 doi:10.20944/preprints202011.0039.v1

Article

Generative Model for Skeletal Human Movements based on conditional DC-GAN applied to pseudo-images

Wang Xi [1], Guillaume Devineau [2], Fabien Moutarde [2], and Jie Yang [1]

[1] Shanghai Jiao Tong University, Shanghai, China; luckyxi94, jieyang@sjtu.edu.cn
[2] MINES ParisTech, Université PSL, Centre de Robotique, 75006 Paris, France; guillaume.devineau, fabien.moutarde@mines-paristech.fr

Abstract: Generative models for images, audio, text and other low-dimension data have achieved great success in recent years. Generating artificial human movements can also be useful for many applications, including improvement of data augmentation methods for human gesture recognition. The object of this research is to develop a generative model for *skeletal* human movement, allowing to control the action type of generated motion while keeping the authenticity of the result and the natural style variability of gesture execution. We propose to use a conditional Deep Convolutional Generative Adversarial Network (DC-GAN) applied to pseudo-images representing skeletal pose sequences using Tree Structure Skeleton Image format. We evaluate our approach on the 3D-skeleton data provided in the large NTU RGB+D public dataset. Our generative model can output qualitatively correct skeletal human movements for any of its 60 action classes. We also quantitatively evaluate the performance of our model by computing Frechet Inception Distances, which shows strong correlation to human judgement. Up to our knowledge, our work is the first successful class-conditioned generative model for human skeletal motions based on pseudo-image representation of skeletal pose sequences.

Keywords: generative model; human movement; conditional Deep Convolutional Generative Adversarial Network; GAN; spatio-temporal pseudo-image

1. Introduction

Human movement generation, although less developed than for instance text generation, is an important applicative field for sequential data generation. It is difficult to get enough effective human movement dataset due to its complexity – human movement is the result of both physical limitation (torque exerted by muscles, gravity, moment preservation) and the intentions of subjects (how to perform an intentional motion) [1]. Generating artificial human movement data enables data augmentation, which should improve the performance of models in all the fields of study on human movement: classification, prediction, generation, etc. Also, it will have important applications in related domains. Imagine in the computer game, each character will not run and jump in the exactly the same way but in their own style, which makes the actions in the game closer to the reality.

The object of this research is to develop a generative model for *skeletal* human movements, allowing to control the action type of generated motion while keeping the authenticity of the result and the natural style variability of gesture execution. Skeleton-based movement generation has recently become an active topic in computer vision, owing to the potential advantages of skeletal representation, and to recent algorithms for automatically extracting skeletal pose sequences from videos. Previously, most of the methods and researches on human gestures, including human movement generation, used RGB image sequences, depth image sequences, videos or some fusion of these modalities as input data. Nowadays, it is easy to get relatively accurate 2D or 3D human skeletal pose data thanks to state-of-the-art human pose estimation algorithms and high-performance sensors. Compared to dense images approaches, skeletal data has the following advantages: (a) lower

 doi:10.20944/preprints202011.0039.v1

computation requirement; (b) robustness under situations such as complicated background and changing body scales, viewpoints, motion speed, etc. Besides these advantages, skeletal data has the three following main characteristics [2]:

- spatial information: strong correlations between adjacent joints, which makes it possible to learn about body structural information within a single frame (intra-frame);
- temporal information: can learn about temporal correlation between frames (inter-frame);
- co-occurrence relationship between spatial and temporal domains when taking joints and bones into account.

Skeletal-based human movement generation can be formalized as a continuous multivariate time-series problem, and is therefore often tackled using Recurrent Neural Network (RNN). In our work, we however decided to rather use representation of each skeletal pose sequence as a spatio-temporal pseudo-image, with space and time as two dimensions of the image. The advantage is that we can directly apply state-of-the-art generative models for images, which are more mature, and simpler in terms of the network architecture, than the ones tailored for continuous time-series generation. Thus, it is important to find a good pseudo-image representation for 3D skeleton data. The related works will be further developed in the next part. The key issues to be solved include: how to generate realistic human movements, how to control the output (action, style, etc.), how to evaluate the model effectively.

Our work proves that as long as we apply a well-adapted pseudo-image representation for 3D skeleton data, such as Tree Structure Skeleton Image (TSSI), we can generate human movement by simple standard image generative model such as Deep Convolutional Generative Adversarial Network (DC-GAN, [3]). Furthermore, by introducing a conditional label, we control the output action type. We also evaluate our generative models by calculating Frechet Inception Distance.

The paper is organized as follows: section 2 presents published works related to Human movement generative models and to pseudo-image representations for skeletal pose sequences; section 3 explains the details of our approach; section 4 analyzes the results we have obtained; and section 5 presents our conclusions and perspectives.

2. Related Works

In this section, we present some of the already published works most related to our work, therefore concerning: 1/ generative models for skeletal human movements; and 2/ pseudo-image representation for skeletal pose sequences.

2.1. Generative models for skeletal Human movements

A family of methods based on deep Recurrent Neural Networks (RNNs) have shown good performance on generative tasks for human skeletal movements. For example, Fragkiadaki et al. [4] proposed in 2015 the Encoder-Recurrent-Decoder (ERD) model that uses curriculum learning and incorporates representation learning in the architecture. In 2016, Jain et al. introduced the concept of Structural-RNN [5], which manually encodes the semantic similarity between different body parts based on spatio-temporal graph. The reason why RNN is a popular generative model for sequential data is that RNNs typically possess both a richly distributed internal state representation and flexible non-linear transition functions. This expressive power and the ability to train via error back-propagation are the key reasons why RNNs have gained popularity as generative models for highly structured sequential data. Inspired by previous works on RNN, Martinez et al. [1] proposed in 2017 a human motion prediction model using sequence-to-sequence (seq2seq) architecture with residual connections, which turned out to be a simple but state-of-the-art method on prediction of short-term dynamics of human movement. Chung et al. [6] explored the inclusion of latent random variables into the hidden state of an RNN by combining the elements of the variational autoencoder. They argued that through the use of high-level latent random variables, their variational RNN (VRNN)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

can model the kind of variability observed in highly structured sequential data such as natural speech.

Besides the models based on RNNs and VAEs, Generative Adversarial Networks (GAN) [7], which achieve great success in generative problem, are recently one of the most active machine-learning research topics. GANs have proved their strength in generation of images and some types of temporal sequences. For example, Deep Convolutional GAN (DC-GAN) [3] proposed a strided conv-transpose layer and introduced a conditional label, which allows to generate impressive images. WaveNet [8], applying dilated causal convolutions that effectively operate on a coarser scale, can generate raw audio waveforms and transform text to speech which human listeners can hardly tell it from the true voice. Recently, several researches on human movement prediction and generation models are based on GAN. In 2018, the Human Prediction GAN (HP-GAN) [9] of Barsoum et al. used also a sequence-to-sequence model as its generator and additionally designed a critic network to calculate a custom loss function inspired by WGAN-GP [10]. Kiasari et al. [11] approached from another way, combining an Auto-Encoder (AE) and a conditional GAN to generate a sequences of human movement.

2.2. Pseudo-image representation for skeletal pose sequences

In many skeleton-based human movement recognition studies, researchers use Convolutional Neural Networks (CNN) because of their efficiency and excellent ability to extract high level information. However, the most common and developed CNNs use 2D convolutions and therefore generally focus on image-based tasks. Meanwhile, human movement problems, including recognition and, the main task of this paper, generation, are definitely a heavily temporal problem. Thus, it is challenging to balance and capture both spatial and temporal information in CNN-based architecture [2].

In general, to meet the need of CNN's input, 3D skeleton sequence data should be transformed into a pseudo-image. Nevertheless, a decent representation that contains both spatial and temporal information is not simple. Many researchers encode the skeleton joints to *multiple* 2D pseudo-images then feed them into the model. Wang et al. [12] introduced the Joint Trajectory Maps (JTM), which represent spatial configuration and dynamics of joint trajectories into three texture images through color encoding. Li et al. [13] proposed a translation-scale invariant image mapping. They pointed out that JTM encoding method is dataset dependent and not robust to the translation and scale variation of human activity. To tackle this problem, they divide all human skeleton joints into five main parts according to human physical structure then map them to images. However, they merely focus on the coordinates of isolated joints, therefore ignoring the spatial relationships between joints and only implicitly learning the movement representations. Take the action "walking" for example, not only the legs and foots should be studied, but also arms and the other body parts should be considered. In a word, human body need to be treated as a whole. To solve the problem, Li et al. [14] proposed in 2019 an effective method to learn comprehensive representations from skeleton sequences by using Geometric Algebra. Firstly, a frontal orientation based spatio-temporal model is constructed to represent the spatial configuration and temporal dynamics of skeleton sequences, which owns the robustness against view variations. Then shape-motion representations which mutually compensate are learned to describe skeleton actions comprehensively. Liu et al. [15] used an enhanced skeleton visualization to represent the skeleton data. SkeleMotion [16] directly encodes data by using orientation and magnitude to provide information regarding the velocity of the movement in different temporal scales. The Tree Structure Reference Joint Image (TSRJI) of Caetano et al. [17] combines the use of reference joints and a tree structure skeleton: while the former incorporates different spatial relationships between the joints, the latter preserves important spatial relations by traversing a skeleton tree with a depth-first order algorithm.

Even if these skeleton sequence representations have made great effort to encode as much information as possible in one or a series of "images", it is still hard to learn global co-occurrences. When these pseudo-images are fed to CNN-based models, only the neighboring joints within the convolutional kernel are considered to learn local co-occurrence features. Thus, in 2018, Li et al. [18]

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

proposed an end-to-end co-occurrence feature learning framework, where features are aggregated gradually from point-level features to global co-occurrence features.

3. Materials and Methods

To generate human skeletal motions, we use a *conditional* Deep Convolutional Generative Adversarial Network (DC-GAN) applied to pseudo-images representing skeletal pose sequences using Tree Structure Skeleton Image format. We evaluate our approach on the 3D-skeleton data provided in the large NTU RGB+D public dataset, and use Frechet Inception Distances as quantitive evaluation. The details of database, pseudo-image representation, data preparation, our generative model and the process of training and evaluation will be provided in this section.

3.1. NTU RGB+D dataset

NTU RGB+D [19] is a large-scale public dataset for 3D human activity analysis. It contains 60 different action classes including daily, mutual, and health-related actions. 56880 samples, 40 human subjects and 80 camera views, this dataset is much larger than other available datasets for RGB+D (RGB videos and depth sequence) action analysis. In our research, we only use the skeleton data (3D locations of 25 major body joints) provided in the dataset.

3.2. Tree Structure Skeleton Image (TSSI)

In our approach, we transform the skeletal pose sequences into Tree Structure Skeleton Image (TSSI) pseudo-images [20]. The principle of TSSI is to create one image for each sequence of skeletons. In these spatio-temporal pseudo-images, horizontal axis corresponds to joints' positions, and vertical axis to time. The joints' positions are encoded as one channel for x, one for y, and one for z, therefore obtaining a color image with Red corresponding to x, Green to y, and Blue to z. One of the important particularity of TSSI is to reorder the skeletons using depth-first tree traversal order, so that neighboring columns in spatio-temporal images are spatially related in original skeleton graph. The spatial relations between connected skeleton joints can thus be well preserved. TSSI can be trained with semantical meaning in normal generative model for image, where the spatial and temporal features are meant to be learnt at the same time. Also, comparing with other pseudo-image method, TSSI is relatively simple.

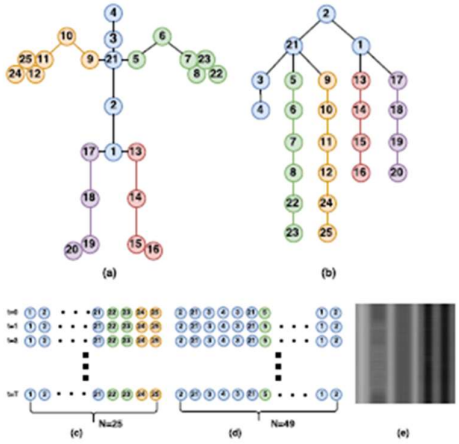

Figure 1. Illustration (from [20]) of Tree Structure Skeleton Image (TSSI): (**a**) Skeleton structure and order in NTU RGB+D; (**b**) Skeleton tree for TSSI generating; (**c**) Joint arrangements of naive skeleton images; (d) Joint arrangements of TSSI, and (**e**) An example frame of TSSI. Different colors represent different body parts.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

3.3. Data preparation

First, the original sequences of skeletons are transformed into TSSI pseudo-images representation format. In this way, each sequence of poses becomes one spatio-temporal image of size (time, joints in TSSI order, 3), where 3 stands for three dimensions of each joint. To unify the sizes of training data, the spatio-temporal images are all resized to 64 x 64 pixels, using bilinear interpolation. Finally, these images are saved by class.

3.4. Conditional Deep Convolution Generative Adversarial Network (conditional DC-GAN)

In this paper, we use a *conditional* Deep Convolution Generative Adversarial Network (conditional DC-GAN), based on DC-GAN [3] and cGAN [21]. In the original DC-GAN architecture, the discriminator is made up of strided convolution layers, batch norm layers, and LeakyReLU activations. Its input is a 64x64x3 (Height x Width x Channel) image, and the output is a scalar probability that the input is from the real data distribution. The generator is comprised of convolutional-transpose layers, batch norm layers, and ReLU activations. Its input is a latent vector z, that is drawn from a standard normal distribution, and the output is a 64x64x3 RGB image. The strided conv-transpose layers allow the latent vector to be transformed into a volume with the same shape as an image.

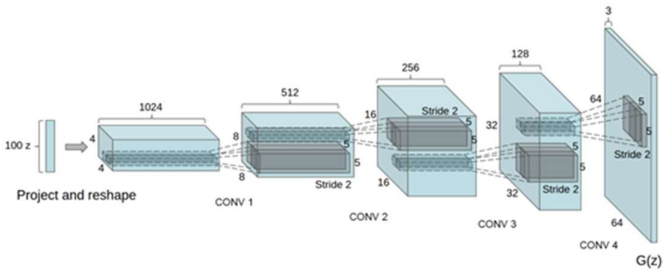

Figure 2. Architecture of Generator in DC-GAN [3]

In the generator, we use upsample layer and convolutional layer, instead of original convolutional-transpose layer, in order to avoid the checkerboard artifacts [22]. If a standard deconvolution (use convolutional-transpose layer) is used to upscale from a lower resolution image to a higher one, it uses every point in the small image to "paint" a square in the larger one. Here comes the problem of "uneven overlap", that is to say when these "squares" in the larger image overlaps. In particular, deconvolution has uneven overlap when the kernel size is not divisible by the stride. These overlapped pixels create an unwanted checkerboard pattern on generated images.
Thus, we instead use upsample layers and convolutional layers to achieve the "deconvolution" operation. We resize the image, using upsample layer with bilinear interpolation (or nearest-neighbor interpolation), and then do a convolutional layer. For best result, before convolutional layer, we add a RelectionPad2d layer of size 1 to avoid boundary artifacts. Details are provided in Appendix A.

In order to control the output action, the model receives the class label at input for both generator and discriminator. The generator receives as input a random noise (1 x 1 x Z) and a one-hot conditional label (1 x 1 x C), where Z is the dimension of z, and C the number of classes. By default, we choose Z=100 and C=60. They are concatenated and passed to a convolutional layer, with a 1 x 1 kernel. The output channel is 8192 (= 4 x 4 x 512). Then, the output data is resized to the shape of 4 x 4 x 512. Finally, we pass the data to several upscaling blocks (Upsample + Conv2D +BatchNorm, except for the output layer), which is exactly the same as the ones in original DC-GAN. The reason for not directly resizing the input data (the concatenation of random noise and one-hot label) is that when the data size is 1 x 1, it can only be upsampled to an equal-valued data. Also, we wish to keep the input noise size as 1 x 1 x Z. Therefore, we decided to add an extra convolutional layer and the resizing step to make it works. In the discriminator, the label is encoded as an image with C channels (C for class number). The label is one-hot encoded along the dimension of channel. To be more specific, the label of the n^{th} action will be a H x W x C sized data, with all 1 on the n^{th} channel and 0 for the rest. At input layer, the image and the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

label will be concatenated: every pixel of the image will be concatenated with a one-hot label in the direction of channel. Thus, the input channel size is 3+C. The architecture of the discriminator consists in several convolutional layers and batch normalization layers (except for the output layer) as in the original architecture of DC-GAN.

3.5. Loss function and training process

The loss function for GAN is defined in [7] as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}}(x)[\log D(x)] + \mathbb{E}_{x\sim p_z}(z)[\log(1 - D(G(x)))] \quad (1)$$

Since a GAN has a generator and a discriminator, we will have two losses: *Loss_D* and *Loss_G*. We use the Binary Cross Entropy (BCE) as loss functions. *Loss_D* will be calculated as the sum of *errD_real* = $-\log(D(x))$ and *errD_fake* = $-\log(1 - D(G(z)))$, representing the BCE of the prediction to the reality when D receives real inputs and fake inputs. *Loss_G* = $-\log(D(G(z)))$, calculates the BCE of the "false positive" prediction when D receives fake inputs.

The Discriminator and the Generator are trained and updated one after another, with Adam as optimizer at learning rate = 0.0002 by default.

3.6. Model Evaluation - Fréchet Inception Distance(FID)

To evaluate the model, we use Fréchet Inception Distance (FID) [23]. FID is a measure of similarity between two datasets of images. It was shown to correlate well with human judgement of visual quality, and is most often used to evaluate the quality of samples of GAN. FID is first proposed and used in [24] to improve the evaluation, being more consistent than the Inception Score [25]. The FID metric calculates the distance between two distributions of images on level of features. As in [25], we compute FID mesures using the features extracted from a pre-trained Inception V3 model [26]. FID is calculated by this function[23]:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \quad (2)$$

μ_r, μ_g are the mean values of features of real and fake images
Σ_r, Σ_g are the covariance matrix of features of real and fake images

Since we would like to evaluate the performance both by class of action and by all actions in the future, we prepare the statistics for each class of action and the whole dataset. When calculating FID, we only need to generate a certain number of fake samples to get its statistics μ_g and Σ_g. Notice there is a tradeoff between the fake sample numbers and computational cost of evaluation. The more the generated samples are, the better the distribution of fake samples are presented. Meanwhile, it spends more time and calculation power. Thus, it is crucial to decide the number of the samples. We choose to generate 250 samples for each class of action.

4. Results

4.1. Qualitative result for generated skeletal actions

We first analyze *qualitatively* the results of our generative. Figure 3 shows four examples of actions generated by our model (with default hyperparameter setting) trained after 200 epochs: (a) sitting down; (b) standing up; (c) hand waving; and (d) kicking something. As can be seen, the generated pose sequences match qualitatively and visually with the condition label input in Generator. The respective characteristics of class of action, such as the movement of body parts, are correctly learnt by model, and the generated movements are continuous and smooth.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

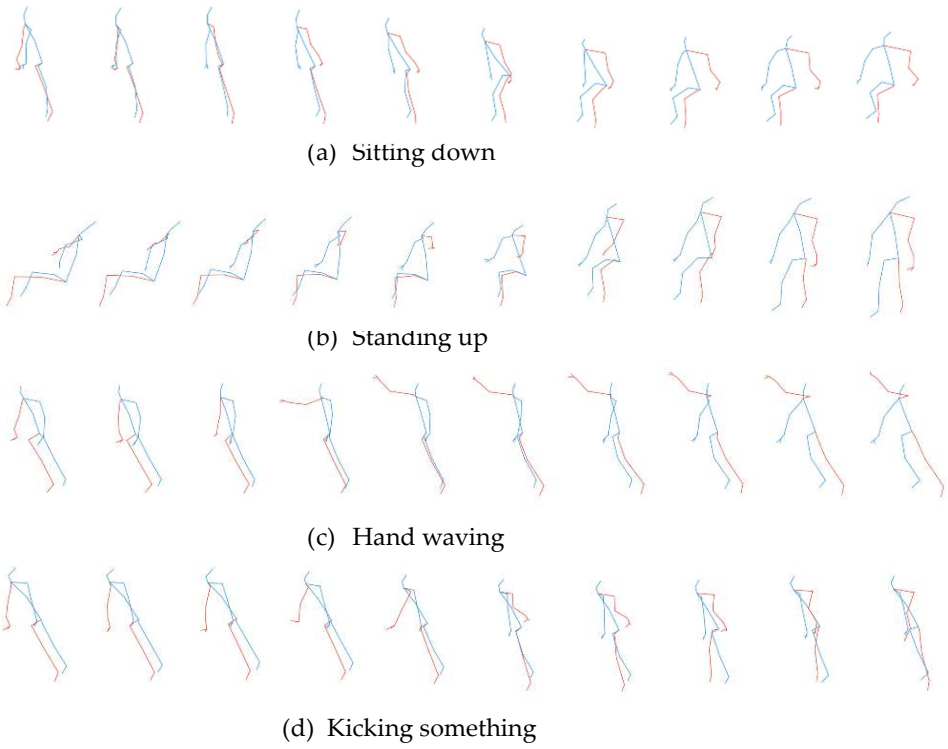

Figure 3. Examples of actions generated by our model. (a) Sitting down; (b) Standing up; (c) Hand waving; and (d) Kicking something

4.2. Quantitative Model Evaluation using Fréchet Inception Distance (FID)

At each evaluation, we record the FID score for each class of action, their average value and FID score of total datasets. Figure 4 shows the action "standing up" generated by the model at 150th epoch and 200th epoch. The FID score is much smaller at 200th epoch, and its generated data is better comparing two sequences by visual. This confirms that the FID is clearly a good mesure of the quality of generated pose sequences.

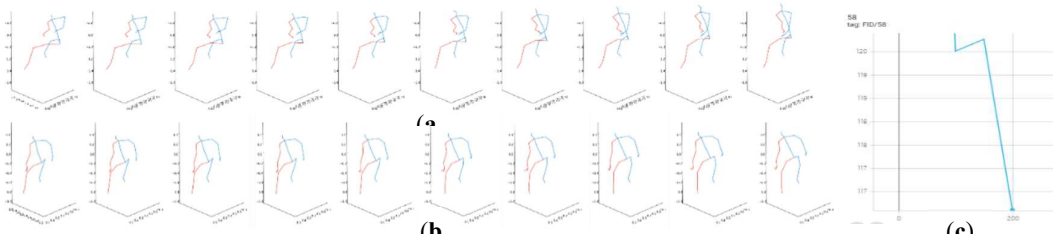

Figure 4. Check the consistency of the performance and FID score. The action "standing up" generated by the model after **(a)** 150 epoch's training, and **(b)** 200 epoch's training. **(c)** FID curve for this action

We now show in Figure 5 some FID scores during the training of our model, evaluating every 50 epochs. The first two rows of curves stand for 8 different actions and the last two curves are the class average FID and the FID for the entire dataset. For most actions, the FID score globally decreases as training goes on. It theoretically means that these types of action generated are closer to the real samples, in terms of realness and diversity. We however note that for some action classes the FID reincreases after the 150th epoch. The same trend is visible on the average FID score. This means that, in general, for one class of action, the results are better for the model trained after 150 epochs than

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

after 200 epochs. When regarding all actions as a whole, we can drive the same conclusion with the total FID score.

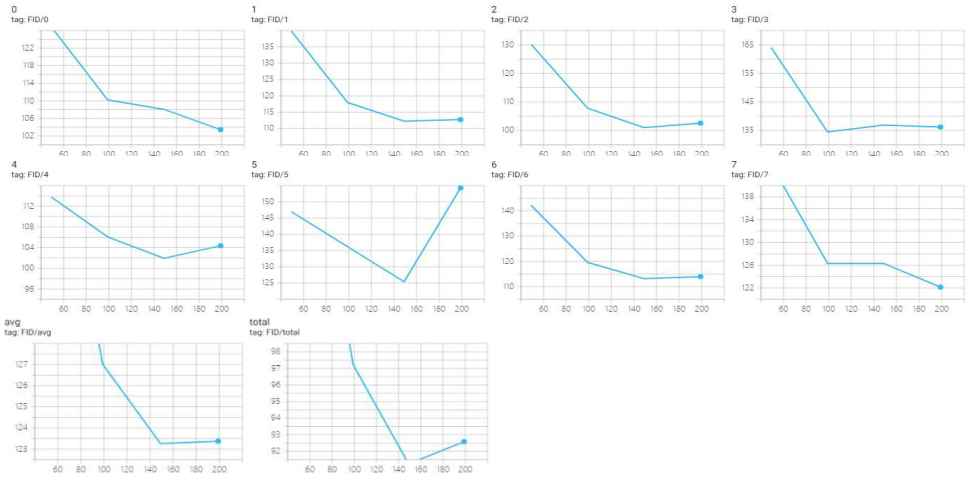

Figure 5. FID curves evaluated on: **(0-7)** 8 different actions, **(avg)** their average score, and **(total)** the entire dataset.

4.3. Importance of the modification of generator using upsample + convolutional layers

As explained in §3.4, we modified the deconvolution operation from original DC-GAN, in order to avoid checkerboard artifacts that arise when using simply convolutional-transpose layers. To tackle this problem, we apply the upsample+convolutional layer method to realize the deconvolution operation. In Figure 6, we compare the learning curves (*Loss_D* and *Loss_G*) of our model (orange) with original c-DC-GAN model (blue), training with the same input. Our model converges much faster than the original one, means the new model is much easier to train.

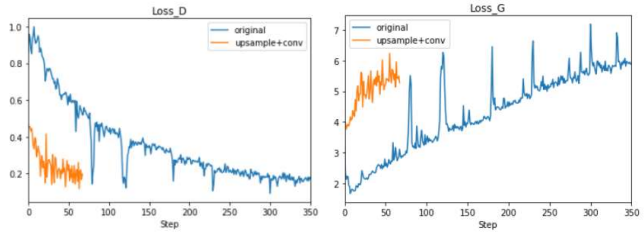

Figure 6. Learning curves of our model (using upsample and convolutional layers) (orange) and original method (using convolutional-transpose layers) (blue). **Loss_D is** Discriminator loss calculated as the average of losses for the all real and all fake samples**. Loss_G** is average Generator loss.

Furthermore, we visually inspect some generated samples in order to check the improvement. On Figure 7, we compare: a real input sample on Figure 7(a); a fake sample generated by the original c-DCGAN model with convolutional-transpose layers, having checkerboard patterns on the entire image on Figure 7(b); and a fake sample without checkerboard artifacts generated with our modified DC-GAN on Figure 7(c). This confirms that thanks to our modification of upsampling in the generator, the checkerboard pattern artifact no longer appears.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

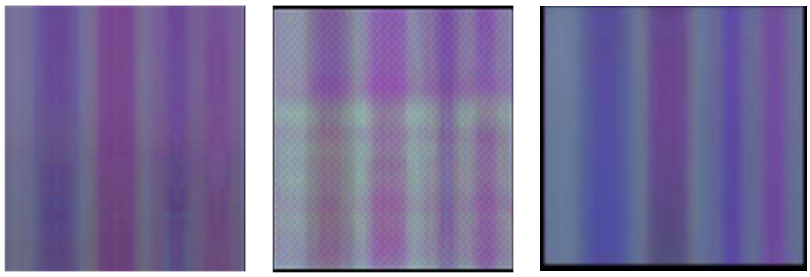

Figure 7. Example of spatio-temporal images (**a**) Ground-truth; (**b**) Image generated by original c_DC-GAN model with convolutional-transpose layers, with checkerboard artifacts; and (**c**) Image generated by our model, without checkerboard artifacts.

In order to check how essential it is for our approach to avoid checkerboard artifacts, we finally compare the corresponding generated actions. Figure 8 shows the action "drinking" generated by: **(a)** new model trained 69 epochs; and **(b)** original models after 349 epochs, when the learning curves of two models reach the same values. Comparing those two generated sequences of action "drinking", the former is obviously more continuous and realistic. The human skeleton is closer to reality as well. The body shape deformations and action unsmoothness visible on Figure 8(b) are the consequences of the checkerdboard artifacts in generated pseudo-image, which translate into very bad pose sequence after reconverting from TSSI pseudo-image.

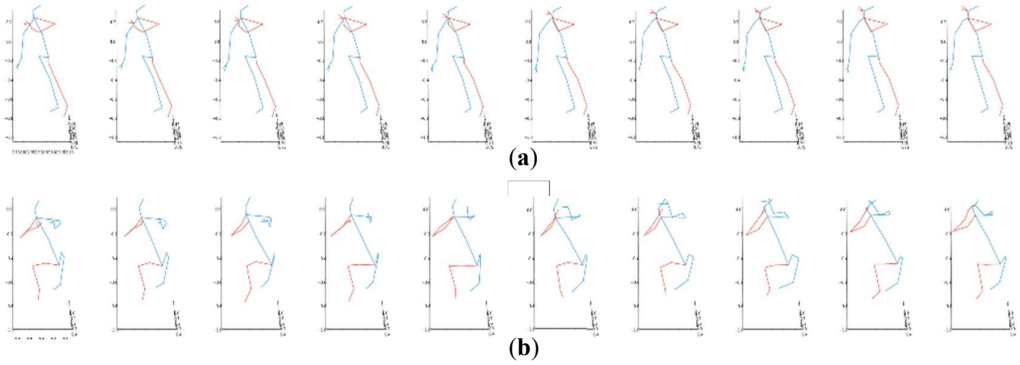

Figure 8. Sequence of action "drinking" generated by **(a)** our model (using upsample + convolutional layers) trained after 69 epochs and **(b)** original model (using convolutional-transpose layer) trained after 349 epochs.

In conclusion, our analysis confirm that for our work, using upsample + convolutional layer structure to realize deconvolution in Generator is efficient and essential for obtaining realistic generated sequences of skeletons.

4.4. Importance of re-ordering of joints in TSSI spatio-temporal images

In order to assess the importance of joints reordering in TSSI skeleton representation, we also train our generative model using pseudo-images *without* tree traversal order representation. The spatial dimension is then simply the joints order, from 1 to 25. Figure 9 compares actions generated by the two trained models, without or with TSSI representation. Figure 9(a) is the training result by non-TSSI images and Figure 9(b) is the result trained by TSSI. It is obvious that after the same training epochs, the model obtained using non-TSSI pseudo-images cannot output a realistic human skeleton.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

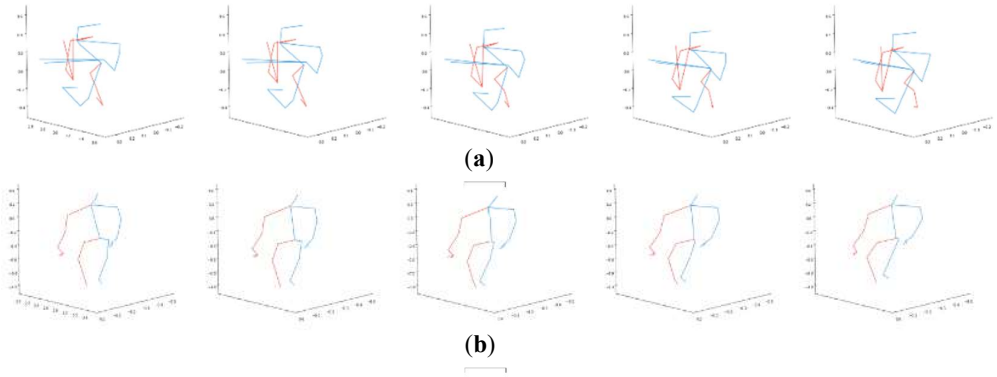

Figure 9. Sequences of action "standing" generated by the models with **(a)** non-TSSI training images **(b)** TSSI training images.

4.5. Hyper-parameters tuning

After justifying the choice of our model architecture and data format, we explore the effect of hyper-parameters values: learning rate, batch size and number of dimensions of latent space z. By default, we set learning rate at 0.0002, batch size at 64 and dimension of z at 100. Figure 10 shows the learning curves for the model trained with different learning rates: 0.001, 0.0005, 0.0002 and 0.0001. Opposite with common sense, when the learning rate is smaller, the *Loss_D* and *Loss_G* converge faster. For models lr=0.001 and lr=0.0005, the average FID for single class reaches the minimum at 150th epoch and then goes up at 200th epoch. For model lr=0.0002, FID scores increase slightly at 200th epoch. For model lr=0.0001, FID scores keep decreasing but converge slowly. For all the four models, FID scores are close to each other at 200th epoch.

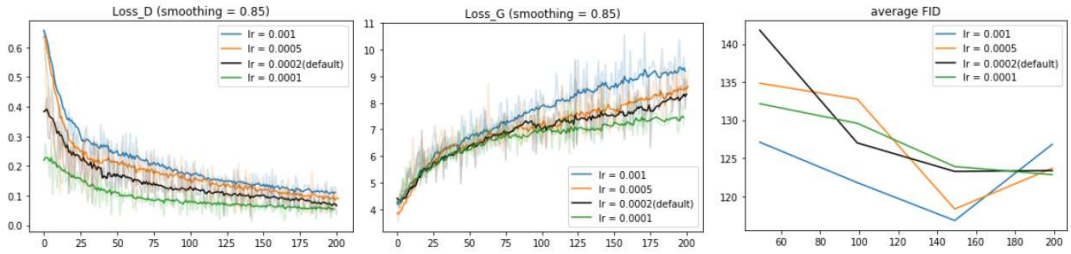

Figure 10. Learning curves for the model trained with different learning rates: 0.001, 0.0005, 0.0002 and 0.0001.

We also train the model with different batch sizes (of training data): 16, 32, 64 and 128 (see Figure 11). When the batch size is small, the learning curves *Loss_D* and *Loss_G* are smoother and Discriminator learns easier. However, FID score converges clearly best with batch size = 64.

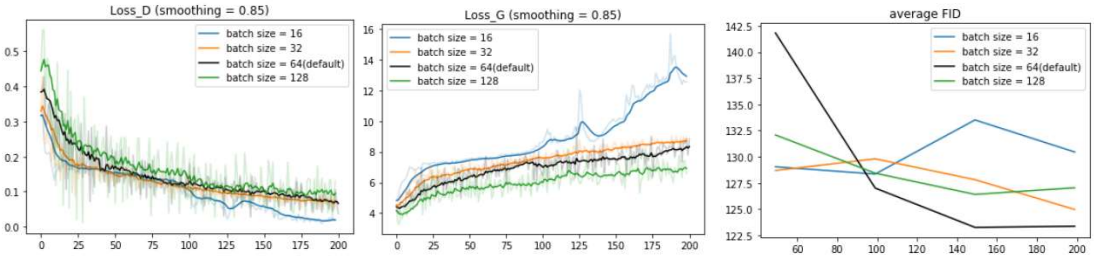

Figure 11. Learning curves for the model trained under three different batch size (of input data): 16, 32, 64 and 128

Finally, we investigate the influence of the dimension dim_z of the latent space z. In Figure 12, *Loss_D* and *Loss_G* are not affected by the choice of dim_z, while FID scores vary significantly. With the increase of dim_z, FID score tend to converge faster. For model with dim_z=5, FID score shows

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

the trend of decreasing along 200 epochs. For the model with dim_z = 10 and 30, FID scores both reach the minimal value near 150th epoch and then go up. The model with dim_z = 100 gets the lowest point of FID at 130th and 180th epoch, much smaller than the results of the others. However, the further training epochs are needed to figure out whether FID for dim_z =5 and 100 would continue decreasing.

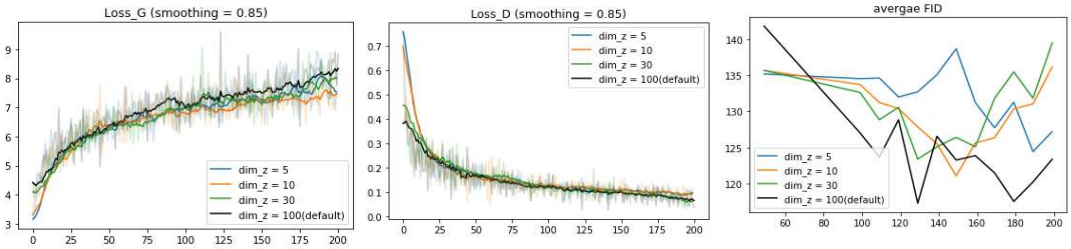

Figure 12. Learning curves for the model trained in different number of dimensions of the latent space z (input of G): 5, 30 and 100. For the average FID, we plot the value at 50, 100, 110, 120…200 epoch.

To sum up, when training the model 200 epochs, the original choice of learning rate and batch size is already a relatively good setup. Regarding the optimal dimension of latent space, further study is needed to determine it robustly.

4.6. Analysis of latent space

We now analyze the influence of the latent variable z on the style of generated pose sequences. To this end, we move separately along each dimension of the latent space, and try to understand the specific influence of each z coordinate on the appeareance of generated sequences. In Figure 13, we visualize an evolution of action "standing up". The model is trained after 200 epochs with dim_z = 5. For each figure, we adopt the linear interpolation on one dimension of z, with 5 points in the range of [-1,1]. Each row is the sequence generated by an interpolation "point" of latent space z. By comparing the first frame of each sequence (the first column), it seems that latent space z influence in particular the initial orientation of the skeleton. For instance, in Figure 13, the body rotates from front to its right side (or the left side from reader's point of view). The evolution from top to bottom is continuous and smooth. At the same time, for each row of sequence, the action "standing up" is well performed. This example shows that by tuning on latent space z, our generative model is able to continuously control properties, such as the orientation, of the action and eventually to define the style of action while generating the correct class of action indicated by conditional label.

5. Conclusions

In this work, we propose a new class-conditioned generative model for human skeletal motions. Our generative model is based on TSSI (Tree Structure Skeleton Image) spatio-temporal pseudo-image representation of skeletal pose sequences, on which is applied a conditional DC-GAN to generate artificial pseudo-images that can be decoded into realistic artificial human skeletal movements. In our set-up, each column of a pseudo-image corresponds to approximately one single skeletal joint, so it is quite essential to avoid artifacts in generated pseudo-image; therefore, we modified the original deconvolution operation in standard DC-GAN in order to efficiently eliminate checkerboard artifacts.

We have evaluated our approach on the large NTU_RGB-D human actions public dataset, comprising 60 classes of actions. We have shown that our generative model is able to generate artificial human movements qualitatively and visually corresponding to the correct action class used as condition for the DC-GAN. We also used Frechet Inception Distance as an evaluating metric, which quantitatively confirms the qualitative results.

The code of our work will soon be made available on Github. As further works, we will continue to improve the quality of generated human movements. For instance, add conditional label in every

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0039.v1

hidden layers in the network[27] and evaluate the performances. Also, try or develop new representation of skeletal pose sequences. Moreover, the evaluation of the model on other human skeletal datasets is important to verify the universality of the model.

Figure 13. An evolution of action "standing up" generated by different latent space z.

Appendix A

Implementation details of our conditional Deep Convolutional Generative Adversarial Networks (c_DCGAN) (with hyperparameters in default setting: dim_z = 100)

```
Generator(
  (input_layer): Conv2d(160, 8192, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (fc_main): Sequential(
    (0): UpscalingTwo(
      (upscaler): Sequential(
        (0): Upsample(scale_factor=2.0, mode=bilinear)
        (1): ReflectionPad2d((1, 1, 1, 1))
        (2): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), bias=False)
        (3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (4): ReLU(inplace=True)
      )
    )
    (1): UpscalingTwo(
      (upscaler): Sequential(
        (0): Upsample(scale_factor=2.0, mode=bilinear)
        (1): ReflectionPad2d((1, 1, 1, 1))
        (2): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), bias=False)
        (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (4): ReLU(inplace=True)
      )
```

 doi:10.20944/preprints202011.0039.v1


```
      )
      (2): UpscalingTwo(
        (upscaler): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): ReflectionPad2d((1, 1, 1, 1))
          (2): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), bias=False)
          (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (4): ReLU(inplace=True)
        )
      )
      (3): UpscalingTwo(
        (upscaler): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): ReflectionPad2d((1, 1, 1, 1))
          (2): Conv2d(64, 3, kernel_size=(3, 3), stride=(1, 1), bias=False)
          (3): Tanh()
        )
      )
    )
  )
  Discriminator(
    (hidden_layer1): Sequential(
      (0): Conv2d(63, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (1): LeakyReLU(negative_slope=0.2, inplace=True)
      (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (4): LeakyReLU(negative_slope=0.2, inplace=True)
      (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): LeakyReLU(negative_slope=0.2, inplace=True)
      (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
      (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (10): LeakyReLU(negative_slope=0.2, inplace=True)
      (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
      (12): Sigmoid()
    )
  )
```

References

1. Martinez, J.; Black, M. J.; Romero, J. and Ieee. "On human motion prediction using recurrent neural networks." In *30th ieee conference on computer vision and pattern recognition*. Ieee conference on computer vision and pattern recognition. 2017, 4674-83.
2. Ren, B.; Liu, M.; Ding, R. and Liu, H. "A survey on 3d skeleton-based action recognition using learning method." (2020):
3. Radford, A.; Metz, L. and Chintala, S. "Unsupervised representation learning with deep convolutional generative adversarial networks." *CoRR* abs/1511.06434 (2016):
4. Fragkiadaki, K.; Levine, S.; Felsen, P. and Malik, J. "Recurrent network models for human dynamics." *2015 IEEE International Conference on Computer Vision (ICCV)* (2015): 4346-54.
5. Jain, A.; Zamir, A. R.; Savarese, S. and Saxena, A. "Structural-rnn: Deep learning on spatio-temporal graphs." In *2016 ieee conference on computer vision and pattern recognition*. Ieee conference on computer vision and pattern recognition. 2016, 5308-17.
6. Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A. and Bengio, Y. "A recurrent latent variable model for sequential data." In *Advances in neural information processing systems 28*. C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett. Advances in neural information processing systems. 28. 2015,
7. Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. and Bengio, Y. "Generative adversarial nets." In *Advances in neural information processing systems 27*. Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger. Advances in neural information processing systems. 27. 2014,
8. van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. and Kavukcuoglu, K. "Wavenet: A generative model for raw audio." (2016): 15 pp.-15 pp. <Go to ISI>://INSPEC:16405938.
9. Barsoum, E.; Kender, J.; Liu, Z. and Ieee. "Hp-gan: Probabilistic 3d human motion prediction via gan." In *Proceedings 2018 ieee/cvf conference on computer vision and pattern recognition workshops*. Ieee computer society conference on computer vision and pattern recognition workshops. 2018, 1499-508.
10. Ishaan, G.; Ahmed, F.; Arjovsky, M.; Dumoulin, V. and Courville, A. "Improved training of wasserstein gans." In *Advances in neural information processing systems 30*. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett. Advances in neural information processing systems. 30. 2017,
11. Kiasari, M. A.; Moirangthem, D. S. and Lee, M. "Human action generation with generative adversarial networks." abs/1805.10416 (2018):
12. Wang, P.; Li, W.; Li, C. and Hou, Y. "Action recognition based on joint trajectory maps with convolutional neural networks." *Knowledge-Based Systems* 158 (2018): 43-53. 10.1016/j.knosys.2018.05.029. <Go to ISI>://WOS:000440529200004.
13. Li, B.; Dai, Y.; Cheng, X.; Chen, H.; Lin, Y. and He, M. "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn." *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (2017): 601-04.
14. Li, Y.; Xia, R.; Liu, X. and Huang, Q. "Learning shape-motion representations from geometric algebra spatio-temporal model for skeleton-based action recognition." *2019 IEEE International Conference on Multimedia and Expo (ICME)* (2019): 1066-71.
15. Liu, M.; Liu, H. and Chen, C. "Enhanced skeleton visualization for view invariant human action recognition." *Pattern Recognition* 68 (2017): 346-62. 10.1016/j.patcog.2017.02.030. <Go to ISI>://WOS:000401381100027.

16. Caetano, C.; Sena, J.; Bremond, F.; Dos Santos, J. A. and Schwartz, W. R. *Skelemotion: A new representation of skeleton joint sequences based on motion information for 3d action recognition*. 2019,
17. Caetano, C.; Bremond, F.; Schwartz, W. R. and Ieee. "Skeleton image representation for 3d action recognition based on tree structure and reference joints." In *2019 32nd sibgrapi conference on graphics, patterns and images*. Sibgrapi - brazilian symposium on computer graphics and image processing. 2019, 16-23.
18. Li, C.; Zhong, Q.; Xie, D. and Pu, S. "Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation." Presented at IJCAI, 2018.
19. Shahroudy, A.; Liu, J.; Ng, T.-T.; Wang, G. and Ieee. "Ntu rgb plus d: A large scale dataset for 3d human activity analysis." In *2016 ieee conference on computer vision and pattern recognition*. Ieee conference on computer vision and pattern recognition. 2016, 1010-19.
20. Yang, Z.; Li, Y.; Yang, J. and Luo, J. "Action recognition with spatio-temporal visual attention on skeleton image sequences." *Ieee Transactions on Circuits and Systems for Video Technology* 29 (2019): 2405-15. 10.1109/tcsvt.2018.2864148. <Go to ISI>://WOS:000480310500016.
21. Mirza, M. and Osindero, S. "Conditional generative adversarial nets." *arXiv* abs/1411.1784 (2014):
22. Sugawara, Y.; Shiota, S. and Kiya, H. "Checkerboard artifacts free convolutional neural networks." *APSIPA Transactions on Signal and Information Processing* 8 (2019): 10.1017/atsip.2019.2.
23. Dowson, D. and Landau, B. "The fréchet distance between multivariate normal distributions." *Journal of Multivariate Analysis* 12 (1982): 450-55.
24. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B. and Hochreiter, S. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." Presented at NIPS, 2017.
25. Salimans, T.; Goodfellow, I. J.; Zaremba, W.; Cheung, V.; Radford, A. and Chen, X. "Improved techniques for training gans." *arXiv* abs/1606.03498 (2016):
26. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J. and Wojna, Z. "Rethinking the inception architecture for computer vision." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016): 2818-26.
27. Giuffrida, M.; Scharr, H. and Tsaftaris, S. "Arigan: Synthetic arabidopsis plants using generative adversarial network." *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (2017): 2064-71.