Diverse Motion Stylization for Multiple Style Domains via Spatial-Temporal Graph-Based Generative Model

SOOMIN PARK, KAIST, Korea
DEOK-KYEONG JANG, KAIST, Korea
SUNG-HEE LEE, KAIST, Korea

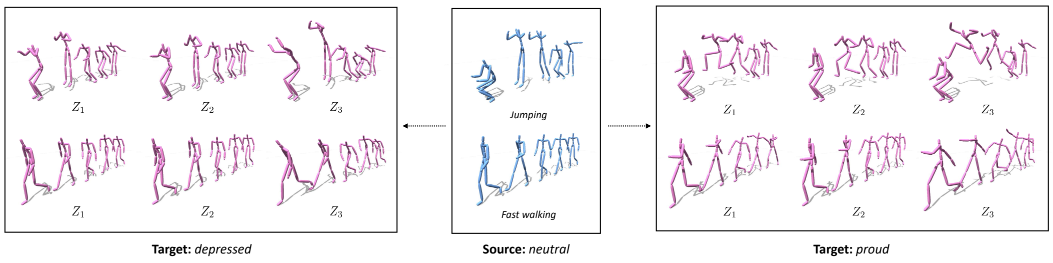


Fig. 1. Our motion style transfer results on two motion clips, *jumping* and *fast walking*, with three random latent variables ($\mathbf{z}_{1,2,3}$), without using style reference motions.

This paper presents a novel deep learning-based framework for translating a motion into various styles within multiple domains. Our framework is a single set of generative adversarial networks that learns stylistic features from a collection of unpaired motion clips with style labels to support mapping between multiple style domains. We construct a spatio-temporal graph to model a motion sequence and employ the spatial-temporal graph convolution networks (ST-GCN) to extract stylistic properties along spatial and temporal dimensions. Through spatial-temporal modeling, our framework shows improved style translation results between significantly different actions and on a long motion sequence containing multiple actions. In addition, we first develop a mapping network for motion stylization that maps a random noise to style, which allows for generating diverse stylization results without using reference motions. Through various experiments, we demonstrate the ability of our method to generate improved results in terms of visual quality, stylistic diversity, and content preservation.

CCS Concepts: • **Computing methodologies** → **Motion processing**; **Neural networks**.

Additional Key Words and Phrases: motion synthesis, generative model, graph convolutional networks, character animation, deep learning

ACM Reference Format:
Soomin Park, Deok-Kyeong Jang, and Sung-Hee Lee. 2021. Diverse Motion Stylization for Multiple Style Domains via Spatial-Temporal Graph-Based Generative Model. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 3 (September 2021), 17 pages. https://doi.org/10.1145/3480145

This work was supported by National Research Foundation, Korea (NRF-2020R1A2C2011541) and by KEIT, Korea (20011076).
Authors' addresses: Soomin Park, KAIST, Daejeon, Korea, sumny@kaist.ac.kr; Deok-Kyeong Jang, KAIST, Daejeon, Korea, shadofex@kaist.ac.kr; Sung-Hee Lee, KAIST, Daejeon, Korea, sunghee.lee@kaist.ac.kr.

2577-6193/2021/9-ART $15.00
https://doi.org/10.1145/3480145

1 INTRODUCTION

The motion style, the way movement is performed, is an essential component of motion as it reflects various attributes of characters such as mood, personality, or identity. The motion style is an essential element to realize life-like virtual characters and enrich storytelling in computer animation.

However, acquiring stylized motions is very challenging since capturing movements with all the necessary content and style is an expensive and time-consuming process. Due to the lack of suitable motion stylization tools, creating a single avatar motion from a motion capture database requires additional work by professional artists to add the motion style manually. As one solution to reduce this workload, motion style transfer aims to extract the target style from a motion example and transfer it to another motion with the desired content.

Motion styles reflect various attributes somewhat ambiguously and are challenging to formulate accurately. Thus, many previous studies relied on the data-driven approaches for inferring motion style, for which researchers have achieved remarkable progress [Aberman et al. 2020; Aristidou et al. 2017; Du et al. 2019; Min et al. 2010; Smith et al. 2019; Xia et al. 2015; Yumer and Mitra 2016]. One primary focus in motion stylization is designing deep neural networks [Aberman et al. 2020; Chan and Ho 2021; Dong et al. 2020; Du et al. 2019; Holden et al. 2016]. The most recent work of Aberman et al. [2020] proposed a generative adversarial networks (GAN) based architecture augmented with the adaptive instance normalization (AdaIN) that learns to generate multiple styles from an unpaired motion collection. In this paper, following the same approach, we propose a significantly improved deep learning architecture for the two limitations of the previous work: First, deep neural network-based models using 1D convolution do not consider the spatial relations between joints, and thus may fail to preserve motion content when transferring style between significantly different action types, e.g., *neutral-punching* and *proud-jumping*, and when translating the style into long-term motions that include multiple action types. Secondly, previous methods can encode the style only from an example motion, which may be impractical in the real-world application. Since the style is highly variable depending on the motion content, e.g., *childlike-punching* vs. *childlike-walking*, it is not always feasible to have a representative motion for a particular style. In addition, style encoding through motion is inefficient in the case of generating diverse stylized output for a specific style. In practical applications, it would be easier for the user to explore the learned style space and control the style using latent variables than to find a relevant reference motion.

In this paper, we present a new deep learning framework to tackle the above limitations. To overcome the problem of 1D convolution on motion, we construct a spatio-temporal graph to model a motion sequence and employ the spatial-temporal graph convolution networks (ST-GCN) [Yan et al. 2018] to extract high-level stylistic properties along both spatial and temporal dimensions. To allow for the diverse stylization of a source motion without using reference motions, we develop a mapping network that generates the style code from random noise. These technical features have not been introduced to the motion style transfer problem before. Our framework learns stylistic features from a collection of unpaired motion clips with style labels while supporting the mapping between multiple style domains. Specifically, we adopt a multi-task learning approach [Choi et al. 2020] to make our network architecture scalable to the increasing number of domains.

The domain in this context refers to a set of motions that can be grouped by a particular style, e.g., *old* or *childlike*, that is distinguished from the motion content, which refers to the nature of action itself, such as *walking* or *jumping*. We assume that the motion content provides critical semantic information about an action type, such as phase and foot contact timing.

Figure 1 shows that our method translates *jumping* and *fast walking* motions into two style domains, *depressed* and *proud*, creating diversely styled motions with three random latent variables. The contributions of our work can be summarized as follows:

- Through spatio-temporal modeling using GCN, our framework shows improved style translation results between significantly different actions and on a long motion sequence containing multiple actions in terms of visual quality and content preservation.
- For the motion style translation task, we develop a network that maps a random noise to style, allowing diverse stylization results to be generated without using reference motion.

We show the effectiveness of our method with respect to the quality of transfer into multiple style domains, diversity of stylization with random noises, and the interpolation of styles. We also present comparisons with previous work in terms of the naturalness and the degree of content preservation of the stylized motions.

2 RELATED WORK

In this section, we focus on discussing the data-driven methods for motion style transfer. We also introduce important studies on the image stylization that greatly contributed to the motion style problem.

2.1 Image Style Transfer

Image style transfer aims to learn the mapping from an image of a source style domain to an image of a target domain. Gatys et al. [2016] proposed an impressive method that transfers style between images by matching feature statistics in convolutional layers. Subsequently, Li et al. [2017] and Ulyanov et al. [2017] improved the quality of image stylization significantly by introducing an instance normalization (IN) technique. They showed that matching a number of statistics, such as channel-wise mean and variance, is efficient for the style transfer. Noting that the instance normalization performs a form of style normalization by normalizing feature statistics, Huang et al. [2017] extended IN to the adaptive instance normalization (AdaIN). AdaIN injects style information into content input by adjusting the mean and variance of the style input. Many recent studies successfully used the AdaIN in the generative model for the image stylization [Baek et al. 2020; Choi et al. 2020; Huang et al. 2018; Liu et al. 2019].

To address the style diversity, Huang et al. [2018] and Lee et al. [2018] proposed to map images onto two disentangled content and style spaces. The code from the content space encodes a domain-invariant feature extracting shared information across domains, and the style code encodes a domain-specific attribute feature. This disentanglement of features greatly reduces mode collapse and improves the quality of translated images. However, these methods are not scalable to an increasing number of domains since they only consider a mapping between two domains.

To solve this scalability problem, a body of work [Choi et al. 2020; Donahue and Simonyan 2019; Isola et al. 2017; Lee et al. 2020; Liu et al. 2019] extended the unsupervised image-to-image translation to handle multi-modal translation. Choi et al. [2020] proposed a scalable approach that can generate diverse images across multiple domains. Their encoding network has multiple output branches, each of which extracts style codes for a specific domain. They also introduced a mapping network that learns to transform a random Gaussian noise into a style code to diversify the generated images. Our approach is similar to [Choi et al. 2020] in that it maps a random noise to style to obtain style diversities and that the training dataset consists of multiple classes.

2.2 Motion Style Transfer

Various data-driven approaches have been developed for motion style research. Hsu et al. [2005] used a linear time-invariant system to model the difference between two motions with different styles and applied the extracted style to a new motion. This method is effective only to a set of aligned, similar motions. Ikemoto et al. [2009] used Gaussian process models of kinematics and dynamics to edit the input motion into the desired style. Taylor and Hinton [2009] used the conditional restricted Boltzmann machine (CRBM) to model the human motion condtioned on style. Xia et al. [2015] presented an online learning algorithm using KNN search to construct a series of local mixtures of auto-regressive models (MAR) for capturing relationships between motion styles. The constructed model enables transferring styles between various motion clips. Yumer and Mitra [2016] extracted spectral intensity features between similar actions with different styles and transferred them to a new motion to control styles. These studies require preprocessing of motion, e.g., alignment and database searching, to model the difference between a motion pair or to model a style feature and apply it to a new motion. Thus, they are effective for the range of motions in the database but may not scale well to unseen data.

Recently, deep learning-based approaches have greatly improved the quality and the possible range of motion stylization. Holden et al. [2017; 2016] applied the Gram matrix method [Gatys et al. 2015] to transfer motion style through motion editing in the latent space. Du et al. [2019] proposed a conditional variational autoencoder (CVAE) to learn the distribution of style motion conditioned by the Gram matrix-based style features. These approaches require much computing time on the test set to extract style features through a slow optimization process and tend to fail to transfer style between significantly different motions. Smith et al. [2019] presented a fast and simple network architecture that generates stylized motions in real-time. During the training time, it requires the preprocessing of frame-by-frame spatial matching between motion data. Mason et al. [2018] proposed a motion controller that learns to synthesize stylized motions with only a limited amount of motion data via few-shot learning. Yet, similarly to [Du et al. 2019], this approach is also limited to certain motion types, such as locomotion.

The work of [Aberman et al. 2020] alleviates the restrictions on training data by allowing for training the networks with an unpaired dataset with style labels. It, also notably, can transfer style from videos to 3D animations by learning a common style embedding for both 3D and 2D joint positions.

The deep learning approaches above learn only one style code for each style domain and transfer it into the content motion. However, there are countless style variations even in the same style class. Our framework allows for creating diverse stylized results within the same style class through mapping a random noise to style. In addition, previous methods model the human motion as the temporal changes of all joints, overlooking the spatial structure of human skeleton. In contrast, our framework considers the spatial relationship between joints by using ST-GCN [Yan et al. 2018] to improve the quality of the stylized motions.

3 OVERVIEW

We develop a GAN-based framework that translates a motion into multiple style domains. The overall pipeline of our framework is shown in Figure 2. Given an input motion sequence, the motion stylizer G learns to transform the motion into the target style using a domain-specific style code, which represents diverse styles found in a particular domain. Similar to the previous study [Aberman et al. 2020], we design the style encoder E to extract the style code from the given reference motion. Further, we train the mapping network F that allows for a more detailed stylistic control by generating diverse style codes for a given domain label and a random Gaussian noise,

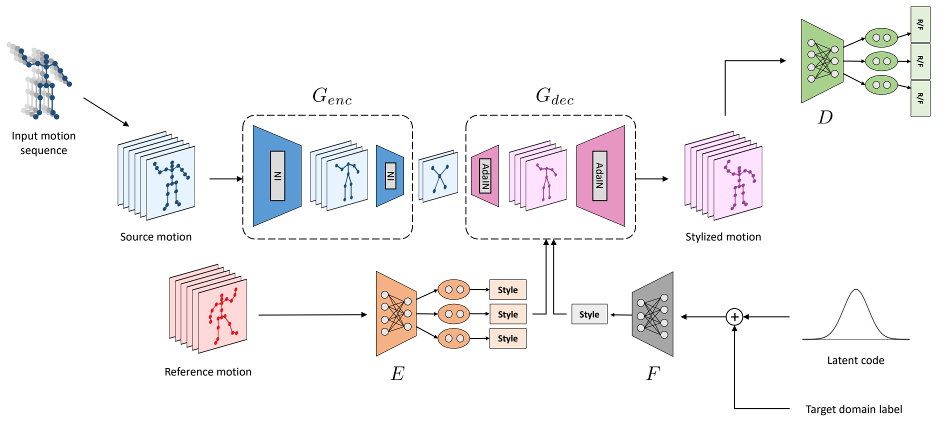


Fig. 2. The overall pipeline of our framework. Given a source motion, the motion stylizer G transforms the motion into the target style using a domain-specific style code, which is created either by the style encoder E or the mapping network F. E extracts the target style code from the given reference motion, while F maps the random noise into the style code, allowing for a more detailed stylistic control given a domain label, without requiring any reference motions.

without requiring any reference motions. As an adversarial architecture to G, the discriminator D learns to classify motion examples as real (from the domains) or fake (generated ones) with respect to the given style domain label. We use the spatio-temporal graph convolutional networks (ST-GCN) as a basis of our network to integrate the spatial and temporal information of complex human motion.

After describing the process for converting the motion clip into the suitable format for our graph-based deep learning algorithm (Sec. 4), we summarize the ST-GCN used in our architecture (Sec. 5). In the following, we explain the four main network components of our architecture and the training process (Sec. 6).

4 MOTION DATA PREPARATION

Dataset. We use motion capture databases acquired from the previous motion style transfer studies [Aberman et al. 2020; Xia et al. 2015]. We divide the databases into $\mathcal{M}_A$ (homogeneous) and $\mathcal{M}_B$ (heterogeneous) datasets based on the content-homogeneity of the motion, i.e., the former has a single content in a single motion sequence while the latter contains multiple contents. The motion datasets are unpaired but grouped by multiple style domains. We train our networks with $\mathcal{M}_A$ and use $\mathcal{M}_B$ as the test data to evaluate whether our method can be extended to previously unseen, long-term motion sequences that consist of various contents.

Preprocessing. The collected motion data is converted into an appropriate format for training using the preprocessing scheme in [Holden et al. 2016]. We first retarget the motion data into a single CMU skeletal system with $N = 21$ joints. Then, the size of the dataset is doubled by mirroring the original motion clips. While the motion data can vary in frame rate and duration at test time, the motion data at training time is sub-sampled into half of the original frame rate (60 fps) and is sliced into $T = 64$ frame clips with $T/2$ frames overlapping with adjacent clips. Motion clips less than T frames are padded with the first and the last frames. For our neural system to be stably trained, we scale the motion data by subtracting the mean pose and dividing by the standard deviation.

Motion data representation. The motion sequence can be represented as $\mathbf{m} = [v_{1,i}, ..., v_{t,i}, ..., v_{T,i}]_{i=1}^{N}$. The pose vector of the i-th joint at the t-th time stamp is $v_{t,i} = [j^p_{t,i}, j^r_{t,i}]$, where $j^p \in \mathbb{R}^3$ and $j^r \in \mathbb{R}^4$ are the joint position and joint rotation (unit quaternion) relative to the reference frame, which is the root transformation projected onto the ground. Thus, we represent the input motion data as $\mathbf{m} \in \mathbb{R}^{T \times N \times d}$, where $d = 7$ is the number of feature dimensions.

Motion graph construction. In this work, we assume that the motion data can be structured as a spatio-temporal graph. Joint nodes within a frame are connected by the spatial edges according to the bone connections H. We represent this intra-body structure with $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ where $\mathcal{V}_s$ is a spatial joint set with $|\mathcal{V}_s| = N$ and $\mathcal{E}_s = \{v_{t,i}v_{t,j} | (i, j) \in H\}$ is a spatial edge set, which reflects the spatial connectivity of each column in $\mathbf{m}$. On the other hand, the temporal edge links the same joint in the adjacent frames. We represent this inter-frame structure $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ where $\mathcal{V}_t$ is a temporal joint set with $|\mathcal{V}_t| = T$ and $\mathcal{E}_t = \{v_{t,i}v_{(t+1),i}\}$ is a temporal edge set, which reflects the temporal connectivity of each row in $\mathbf{m}$. Finally, the spatio-temporal graph on a motion sequence can be represented as a unified graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = NT$, allowing us to process data jointly across space and time.

5 GRAPH CONVOLUTIONAL NETWORKS

To encode high-level features from the spatio-temporal graph, ST-GCN [Yan et al. 2018] is used in our framework. Here we summarize the ST-GCN implementation in our architecture. Please refer to [Yan et al. 2018] for the detail of the ST-GCN.

5.1 Spatial Graph Convolution

We first explain the spatial graph convolution on motion data at one frame. GCN can be viewed as a generalization of CNN with no canonical spatial representation. We represent the spatial graph convolution as:

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(l_{ti}(v_{tj})), \tag{1}$$

where $f_{in}(v)$ and $f_{out}(v)$ are input and output feature maps of a graph node v. The neighbor set B of v_{ti} is a set of nodes v_{tj} that are within D edge hops, i.e., $B(v_{ti}) = \{v_{tj} \mid d(v_{ti}, v_{tj}) \leq D\}$, where $d(v_{ti}, v_{tj})$ is the minimum number of edge hops from v_{ti} to v_{tj}. We only take into account neighbors in 1-hop (i.e., D=1). $B(v_{ti})$ is further partitioned into a fixed number of subsets with numeric labels by a mapping $l_{ti} : B(v_{ti}) \rightarrow \{0, 1, 2\}$. The joint nodes are labeled as 0 for v_{ti}, 1 if closer to the root than v_{ti}, and 2 otherwise. The weight function $\mathbf{w}$ is implemented as $\mathbf{w}(l_{ti}(v_{tj}))$ to apply the same weight to the nodes with the same label. The normalizing term $Z_{ti}(v_{tj}) = |\{v_{tk}|l_{ti}(v_{tk}) = l_{ti}(v_{tj})\}|$ is the cardinality of each subset that v_{tk} belongs to.

After completing the spatial convolution according to the above equation, we proceed with the temporal convolution over the time axis. Since input vectors on the time axis are aligned in chronological order, we continue the temporal convolution in accordance with the temporal locality.

5.2 Graph Pooling and Unpooling

We use the average pooling operation to reduce the output dimensionality after convolutional layers. Since the temporal dimension of the graph is uniformly structured with vertices sequentially connected along time, the traditional average pooling method can be used. In contrast, as its spatial dimension is non-uniformly structured, we consider the spatial pooling operation as the hierarchical graph clustering, i.e., from the finer graph into the sparser graph. At each downsampling step, we average the consecutive joint nodes only within the local body part (cluster) in order to create a coarser-grained hierarchical graph structure, as illustrated in Figure 3. As a result of

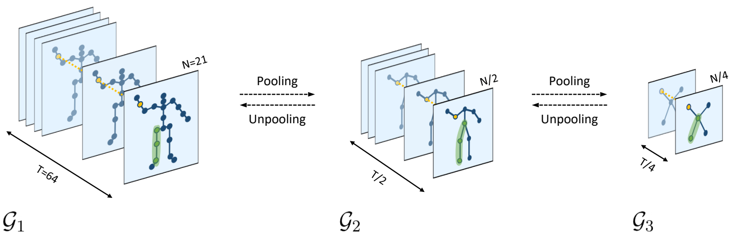


Fig. 3. Illustration of the spatio-temporal graph topology obtained by pooling and unpooling. The consecutive joint nodes within the local body part are averaged via pooling to create a coarser-grained hierarchical graph structure. The averaged nodes are mapped back to the previous skeleton structure via unpooling.

pooling, our framework uses graphs $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ in varying resolutions. Specifically, $|\mathcal{V}_1| = NT$, $|\mathcal{V}_2| = N/2 \times T/2$, and $|\mathcal{V}_3| = N/4 \times T/4$.

Upsampling takes place in the opposite direction of the downsampling. The linear interpolation is used for upscaling the temporal resolution of data. The spatial unpooling upscales the joint dimension by mapping the pooled nodes to the previous skeleton structure.

6 MOTION STYLE TRANSFER FRAMEWORK

In this section, we describe the network components of our framework. During training, we arbitrarily sample a source motion $\mathbf{m} \in \mathcal{M}_A$ within the source domain y, a reference motion $\overline{\mathbf{m}} \in \mathcal{M}_A$ within the target domain $\bar{y}$, and a latent code $\mathbf{z} \in \mathcal{Z}$.

6.1 Architecture

Motion Stylizer. Motion Stylizer G is an encoder-decoder architecture, which transfers a motion from one style domain to another. We provide the encoder with an input motion $\mathbf{m}$ and then feed its output to the decoder along with a style code. The encoder G_{enc} consists of two ST-GCN residual blocks, each accompanied by the instance normalization (IN) layer [Wang et al. 2019]. In the encoding process, IN removes the style variation in motion by normalizing the feature statistics across each channel in each training example. As a result, we obtain an output feature map h, which is normalized with respect to style, thus containing only the domain-invariant information. The encoding process can be written as:

$$h = G_{enc}(\mathbf{m}) \tag{2}$$

The decoder G_{dec} is a network composed of two ST-GCN residual blocks, which transform the intermediate feature map h into the output motion $\hat{\mathbf{m}}$ using the target style code $\hat{\mathbf{s}}$.

$$\hat{\mathbf{m}} = G_{dec}(h, \hat{\mathbf{s}}) \tag{3}$$

The output motion is required to satisfy two essential properties. Firstly, it should reveal distinct style differences as $\hat{\mathbf{s}}$ varies. Secondly, it should always preserve the content of the original motion, regardless of $\hat{\mathbf{s}}$. Keeping these aspects in mind, we employ an adaptive instance normalization (AdaIN) [Huang and Belongie 2017] and a skip architecture. AdaIN, in particular, is key to a successful style deformation. This normalization technique injects a style to the input by adjusting the feature statistics as:

$$AdaIN(\mathbf{m}, \hat{\mathbf{s}}) = \mu(\hat{\mathbf{s}}) \left(\frac{\mathbf{m} - \sigma(\mathbf{m})}{\mu(\mathbf{m})} \right) + \sigma(\hat{\mathbf{s}}) \tag{4}$$

where σ and μ represent the channel-wise mean and variance, respectively. AdaIN adaptively normalizes the source motion $\mathbf{m}$ to the corresponding style $\hat{\mathbf{s}}$, which helps characterize the output motion $\hat{\mathbf{m}}$ according to the domain-specific information. The skip connections are used to take advantage of the early layers rich in spatial details. We observed that the skip architecture helps convey the semantic, contextual information of motion during decoding and thus preserve the essential content of the original motion.

Style encoder. Style Encoder E takes a reference motion $\overline{\mathbf{m}}$ with its domain label $\bar{y}$ as input.

$$\hat{\mathbf{s}} = E_{\bar{y}}(\overline{\mathbf{m}}) \tag{5}$$

We use the multi-task learning (MTL) approach, which allows E to extract a per-domain style code. Our style encoder is composed of two ST-GCN blocks with an MLP for multiple output branches. The ST-GCN blocks are a shared component that extracts shared features across all domains. We use a separate fully-connected layer for each branch to obtain domain-specific information from the shared features.

Mapping network. In addition to Style Encoder, we also create Mapping Network F that generates a style code $\hat{\mathbf{s}}$ from a random latent vector $\mathbf{z}$, conditioning on a domain label $\bar{y}$:

$$\hat{\mathbf{s}} = F(\mathbf{z} \mid \bar{y}). \tag{6}$$

Mapping Network F allows us to stylize a source motion without using a reference motion. We sample $\mathbf{z}$ from the standard Gaussian distribution, with zero mean and unit variance. Our mapping network is comprised of four fully-connected layers, where the domain label is one-hot encoded and concatenated with the latent vector. The size of the domain label vector is the same as the number of domains via one-hot encoding. F can generate diverse style codes by varying the latent vector. The conditional setup enables F to learn style representations for all domains effectively.

Style discriminator. Our discriminator D learns to discriminate multiple styles. Its network components consist of an ST-GCN-based shared architecture with multiple linear output branches. Each domain-specific branch D_y outputs a binary value depending on whether an input motion is a real motion from the domain y or a fake one synthesized by G.

The detailed structure and implementation of our network architecture are presented in Appendix A.

6.2 Training Objectives

Our networks are trained by minimizing the following loss functions.

Adversarial loss. The motion stylizer G is trained to synthesize a motion $\hat{\mathbf{m}} = G(\mathbf{m}, \hat{\mathbf{s}})$ while the discriminator D is trained to distinguish real motions from the synthesized ones by optimizing the following loss function:

$$\mathcal{L}_{adv} = \mathbb{E}_{\mathbf{m},y}\left[\log\ D_y(\mathbf{m})\right] + \mathbb{E}_{\mathbf{m},\bar{y},\mathbf{z}}\left[\log\left(1 - D_{\bar{y}}(\hat{\mathbf{m}})\right)\right] \tag{7}$$

With this adversarial loss, we penalize D for misclassifying fake examples of G and G for failing to fool D. Meantime, F learns to map $\mathbf{z}$ to $\hat{\mathbf{s}}$ that corresponds to the domain $\bar{y}$, and E learns to encode the style of $\overline{\mathbf{m}}$ into $\hat{\mathbf{s}}$.

Cycle consistency loss. We enforce G to preserve the semantic consistency by applying the cycle consistency loss $\mathcal{L}_{cyc}$ [Zhu et al. 2017]. The cyclic loss encourages G to remap the output motion $\hat{\mathbf{m}}$ to the source motion $\mathbf{m}$ using the estimated style code $\mathbf{s} = E_y(\mathbf{m})$. This loss function allows the model to reliably transform the motion style while maintaining the domain-invariant characteristics of $\mathbf{m}$.

$$\mathcal{L}_{cyc} = \mathbb{E}_{\mathbf{m},y,\bar{y},\mathbf{z}}\left[\ \|\mathbf{m} - G(\hat{\mathbf{m}}, \mathbf{s})\|_2\ \right] \tag{8}$$

Identity loss. We define an additional loss function that encourages identity mapping of G through Equation 9. Given an input motion with its style $\mathbf{s}$, this loss function ensures that G produces the identical motion to the original.

$$\mathcal{L}_{id} = \mathbb{E}_{\mathbf{m},y} \left[\|\mathbf{m} - G(\mathbf{m}, \mathbf{s})\|_2 \right] \tag{9}$$

Style reconstruction loss. The network G learns to utilize $\hat{\mathbf{s}}$ for style deformation by the style reconstruction loss (Equation 10). This loss function guarantees that the target style is reconstructed from the stylized output $\hat{\mathbf{m}}$. Thereby, G learns to transform the input motion while reflecting the stylistic details in the reference motion.

$$\mathcal{L}_{sty} = \mathbb{E}_{\mathbf{m},\overline{y},\mathbf{z}} \left[\left\| \hat{\mathbf{s}} - E_{\overline{y}}(\hat{\mathbf{m}})) \right\|_1 \right] \tag{10}$$

Diversity-sensitive loss. In this work, we use a diversity-sensitive loss [Yang et al. 2019] to avoid the mode collapsing problem, in which the generator produces a deterministic output (i.e., single mode). We regularize the generator to address this problem by maximizing the following loss term:

$$\mathcal{L}_{ds} = \mathbb{E}_{\mathbf{m},\overline{y},\mathbf{z}_1,\mathbf{z}_2} \left[\| G(\mathbf{m}, \hat{\mathbf{s}}_1) - G(\mathbf{m}, \hat{\mathbf{s}}_2) \|_1 \right] \tag{11}$$

where $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$ are two different style codes generated by F using $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{Z}$ or E using $\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2 \in \mathcal{M}_A$. With this loss term, we enforce G to generate diverse outputs as the style code varies.

Full loss function. Finally, we perform the following optimization with the above loss functions:

$$\min_{G,F,E} \max_{D} \mathcal{L} = \lambda_{adv}\, \mathcal{L}_{adv} + \lambda_{cyc}\, \mathcal{L}_{cyc} + \lambda_{id}\, \mathcal{L}_{id} + \lambda_{sty}\, \mathcal{L}_{sty} - \lambda_{ds}\, \mathcal{L}_{ds} \tag{12}$$

where $\lambda_{adv}, \lambda_{cyc}, \lambda_{id}, \lambda_{sty}$, and λ_{ds} are weight parameters for each term.

7 EXPERIMENTS AND EVALUATION

This section presents various experiments to evaluate its performance. All experiments are conducted with unseen examples during the training and evaluated on eight distinctive styles: *neutral, angry, depressed, childlike, old, proud, sexy,* and *strutting*. While the length of input motion is fixed to 64 frames during the training phase, it can vary during the test phase as it is allowed by our convolutional network-based architecture.

Metrics. Two metrics are used for ablation analysis and quantitative evaluation; Fréchet Motion Distance (FMD) and recognition accuracy. The FMD is a variant of the Fréchet Inception Distance (FID) [Heusel et al. 2017] that is used to evaluate the fidelity and diversity of image generative models. The FID is calculated as the distance between the two feature distributions of real and fake images, which are extracted from Inception v3 network [Szegedy et al. 2016] trained on ImageNet [Russakovsky et al. 2015]. However, since there is no standard feature extractor for motion, we train an action classifier [Yan et al. 2018] instead to classify the motion content and calculate the FMD between the real and generated motion samples from the feature vectors that are extracted from its final pooling layer. A lower FMD indicates better performance in terms of generation quality.

In addition, the same action classifier is used to measure the recognition accuracy on generated samples under an assumption that a model with a higher degree of content preservation generates more results that are correctly classified as the content of source motion by the action classifier. The higher accuracy means better performance in terms of content preservation.

In the test, we divide $\mathcal{M}_A$ into two halves, $\mathcal{M}_{gen}$ for generation and $\mathcal{M}_{cls}$ for classification. We train the model with $\mathcal{M}_{gen}$ and the action classifier with $\mathcal{M}_{cls}$ on six content types: *walking, running, jumping, kicking, punching* and *transitions*. The action classifier shows 94.28% accuracy on $\mathcal{M}_{gen}$. We validate the classifier for each stylized set made by the trained models.

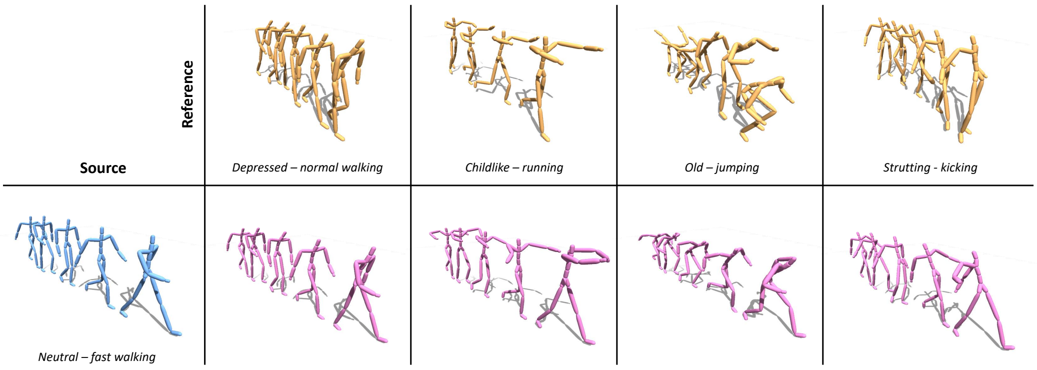


Fig. 4. Transferring styles of various reference motions to *neutral-fast walking* motion.

All results are presented after post-processing by the same scheme used in [Aberman et al. 2020] including foot-sliding removal. In the figures, the source motion is shown with a blue skeleton, the reference motion with a yellow skeleton, and the resulting motion in pink. While our network outputs both the joint position and rotation data, we visualize the resulting motion by using only the joint position data. The supplemental video shows the resulting motions from the experiments.

7.1 Visual Results

Transferring to multiple domains. To examine the ability of our framework to transfer styles to multiple domains, we use the *fast walking* motion from the *neutral* domain as the source motion and perform style transfer using reference motions in various style domains, such as *depressed-normal walking, childlike-running, old-jumping,* and *strutting-kicking.* As shown in Figure 4, the results demonstrate that our framework can translate the distinctive style of the reference motion in each style domain into the source motion, without affecting its key content information, such as the locomotion phase and foot contact timing.

Stylization diversity using random noise. A distinctive feature of our method is that it can generate the desired style code from a randomly sampled latent vector without using any reference motions. Figure 1 shows the diverse results of transferring *neutral* motion to *proud* and *depressed* domains with three different latent codes. The results show that the latent code allows for detailed stylistic control, which is applicable to multiple domains and contents. In *fast walking* motion, we find that the range of arm swing, step length, and torso tilting vary in response to the latent code. In *jumping* motion, on the other hand, the latent code mostly affects the shape of the arms and legs and the jumping height. Notably, an emerged phenomenon is that the same latent code tends to show similar style intensity across domains. For example, $\mathbf{z}_3$ and $\mathbf{z}_2$ in Figure 1 create the highest and lowest stylization intensities, respectively, in both motion contents and both style domains.

Style transfer on long-term heterogeneous data. We test whether our method can stylize a long-term heterogeneous motion. Specifically, we apply our framework trained with $\mathcal{M}_A$ dataset to the heterogeneous dataset $\mathcal{M}_B$. Figure 5 shows the style transfer results of a *neutral* motion over 1000 frames, including *walking* $\longrightarrow$ *jumping* $\longrightarrow$ *transition* $\longrightarrow$ *left kicking* $\longrightarrow$ *right kicking* to *strutting* and *old* domains using a random style code. We find that our style transfer method can be applied to a long sequential motion with multiple contents.

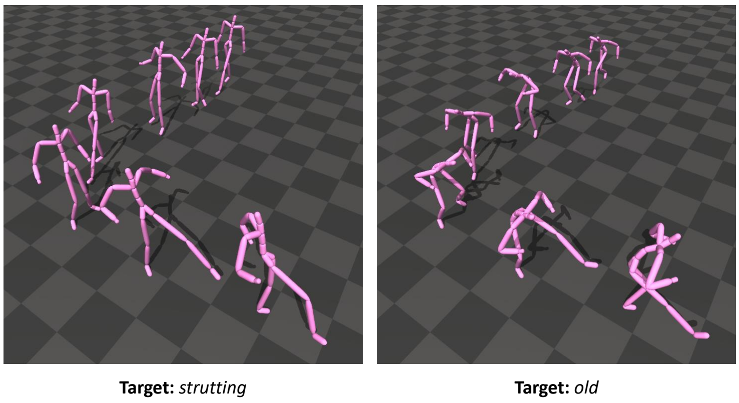


Fig. 5. Style transfer of a heterogeneous *neutral* motion including *walking* → *jumping* → *transition* → *left kicking* → *right kicking* to *strutting* and *old* domains.

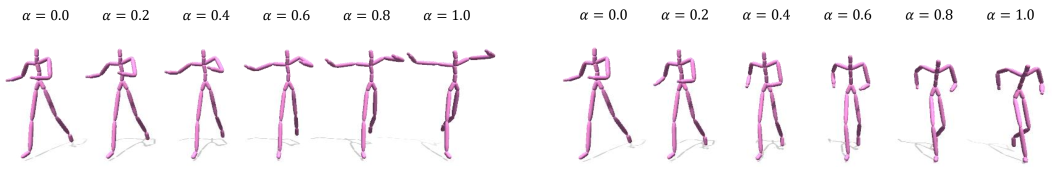


Fig. 6. Style interpolation results. *Running* motion is stylized with interpolated style codes (a) within *childlike* domain and (b) between *childlike* and *old* domains.

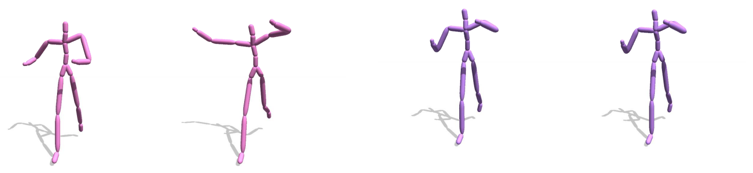
Style interpolation. Our framework allows for the style mixing within-domain and inter-domain by linearly interpolating different style codes. Figure 6a shows *running* motion is stylized with interpolated style codes within *childlike* domain. Figure 6b shows the same motion is transferred with interpolated style codes between *childlike* and *old* domains.

7.2 Ablation Analysis

We conduct an ablation study to analyze the influence of our loss terms. We first experiment by using only the adversarial ($\mathcal{L}_{adv}$) and content reconstruction losses ($\mathcal{L}_{cyc}$, $\mathcal{L}_{id}$), gradually adding losses for style reconstruction ($\mathcal{L}_{sty}$) and style diversification ($\mathcal{L}_{ds}$). We measure FMD and the recognition accuracy on the stylized samples that are generated from random noise for each loss combination. As shown in Table 1, adding $\mathcal{L}_{sty}$ increases the performance by a large margin. However, the FMD and recognition accuracy are slightly degraded when $\mathcal{L}_{ds}$ is added since $\mathcal{L}_{ds}$ encourages the model to generate different motion from each other, leading to the stylized set being largely diversified from the original training set $\mathcal{M}_{gen}$. However, the model can generate diverse but reliable stylization results through $\mathcal{L}_{ds}$. Figure 7 shows the stylization results on *running* motion into *childlike* domain with two randomly sampled latent codes trained with and without $\mathcal{L}_{ds}$ term. Figure 7a shows distinctive results with two latent codes whereas Figure 7b exhibits the mode collapsing problem, resulting in almost the same motions. Once the model is trained with $\mathcal{L}_{ds}$, we can expect that the random noise will control the stylistic variation within a given target domain.

Table 1. Ablation analysis on loss terms

Loss function	FMD↓	Accuracy↑ (%)
$\mathcal{L}_{adv} + \mathcal{L}_{cyc} + \mathcal{L}_{id}$	14.14	57.92
$\mathcal{L}_{adv} + \mathcal{L}_{cyc} + \mathcal{L}_{id} + \mathcal{L}_{sty}$	8.58	77.86
$\mathcal{L}_{adv} + \mathcal{L}_{cyc} + \mathcal{L}_{id} + \mathcal{L}_{sty} + \mathcal{L}_{ds}$	9.09	72.64



(a) Two stylized transfer results using random noises when trained with $\mathcal{L}_{ds}$.

(b) Two stylized transfer results using random noises when trained without $\mathcal{L}_{ds}$.

Fig. 7. Effectiveness of the diversity term $\mathcal{L}_{ds}$. A *running* motion is stylized into *childlike* with two randomly sampled noises. (a) When trained with $\mathcal{L}_{ds}$ term, distinctive results are obtained with different latent codes. (b) Without the term, mode collapsing phenomena may occur.

7.3 Comparison

Qualitative evaluation. We compare the performance of our method in terms of the generation quality and content preservation to the previous methods: [Aberman et al. 2020] using AdaIN and [Holden et al. 2016] using Gram matrix for style transfer. Both methods are different from ours in that they use the 1D convolution over the time domain for feature extraction; therefore, they do not explicitly model the spatial relations among body parts in the motion data.

Figure 8 shows the qualitative comparison of three methods. Previous methods struggle to reliably perform style transfer while preserving the content of source motion, especially when two input motions are non-periodic and spatially irrelevant to each other, such as *neutral-punching* and *angry-kicking*. In particular, the method of [Holden et al. 2016] fails to separate the content of reference motion, resulting in the lifted leg of *angry-kicking* and the elevated arm of *sexy-punching* being translated into stylized motions. On the other hand, our method can reflect the domain-specific styles without harming the semantics of source motion and does not exhibit artifacts such as motion twist or foot floating observed in the results of [Aberman et al. 2020]. This comparison suggests that our approach is more effective at disentangling style and content, therefore providing a higher degree of visual quality and content preservation than the previous method. We attribute this ability of our model to the spatial-temporal motion modeling. We include the full comparison in the supplementary video.

Quantitative evaluation. We quantitatively measure the degree of the generation quality and content preservation of three generative models: [Holden et al. 2016], [Aberman et al. 2020], and our network. We calculate the FMD and recognition accuracy on the stylized set with all possible combinations of source (content) and reference (style) motions created with each model.

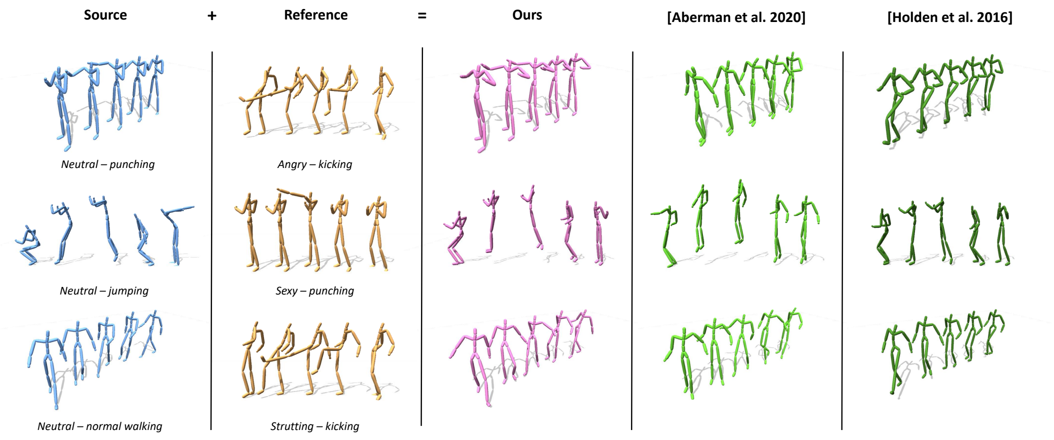


Fig. 8. Comparison between our method, [Aberman et al. 2020], and [Holden et al. 2016].

Table 2. Quantitative evaluation

Methods	Motion modeling	FMD↓	Accuracy↑ (%)
Ours	ST-GCN	**6.82**$^{\pm 0.86}$	**60.86**$^{\pm 2.06}$
[Aberman et al. 2020]	1D CNN	15.86$^{\pm 2.51}$	42.81$^{\pm 2.53}$
[Holden et al. 2016]	1D CNN	14.10$^{\pm 0.65}$	37.99$^{\pm 1.45}$

Results are presented in Table 2. Our model shows superior results to the competing methods in both metrics, suggesting that our method produces higher-quality stylized motions and preserves the motion content more accurately.

8 DISCUSSION

There are a few possible factors that may have contributed to the performance of our framework. Previous works attempted to extract the style along the time axis, which often fails to reflect the spatial dynamics of motion that may be associated with the style. In our study, on the other hand, spatial-temporal modeling of motion is accomplished by graph convolutional operations, allowing more structural and hierarchical features of skeleton data to be extracted from the local to global scale and vice versa via graph pooling and unpooling, whereas 1D CNN does not take into account the correlation between joints.

Moreover, our style encoder and mapping network produce the style code separately so that our generator can solely focus on utilizing the given style code when generating a motion. Lastly, as stated in [Choi et al. 2020], the shared parts of our style encoder and discriminator learn domain-invariant features, which induce the regularization effect so that the model can better generalize over unseen samples.

Our framework has several limitations that need to be addressed in future work. First, the linear and angular velocities of the root are not used in the input or output of the network. Instead, the root trajectory in the generated motion is calculated directly from the source motion. Therefore, applying a high-intensity style to a walking motion through random noise, causing a drastic change of the step length, may result in foot skating artifacts. A promising approach to improving the motion quality would be to disentangle the spaces of the joint pose and velocity to separately stylize the two spaces, as proposed by [Lim et al. 2019].

Second, our framework is only applicable to the motions with explicit style labels. This limitation reduces the range of motion data available because most motion databases do not come with style labels. In addition, many styles are difficult to label with only a few words. An important future work to overcome this limitation is to develop a network capable of fully unsupervised learning to learn style space from the motion data in the wild.

9 CONCLUSION

In this paper, we have proposed a deep learning-based framework for character motion style transfer. Our framework is a generative architecture that learns mappings across multiple style domains and generates a variety of stylized outputs. We construct a spatio-temporal graph on a motion sequence and apply spatial-temporal graph convolutional operations to the graph-based data. Through experiments, we showed that our framework can perform reliable style transfer between motions with arbitrary contents while ensuring a high degree of visual quality, stylistic diversity, and content preservation.

REFERENCES

Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2020. Unpaired motion style transfer from video to animation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 64–1.

Andreas Aristidou, Qiong Zeng, Efstathios Stavrakis, KangKang Yin, Daniel Cohen-Or, Yiorgos Chrysanthou, and Baoquan Chen. 2017. Emotion control of unstructured dance movements. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. 1–10.

Kyungjune Baek, Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Hyunjung Shim. 2020. Rethinking the truly unsupervised image-to-image translation. *arXiv preprint arXiv:2006.06500* (2020).

Jacky CP Chan and Edmond SL Ho. 2021. Emotion Transfer for 3D Hand and Full Body Motion Using StarGAN. *Computers* 10, 3 (2021), 38.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. 2020. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8188–8197.

Jeff Donahue and Karen Simonyan. 2019. Large scale adversarial representation learning. *arXiv preprint arXiv:1907.02544* (2019).

Yuzhu Dong, Andreas Aristidou, Ariel Shamir, Moshe Mahler, and Eakta Jain. 2020. Adult2child: Motion Style Transfer using CycleGANs. In *Motion, Interaction and Games*. 1–11.

Han Du, Erik Herrmann, Janis Sprenger, Klaus Fischer, and Philipp Slusallek. 2019. Stylistic locomotion modeling and synthesis using variational generative models. In *Motion, Interaction and Games*. 1–10.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015).

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2414–2423.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

Daniel Holden, Ikhsanul Habibie, Ikuo Kusajima, and Taku Komura. 2017. Fast neural style transfer for motion data. *IEEE computer graphics and applications* 37, 4 (2017), 42–49.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Eugene Hsu, Kari Pulli, and Jovan Popović. 2005. Style translation for human motion. In *ACM SIGGRAPH 2005 Papers*. 1082–1089.

Xun Huang and Serge Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*. 1501–1510.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal Unsupervised Image-to-Image Translation. arXiv:1804.04732 [cs.CV]

Leslie Ikemoto, Okan Arikan, and David Forsyth. 2009. Generalizing motion edits with gaussian processes. *ACM Transactions on Graphics (TOG)* 28, 1 (2009), 1–12.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1125–1134.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations.* 1–15.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. 2018. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European conference on computer vision (ECCV).* 35–51.

Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. 2020. Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision* 128, 10 (2020), 2402–2417.

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. 2017. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036* (2017).

Jongin Lim, Hyung Jin Chang, and Jin Young Choi. 2019. PMnet: Learning of Disentangled Pose and Movement for Unsupervised Motion Retargeting.. In *BMVC.* 136.

Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. 2019. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 10551–10560.

Ian Mason, Sebastian Starke, He Zhang, Hakan Bilen, and Taku Komura. 2018. Few-shot Learning of Homogeneous Human Locomotion Styles. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 143–153.

Jianyuan Min, Huajun Liu, and Jinxiang Chai. 2010. Synthesis and editing of personalized stylistic human motion. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games.* 39–46.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Harrison Jesse Smith, Chen Cao, Michael Neff, and Yingying Wang. 2019. Efficient neural networks for real-time motion style transfer. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 2 (2019), 1–17.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2818–2826.

Graham W Taylor and Geoffrey E Hinton. 2009. Factored conditional restricted Boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning.* 1025–1032.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2017. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 6924–6932.

Xinyao Wang, Liefeng Bo, and Li Fuxin. 2019. Adaptive wing loss for robust face alignment via heatmap regression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 6971–6981.

Shihong Xia, Congyi Wang, Jinxiang Chai, and Jessica Hodgins. 2015. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tiangchen Zhao, and Honglak Lee. 2019. Diversity-Sensitive Conditional Generative Adversarial Networks. In *International Conference on Learning Representations.* https://openreview.net/forum?id=rJliMh09F7

M Ersin Yumer and Niloy J Mitra. 2016. Spectral style transfer for human motion between independent actions. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–8.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision.* 2223–2232.

A APPENDIX: IMPLEMENTATION

In this section, we provide details about implementing our framework.

A.1 Architectural Details

Table 3 shows the details of our network architecture. Here, the size of spatial-temporal graph convolution filter is denoted as $k_t \times k_s$, where k_t and k_s are the kernel sizes along the temporal and spatial dimensions, respectively.

Our motion stylizer G has two ST-GCN residual blocks with IN layers (ST-Resblk) for G_{enc} and two ST-GCN residual blocks with AdaIN layers (ST-AdaINResblk) for G_{dec}. The leaky ReLU is used as the activation function. G_{enc} takes a source motion $\mathbf{m}$ as input and downsample it via graph pooling to produce the feature map represented by the graph $\mathcal{G}_3$ with $|\mathcal{V}_3| = N/4 \times T/4$. G_{dec} receives the output of G_{enc} with a style code and upsamples the motion data via graph unpooling, while injecting the style code in all AdaIN layers in ST-AdaINResblk.

The style encoder E consists of two ST-Resblks followed by D linear output branches, where D is the number of style domains. Given a reference motion $\overline{\mathbf{m}}$, E encodes the style code from $\overline{\mathbf{m}}$ via graph pooling. Here, the size of the style code is set to 64, and thus the output dimension in Table 3b becomes $64 \times \mathtt{D}$, which represents the dimension of the created style codes for all D domains.

Our mapping network F consists of four fully connected layers. F creates a style code using a latent code and a target domain label as inputs. The one-hot encoded domain label and the latent vector are concatenated after passing through FC1a and FC1b, respectively. Here, the dimensionality of the hidden layer (the output of FC3) is set to 1024.

Similarly to E, the discriminator D consists of two ST-Resblks with D output branches. D downsamples the input motion via graph pooling and outputs a binary value depending on its classification between real and fake. The output dimension in Table 3d is set to D, which indicates the dimension of the classifications for all domains.

A.2 Implementation Details

Our framework is implemented in Pytorch and optimized by the Adam optimizer [Kingma and Ba 2015] with $\beta_1 = 0.99$ and $\beta_2 = 0.999$. We train our model for 100K iterations with a batch size of 8. Style Encoder E and Mapping Network F are used alternately during training. The learning rates were set to 10^{-4} for G, 10^{-6} for D, E, and 10^{-4} for F. We set $\lambda_{adv}, \lambda_{cyc}, \lambda_{id}, \lambda_{sty}$, and $\lambda_{ds} = 1$. We initialize the weights of network modules using He initialization. The whole training session takes 10 hours for $\mathcal{M}_A$ using a single NVIDIA GTX Titan XP GPU.

Type	Filter size	Actv.	Norm.	Resample	Output shape
$\mathbf{m}$	-	-	-	-	$T \times N \times 7$
Conv1x1	1×1	-	-	-	$T \times N \times 64$
ST-Resblk	3×3	LReLU	IN	Graph pool	$T/2 \times N/2 \times 128$
ST-Resblk	3×3	LReLU	IN	Graph pool	$T/4 \times N/4 \times 256$
ST-AdaINResblk	3×3	LReLU	AdaIN	Graph unpool	$T/2 \times N/2 \times 128$
ST-AdaINResblk	3×3	LReLU	AdaIN	Graph unpool	$T \times N \times 64$
Conv1x1	1×1	-	-	-	$T \times N \times 7$

(a) Motion stylizer G.

Type	Filter size	Actv.	Resample	Output shape
$\overline{\mathbf{m}}$	-	-	-	$T \times N \times 7$
Conv1x1	1×1	-	-	$T \times N \times 64$
ST-Resblk	3×3	LReLU	Graph pool	$T/2 \times N/2 \times 128$
ST-Resblk	3×3	LReLU	Graph pool	$T/4 \times N/4 \times 256$
LReLU	-	-	-	$T/4 \times N/4 \times 256$
Conv2D	$T/4 \times N/4$	-	-	$1 \times 1 \times 256$
LReLU	-	-	-	$1 \times 1 \times 256$
Reshape	-	-	-	256
FC	64	-	-	$64 \times \mathtt{D}$

(b) Style Encoder E.

Type	Actv.	Output shape
$\mathbf{z}$, $\overline{y}$	-	16, $\mathtt{D}$
FC1a	ReLU	256
FC1b	ReLU	256
Concat	-	512
FC2	ReLU	512
FC3	ReLU	1024
FC4	ReLU	64
Tanh	-	64

(c) Mapping network F.

Type	Filter size	Act.	Resample	Output shape
$\mathbf{m}$ or $\widehat{\mathbf{m}}$	-	-	-	$T \times N \times 7$
Conv1x1	1×1	-	-	$T \times N \times 64$
ST-Resblk	3×3	LReLU	Graph pool	$T/2 \times N/2 \times 128$
ST-Resblk	3×3	LReLU	Graph pool	$T/4 \times N/4 \times 256$
LReLU	-	-	-	$T/4 \times N/4 \times 256$
Conv2D	$T/4 \times N/4$	-	-	$1 \times 1 \times 256$
LReLU	-	-	-	$1 \times 1 \times 256$
Conv1x1	1×1	-	-	$1 \times 1 \times \mathtt{D}$
Reshape	-	-	-	$\mathtt{D}$
FC	$\mathtt{D}$	-	-	$\mathtt{D}$

(d) Discriminator D.

Table 3. Architecture of our motion style transfer framework.