not in the context of a motion sequence and without the requirement of temporal coherence.

The novel contribution of this paper is a new approach to learning the temporal relationships between poses in a pose-centric motion synthesis solution. Our approach, inspired by AdaIN's method of decoupling the style and content of images, aims to combine the content from one pose with the temporal aspect of another to create smooth and realistic animations.

3 METHODOLOGY

The proposed methodology is underpinned by a VAE-GAN hybrid architecture, codified by Razghandi et al (Razghandi et al., 2022). It incorporates a Discriminator alongside the Variational Auto-Encoder (VAE) architecture combined in such a way that the VAE decoder plays the role of the generator within a GAN structure.

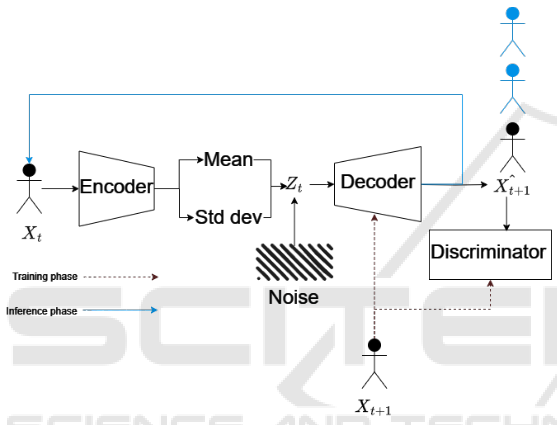

Figure 1: The structure of the proposed solution.

As shown in Figure 1, we propose a training stage and a dedicated sequence generation stage which allow us to manipulate the proposed system to generate a full sequence of motion. The solution treats each pose as a one-dimensional vector of floating point values representing Euler angles of joints in the animated skeleton.

The core components of the solution are described in the following subsections; 3.1 (Encoder), 3.2 (Decoder), 3.3 (AdaIN), and 3.4 (Discriminator). 3.5 outlines the training loop employed, and finally 3.6 describes the iterative sequence generation process.

3.1 Encoder

The structure of the encoder differs from existing VAE-based solutions, such as Action2Motion (Guo et al., 2020), which employs a single-layered gated recurrent unit (GRU) as the encoder. Our multi-layer approach is driven by prior experimentation, demonstrating that a shallower network resulted in poor performance.

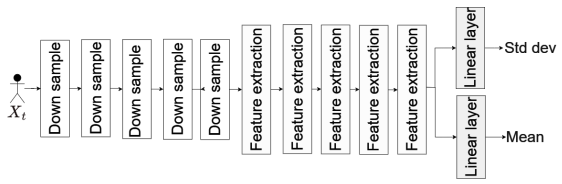

Figure 2: The high-level structure of the encoder.

The encoder is structured to take a single pose as input and encode it into a compressed, lower-dimensional representation. This representation is then passed through two separate linear layers to output the mean and standard deviation of the latent space.

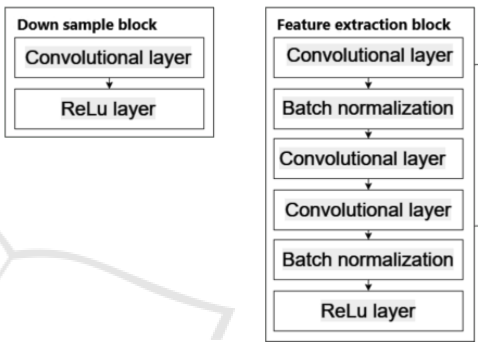

Figure 3: The internal structure of each block of the encoder.

The encoder consists of ten blocks, as depicted in Figure 2, with the internal structure of each block shown in Figure 3. The downsample block, repeated five times, is designed to sample down the incoming data by applying a convolution with a kernel size of 3×3 and a stride of 2. Then, the feature extraction block, also repeated five times, contains three convolutional layers: the first two layers are point-wise convolutions, and the third layer uses a kernel size of 1×1 and a stride of 3.

3.2 Decoder

The input to the VAE decoder is the sampled latent vector, denoted as Z_t in Figure 4. This vector consists of a mean μ_t and a standard deviation σ_t, and is subsequently mapped into a meaningful point in the pose space, using a process known as the reparameterization trick. In the training phase, we augment this architecture by adding dedicated blocks which accept the next pose X_{t+1} (the real pose following the currently processed pose). This trick allows us to guide the decoder and enforce the temporal coherence of the generated sequence.

Compared with state-of-the-art VAE models, the proposed decoder architecture is noticeably more

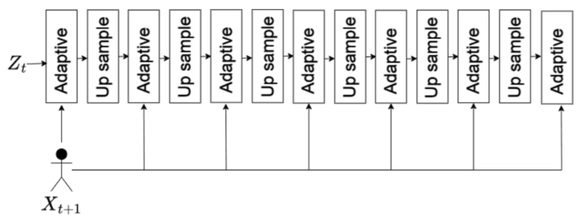

Figure 4: The high-level overview of the structure of the decoder.

complex. For example, Action2Motion (Guo et al., 2020) decoder consists of just a GRU cell with 2 layers. while MotionNet (Hassan et al., 2021) features several sub-encoders and one main decoder, which comprise only a set of linear layers.

In contrast, our decoder network consists of thirteen blocks in total, alternating between adaptive-focused and upsampling-focused blocks. The upsampling blocks are responsible for manipulating the dimensions of the incoming vectors, while the adaptive blocks enforce the temporal coherence between the poses. The internal structure of each block is illustrated in Figure 5. The adaptive blocks, which constitute the primary contribution of this paper, incorporate Adaptive Instance Normalisation layers (AdaIN), inspired by the original work outlined in (Huang and Belongie, 2017). By incorporating these layers at various stages, they can effectively influence the vectors as they progress through the decoder's architecture.

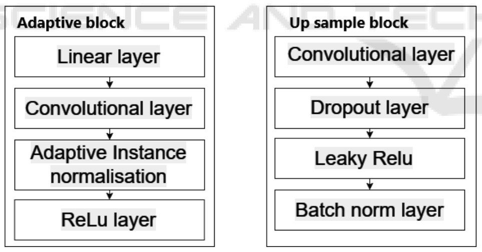

Figure 5: The internal structure of each block employed in the decoder.

3.3 Adaptive Instance Normalization

The decoder's performance is augmented by the inclusion of adaptive blocks. The content of these blocks is shown in Figure 5 and described in section 3.2. The adaptive blocks are structured so that they can be applied at multiple resolutions, as the dimension of the decoded vector is increased through the forward pass of the decoder. Whilst inspired by AdaIN, the proposed solution deviates noticeably.

Our decoder is supplied with two values: the latent vector Z_t, derived from the characteristics provided by the encoder, and the next pose vector X_{t+1}, taken directly from the input dataset. The first step in the proposed solution is to calculate the mean and standard deviation of the next pose vector X_{t+1}:

$$\mu_{t+1} = \mu(X_{t+1}) \tag{3}$$

$$\sigma_{t+1} = \sigma(X_{t+1}) \tag{4}$$

where μ_{t+1} and σ_{t+1} represent the mean and the standard deviation, respectively, calculated for each angle in the pose vector. Further to this we pass μ_{t+1} and σ_{t+1} through separate linear layers:

$$\hat{\mu}_{t+1} = \mu_{t+1} A_{\mu}^{T} + b_{\mu} \tag{5}$$

$$\hat{\sigma}_{t+1} = \sigma_{t+1} A_{\sigma}^{T} + b_{\sigma} \tag{6}$$

where $\hat{\mu}_{t+1}$ and $\hat{\sigma}_{t+1}$ refer to the final output of the linear layers. Subsequently, we apply a variant of Adaptive Instance Normalization to the incoming latent vector Z_t. This form of normalization is implemented in PyTorch (PyTorch, 2024). We further shift and scale the instance normalized vector using $\hat{\sigma}_{t+1}$ and $\hat{\mu}_{t+1}$:

$$\hat{Z}_t = \hat{\sigma}_{t+1} * \left(\frac{Z_t - \mu(Z_t)}{\sigma(Z_t) + \varepsilon} * \gamma + \beta \right) + \hat{\mu}_{t+1} \tag{7}$$

where $\mu(Z_t)$ and $\sigma(Z_t)$ are the mean and standard deviation of the latent vector Z_t, ε is a small value added to avoid dividing by zero, γ and β are both learnable parameters and, lastly, $\hat{\mu}_{t+1}$ and $\hat{\sigma}_{t+1}$ are the vectors calculated in equation 5 and equation 6, respectively. The result of this equation, $\hat{Z}_t$, is the final latent vector which is then passed on to the next layer in the decoder as shown in section 3.2.

3.4 Discriminator

A discriminator is normally part of a GAN, with the purpose of evaluating the output of the generator and therefore guiding it to the target distribution. Figure 6 outlines the internal configuration of the discriminator.

3.5 Training

In the training stage, the VAE architecture is trained so that the decoder outputs poses which are temporally coherent with the one pose provided to the encoder. To achieve this we train the system in such a way as to supply the system with a pair of sequential poses, one is passed to the encoder and the other is sent to the decoder and the discriminator. In Figure 1, the pose sent to the encoder is referred to as X_t and

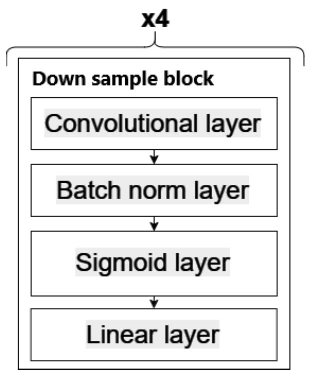

Figure 6: The structure of the discriminator.

the real next pose is referred to as X_{t+1}. Figure 1 also details that in the proposed solution we incorporate a noise vector by adding one to the latent Z_t vector.

3.5.1 Loss Function

Typically, a variational auto-encoder employs two components in its loss function: a reconstruction loss, which we have replaced with a temporal adaptation loss, and the Kullback–Leibler divergence (KL divergence). The KL divergence is a distance calculation between the distribution of the latent space, represented by the mean and standard deviation, and a Gaussian distribution. We further augment this by incorporating a discriminator loss, resulting in a loss function with three distinct components. To prevent any single component from dominating the others, we assign weightings to each of them:

$$L = w_d \cdot L_d + w_t \cdot L_t + w_{kl} \cdot L_{kl} \tag{8}$$

where L_d refers to the discriminator loss, L_t is the temporal adaptation loss, L_{kl} is the KL divergence loss, and w_d, w_t, and w_{kl} are their respective weightings.

3.5.2 Discriminator Loss

The goal of the discriminator during the training phase is to assist the VAE by guiding the distribution to be closer to the target distribution. The exact method of how the discriminator is employed is taken from the initial paper on GANs by Goodfellow et al (Goodfellow et al., 2014).

The discriminator loss is calculated as a mean of two mean square errors, one calculated for the real next poses, X_{t+1}, and one for the poses generated by the decoder, $\hat{X}_{t+1}$:

$$L_d = (L_{mse}(X_{t+1}) + L_{mse}(\hat{X}_{t+1}))/2 \tag{9}$$

The mean square error for the real next poses X_{t+1} is given as:

$$L_{mse}(X_{t+1}) = \left(\frac{1}{n} \sum_{i=1}^{n} (d\left(X_{t+1}\right) - 1)^2 \right) \tag{10}$$

where $d(X_{t+1})$ denotes the discriminator output for the real next pose, which is the predicted probability that the samples are real. The discriminator performance on the real data should return values closer to 1, hence we find the mean square error between the discriminator results on the real poses and a vector of the same length containing just ones.

Similarly, the mean square error for the output of the decoder/generator $\hat{X}_{t+1}$ is given as:

$$L_{mse}(\hat{X}_{t+1}) = \left(\frac{1}{n} \sum_{i=1}^{n} (d\left(\hat{X}_{t+1}\right) - 0)^2 \right) \tag{11}$$

This time, the discriminator performance on the generated (fake) data should return values closer to 0, hence we find the mean square error between the discriminator results and a vector of the same length containing just zeroes.

3.5.3 Temporal Adaptation Loss

One aspect of the traditional VAE loss is the use of a reconstruction loss. Normally this is to encourage the system to reconstruct the input to the encoder. The proposed solution encourages the system to reconstruct the next pose in the sequence relative to the encoded pose. As such, we have referred to this as a Temporal Adaptation Loss, and defined as the mean square error between the generated output $\hat{X}_{t+1}$ and the real next pose in the sequence X_{t+1}:

$$L_t = (\frac{1}{n} \sum_{i=1}^{n} (X_{t+1} - \hat{X}_{t+1})^2) \tag{12}$$

3.5.4 KL Divergence

The primary loss applied to a variational auto-encoder is the KL divergence, used to evaluate the distance between two distributions, typically the latent space and a Gaussian distribution as expressed by the mean and variance, provided by the variational encoder.

We use the following equation:

$$L_{kl} = \frac{1}{N} \sum_{i=1}^{N} \left(-\frac{1}{2} \sum_{j=1}^{D} \left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right) \right) \tag{13}$$

where N refers to the batch size, D refers to the dimensionality of the latent space, μ_j refers to the mean of latent space as determined by the encoder, and lastly σ_j^2 refers to the standard deviation of the latent space as determined by the encoder.

3.6 Iterative Sequence Generation

One of the features of a pose-centric approach is the presence of a dedicated sequence generation stage. In the case of the proposed system, this is achieved via an iterative process which takes advantage of the training goal of the first stage and is shown in Figure 1.

In the first iteration, a pose from the real dataset is selected as the system input. In subsequent iterations, the decoder output $X_{\hat{t}+1}$ is fed back as the input for the encoder. Consecutive output values are combined into the resultant sequence of poses, generated in real time. We rely on the training stage to learn the inter-pose relationships and the temporal coherence. Notably, unlike in the training session, during the inference phase, there is no known next pose x_{t+1} that we can feed to the decoder as additional input; instead, we provide a tensor in the shape of a pose containing all ones.

4 RESULTS

4.1 Datasets

The proposed methodology is tested on two commonly used datasets: CMU MoCap dataset (University, 2000) and HumanAct12 dataset (Guo et al., 2020). The CMU MoCap dataset is a diverse collection of several thousand motion clips of various lengths. To manage the output we focused on those that fit the description of "walk" and "run". To determine the clips that fit into this description, we used the dataset's website to generate a valid list. These criteria gave us a working dataset of 206 animation sequences and a total pose count of 257,633. The HumanAct12 dataset is considerably smaller, therefore we do not constrain it to a specific category. It contains a total of 1191 motion clips with a total of 90,099 individual poses.

4.2 Quantitative Results

We use two separate metrics to evaluate the impact of the proposed changes: Fréchet Inception Distance (FID) and diversity (Div). These metrics quantify different aspects of the generative process. FID, a standard metric introduced by Heusel et al. (Heusel et al., 2017), measures the quality of generated samples by comparing them to real ones. It evaluates the similarity of the distributions of features extracted by a pre-trained classifier for both generated and real samples. A lower FID score has been shown to correlate with generated samples that more closely resemble those from the real dataset.

Diversity, on the other hand, focuses solely on the generated samples, quantifying their inherent variance, and reflecting the model's ability to produce a wide range of samples. This metric is inherently constrained by the diversity within the training dataset. Ideally, a generative system should achieve both a low FID score and high diversity, indicating that it can produce high-quality samples that accurately capture the variability of the entire dataset while remaining indistinguishable from real samples. The complementary nature of these metrics makes them valuable when used together, and we have found no compelling reason to deviate from this approach.

The quantitative results for the CMU MoCap dataset are shown in Table 1 and the results for the HumanAct12 dataset are shown in Table 2. This comparison only includes unguided solutions, as guided motion systems are not directly comparable. In the case of the MotionNet method (Hassan et al., 2021), we used a single component of a larger system called SAMP, and analysed its performance as an unguided solution after removing its interactive elements.

Table 1: Quantitative results of the systems against existing methodologies which explicitly evaluate themselves on the CMU dataset.

Methods	FID	Div
Two-stage GAN (Cai et al., 2017)	14.34	4.41
CondGRU (Shlizerman et al., 2017)	51.72	0.79
MoCoGAN (Tulyakov et al., 2017)	11.15	5.28
MoFusion (Dabral et al., 2023)	50.31	9.09
MotionNet (Hassan et al., 2021)	4.31	7.33
action2motion (Guo et al., 2020)	2.8	6.5
Ours	1.83	9.21

Table 2: Quantitative results of the systems against existing methodologies which explicitly evaluate themselves on the HumanAct12 dataset.

Methods	FID	Div
ACTOR (Petrovich et al., 2021)	48.8	14.1
MDM (Tevet et al., 2022)	31.92	17.00
MoDi(Raab et al., 2024)	13.03	17.57
Ours	10.11	15.33

In terms of the FID metrics, these results demonstrate consistently superior performance of our approach compared to the state-of-the-art. In terms of diversity, the results are still superior for the CMU dataset, and comparable with other methods for the HumanAct12 dataset.

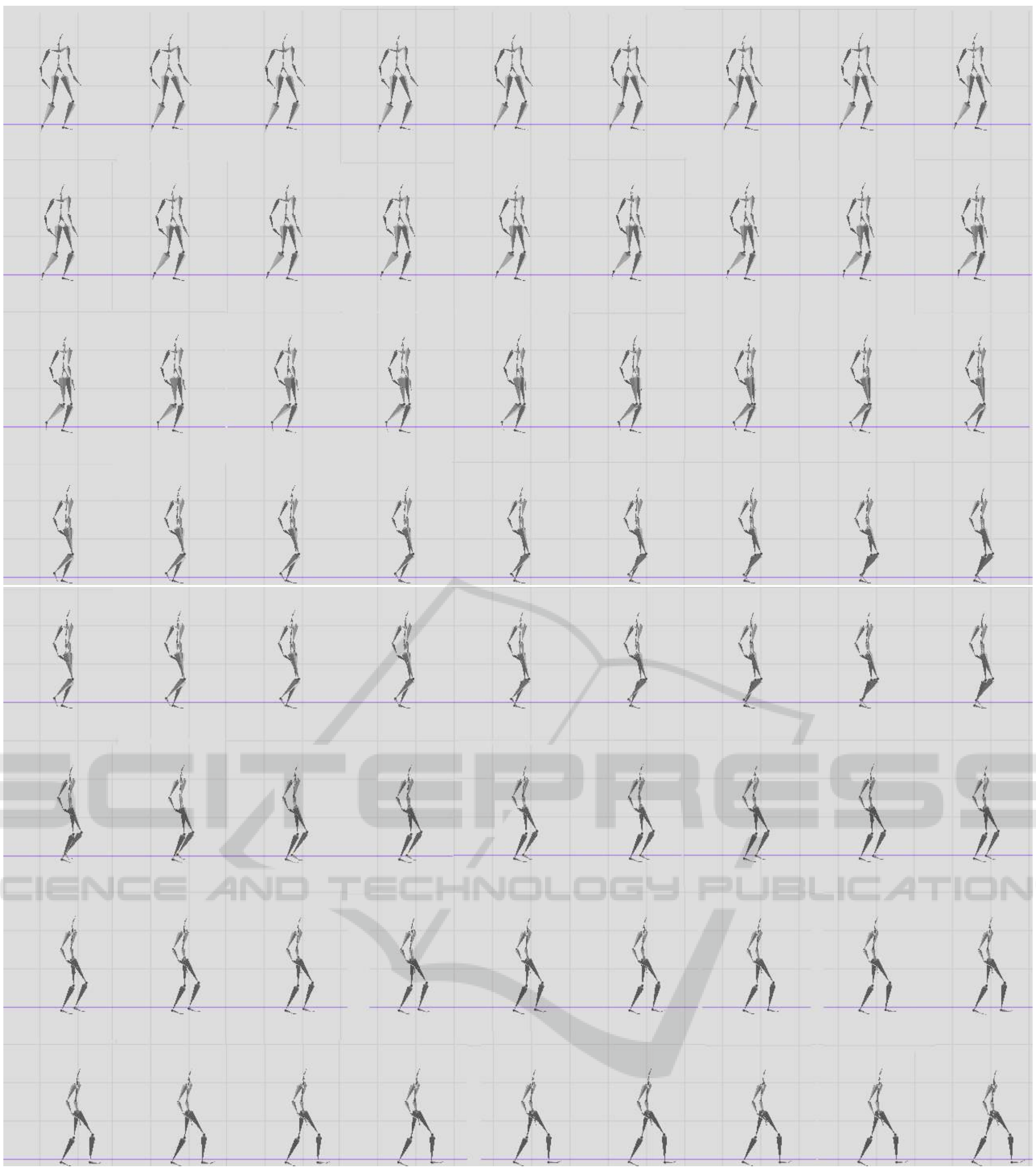
Figure 7: This shows an example of a walk sequence generated using the CMU dataset. The results should be read left to right with the top left pose showing the first frame taken from the VAE.

4.3 Qualitative Results

The evaluation shown in Figure 7 demonstrates two key aspects of the proposed solution: maintaining the temporal coherence and the influence of the selected pose on the diversity of the generated sequences.

4.3.1 Temporal Coherence

By examining the frame-by-frame breakdown in Figure 7 we can see that including AdaIN layers within the decoder has allowed the proposed solution to learn and maintain temporal coherence between a diverse set of poses.

4.3.2 The Influence of Selected Pose

The sequences generated are constrained considerably by the initial pose provided to the encoder. This is evidenced by our diversity score that is comparable but not higher than other state-of-the-art methods, as shown in Table 2.

4.4 Ablation Study

To evaluate the efficacy of the proposed AdaIN layers, we conducted a comprehensive ablation study focusing on two key aspects:

1. Comparative analysis with recurrent architectures: we investigated the impact of replacing adaptive instance normalization with established recurrent neural network architectures, specifically LSTM and GRU layers. This comparison was motivated by the methodology employed in Action2Motion (Guo et al., 2020), which represents the closest existing approach in the literature. The results, shown in Table 3, highlight that, despite LSTM and GRU layers being capable of producing good results, in the specific context of a pose-centric employing an adaptive layer can result in superior results.
2. Integration with existing systems: To further validate our approach, we examined the effect of incorporating Adaptive Instance Normalization into existing motion synthesis frameworks. To investigate this we selected two separate frameworks which utilize different architectures therefore highlighting the applicability of the proposed approach. The first framework selected is Action2Motion (Guo et al., 2020). This was selected due to it similarly employing a VAE-based approach to the proposed system to generate sequences in a pose-centric manner. The results of retraining Action2Motion to include adaptive layers within the decoder are shown across three separate tables, Table 4, Table 5 and Table 6. We show that employing both Lie algebra and adaptive layers results in superior performance. The second framework selected is (Wen et al., 2021). This was selected as it employs a different architecture, in this case normalizing flow, whilst also generating sequences in a pose-centric manner. We modified this framework by incorporating the adaptive layers from our solution within the transformation mapping employed by the authors. It was subsequently trained using the original dataset provided by them, once with adaptive layers, and once without. The results are shown in Table 7. These results demonstrate the versatility of integrating adaptive layers across diverse architectures, underscoring their ability to enhance model performance. By incorporating these layers, we observe a consistent improvement in performance, highlighting their positive contribution to the overall system efficacy.

Table 3: Ablation study on the impact of different methods of maintaining temporal coherence.

Methods	FID	Div
With AIN layer	1.83	9.21
Without AIN layer	7.11	6.39
Replaced by LSTM layer	5.95	4.92
Replaced by GRU layer	4.42	5.71

Table 4: Ablation study on the impact of adaptive instance normalization in Action2Motion for Human-Act-12.

Normalization	FID	Div
With AdaIN with Lie	2.151	7.115
With AdaIN w/o Lie	2.832	6.821
W/o AdaIN with Lie	2.458	7.032
W/o AdaIN w/o Lie	3.299	6.742

Table 5: Ablation study on the impact of adaptive instance normalization in Action2Motion for CMU.

Normalization	FID	Div
With AdaIN with Lie	2.441	6.740
With AdaIN w/o Lie	2.712	6.210
W/o AdaIN with Lie	2.885	6.500
W/o AdaIN w/o Lie	2.994	5.790

Table 6: Ablation study on the impact of adaptive instance normalization in Action2Motion for NTU.

Normalization	FID	Div
With AdaIN with Lie	0.217	7.123
With AdaIN w/o Lie	0.433	6.833
W/o AdaIN with Lie	0.350	6.926
W/o AdaIN w/o Lie	0.540	7.065

Table 7: Quantitative results of placing adaptive layers within a system which employs normalizing flow (Wen et al., 2021).

Normalization	FID	Div
With AdaIN	5.127	6.341
w/o AdaIN	7.737	5.327

5 CONCLUSION

This paper deals with the challenge of temporal coherence in sequences produced by pose-centric methods, so to achieve real-time motion synthesis. We propose adopting the Adaptive Instance Normalization (AdaIN) (Huang and Belongie, 2017) within a VAE-GAN architecture, including AdaIN layers in the decoder.

The inclusion of AdaIN layers leads to superior results, in comparison to other state-of-the-art architectures on the CMU MoCap and HumanAct12 datasets, evaluated by the FID and diversity metrics. We have also demonstrated that inclusion of the proposed AdaIN layers improves the performance of other pose-centric methods such as Action2Motion (Guo et al., 2020) and (Wen et al., 2021) regardless of the underlying architecture employed. Therefore, a key strength of our proposed solution is the versatility of the AdaIN layers and the potential to be included in other pose-centric motion synthesis architectures.

In future work, we plan to further investigate the potential of the AdaIN in pose-centric motion synthesis, and especially focus on guided methods that will allow the interaction of virtual characters with other elements in computer games.

REFERENCES

Cai, H., Bai, C., Tai, Y.-W., and Tang, C.-K. (2017). Deep video generation, prediction and completion of human action sequences. *ECCV*.

Dabral, R., Mughal, M. H., and Vladislav Golyanik, C. T. (2023). Mofusion: A framework for denoising-diffusion-based motion synthesis. *CVPR*.

Goodfellow, I. J., Pouget-Abadie, J., Xu, M. M. B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *neurips*.

Guo, C., Zuo, X., Wang, S., Zou, S., Sun, Q., Deng, A., Gong, M., and Cheng, L. (2020). Action2motion: Conditioned generation of 3d human motions. *Proceedings of the 28th ACM International Conference on Multimedia*.

Hassan, M., Ceylan, D., Villegas, R., Saito, J., Yang, J., Zhou, Y., and Black, M. (2021). Stochastic scene-aware motion prediction. *ICCV*.

Heusel, M., Unterthiner, H. R. T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)*.

Kania, K., Kowalski, M., and Trzciński, T. (2021). Trajevae: Controllable human motion generation from trajectories. *arXiv preprint arXiv:2104.00351*.

Li, P., Aberman, K., Zhang, Z., Hanocka, R., and Sorkine-Hornung, O. (2022). Ganimator: Neural motion synthesis from a single sequence. *ACM Transactions on Graphics*.

Luo, Y., Soeseno, J. H., Chen, T. P., and Chen, W. (2020). CARL: controllable agent with reinforcement learning for quadruped locomotion. *CoRR*, abs/2005.03288.

Mao, Y., Liu, X., Zhou, W., Lu, Z., and Li, H. (2024). Learning generalizable human motion generator with reinforcement learning.

Mourot, L., Hoyet, L., Le Clerc, F., Schnitzler, F., and Hellier, P. (2021). A survey on deep learning for skeleton-based human animation. *Computer Graphics Forum*, 41(1):122–157.

Petrovich, M., Black, M., and Varol, G. (2021). Action-conditioned 3d human motion synthesis with transformer vae. *ICCV*.

PyTorch (2024). torch.nn.instancenorm1d. Accessed: 2024-06-27.

Raab, S., Leibovitch, I., Li, P., Popa, T., Aberman, K., and Sorkine-Hornung, O. (2024). Modi: Unconditional motion synthesis from diverse data. *CVPR*.

Razghandi, M., Zhou, H., Erol-Kantarci, M., and Turgut, D. (2022). Variational autoencoder generative adversarial network for synthetic data generation in smart home.

Ruder, M., Dosovitskiy, A., and Brox, T. (2016). *Artistic Style Transfer for Videos*, page 26–36. Springer International Publishing.

Shlizerman, E., Dery, L. M., Schoen, H., and Kemelmacher-Shlizerman, I. (2017). Audio to body dynamics. *CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Tevet, G., Raab, S., Gordon, B., Cohen-Or, Y. S. D., and Bermano, A. H. (2022). Human motion diffusion model. *ICLR*.

Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. (2017). Mocogan: Decomposing motion and content for video generation. *CVPR*.

Ulyanov, D. and Vedaldi, A. (2017). Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis.

University, C. M. (2000). Cmu graphics lab motion capture database. https://mocap.cs.cmu.edu/. Accessed: 2024-09-15.

Wang, J., Wen, C., Fu, Y., Lin, H., Zou, T., Xue, X., and Zhang, Y. (2020). Neural pose transfer by spatially adaptive instance normalization. *CVPR*.

Wen, Y.-H., Yang, Z., Fu, H., Gao, L., Sun, Y., and Liu, Y.-J. (2021). Autoregressive stylized motion synthesis with generative flow.

Xu, L., Song, Z., Wang, D., Su, J., Fang, Z., Ding, C., Gan, W., Jin, Y. Y. X., Yang, X., Zeng, W., and Wu, W. (2022). Actformer: A gan-based transformer towards general action-conditioned 3d human motion generation. *ICCV*.

Zhang, M., Guo, X., Pan, L., Cai, Z., Hong, F., Li, H., Yang, L., and Liu, Z. (2023). Remodiffuse: Retrieval-augmented motion diffusion model.