# Human Motion Generative Model using Wasserstein GAN

AYUMI SHIOBARA[1] MAKOTO MURAKAMI[1]

***Abstract***: Human motion control, edit, and synthesis are important tasks to create 3D computer graphics video games or movies, because some characters act like humans in most of them. Our aim is to construct a system which can generate various natural character motions. We assume that the process of human motion generation is complex and nonlinear, and it can be modeled by deep neural networks. However, this process cannot be observed, and it needs to be estimated by learning from observable human motion data. On the other hand, the process of discrimination which is opposite to the generation is also modeled by deep neural networks. And the generator and discriminator are trained using human motion data. In this paper we constructed a human motion generative model using Wasserstein GAN. As a result, our model can generate various human motions from a 512-dimensional latent space.

***Keywords***: Deep Neural Networks, Generative Model, Human Motion, Generative Adversarial Networks.

## 1. Introduction

Human motion control, edit, and synthesis are important tasks to create 3D computer graphics video games or movies, because some characters act like humans in most of them. Key frame interpolation method is useful for producing human motion. But the user has to set a lot of parameters of human joints in some key frames manually, and the produced motion is less realistic than the data captured with motion capture system. Therefore motion capture data-driven method is used for motion control, edit, and synthesis, and many techniques have been proposed [1].

Deep neural networks have been used in human motion control. Holden et al. [2] used convolutional autoencoders to learn human motion manifold from a large motion dataset captured with an optical motion capture system. And they stack deep neural networks on top of the autoencoders, which can map from high level parameters to the motion manifold. The proposed system can synthesize character motion from given trajectories over the floor that the character should follow, and can edit motion by optimizing in the motion manifold with some constraints.

On the other hand, some generative models using deep neural networks have been proposed. Kingma et al. [3], [4] proposed variational autoencoder, and applied it for image generation. The proposed system can generate natural and various images. Sabaththe et al. [5] used LSTM-based variational autoencoder for automatic music composition and it can generate various music pieces that represent some musical characteristics and properties. Radford et al. [6] proposed deep generative adversarial networks, which can also generate natural and various images.

Our aim is to construct a human motion generative model using generative adversarial networks which can generate more natural data than variational autoencoder.

## 2. Generative Adversarial Networks (GAN)

GAN have two different kinds of networks which are a generator and a discriminator as shown in Fig. 1. The parameters of each network are optimized by training using dataset. In generation of images, the generator samples latent variables $z$ from a uniform distribution and generates images closer to training images from $\boldsymbol{z}$. On the other hand the discriminator tries to identify images coming from the generator as fake or taken from the actual training dataset. Therefore the discriminator is trained to maximize the probability of identifying generated images and training images correctly. And the generator is trained to minimize the probability that the discriminator identifies the generated images. The loss function is represented as

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

where $D(\boldsymbol{x})$ is the probability that the discriminator identifies $\boldsymbol{x}$ as a training data, and $G(\boldsymbol{z})$ is generated data from latent variable $\boldsymbol{z}$. When the discriminator is trained well, $D(\boldsymbol{x})$ increases and $D(G(\boldsymbol{z}))$ decreases. On the other hand when the generator outputs data close to training data, $D(G(\boldsymbol{z}))$ increases.

In this way the discriminator and the generator are trained adversarially by competing with each other. When the generator outputs data closer to training data enough not to identify, the probability that the discriminator identifies the data equals to 50%. The trained generator is used to create realistic data.

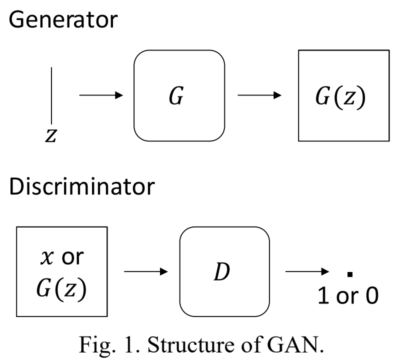

Fig. 1. Structure of GAN.

## 3. Motion Data

In this section we describe human motion dataset and

1 Toyo University, Kujirai, Kawagoe, Saitama 350-0815, Japan

preprocessing of it for learning motion generative model.

We use the CMU Graphics Lab Motion Capture Database [7], which consists of 2,505 recordings of human motion captured with an optical motion capture system. We downsample this original data with 120 fps to 30 fps.

The original motion data is represented as 3-DOF local rotations of 19 joints and 3-DOF global translation of root joint (hip) as shown in Fig. 2. We convert them into the global positions $\boldsymbol{p}_j^{(g)}(t)$, where $j$ is a joint and $t$ is time.

Given the global position of hip $\boldsymbol{p}_h^{(g)}(t)$, left shoulder $\boldsymbol{p}_{ls}^{(g)}(t)$, and right shoulder $\boldsymbol{p}_{rs}^{(g)}(t)$ in time $t$, the forward vector of body $v_f(t)$ is calculated as

$$\begin{aligned} \boldsymbol{v}_l(t) &= \boldsymbol{p}_{ls}^{(g)}(t) - \boldsymbol{p}_h^{(g)}(t) \\ \boldsymbol{v}_r(t) &= \boldsymbol{p}_{rs}^{(g)}(t) - \boldsymbol{p}_h^{(g)}(t) \\ \boldsymbol{v}_f(t) &= \boldsymbol{v}_l(t) \times \boldsymbol{v}_r(t) \end{aligned} \tag{2}$$

The basis vectors of local coordinate system at time $t$ are calculated as

$$\begin{aligned} \boldsymbol{e}_y &= [0 \quad 1 \quad 0]^T \\ \boldsymbol{e}_x(t) &= \frac{\boldsymbol{v}_f(t) \times \boldsymbol{e}_y}{\|\boldsymbol{v}_f(t) \times \boldsymbol{e}_y\|} \\ \boldsymbol{e}_z(t) &= \boldsymbol{e}_y \times \boldsymbol{e}_x(t) \end{aligned} \tag{3}$$

The rotation matrix at time $t$ is represented as

$$R_{gl}(t) = \begin{bmatrix} \boldsymbol{e}_x(t) & \boldsymbol{e}_y & \boldsymbol{e}_z(t) & \boldsymbol{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Let $p_{min,y}^{(g)}(0)$ be the minimum of $y$-coordinate over all joints in the initial frame, which indicates the floor height if one of joints is on the ground in the initial frame. We set the origin in local coordinate system at time $t$ as a point on the ground where the root joint position is projected onto. The translation matrix at time $t$ is represented as

$$T_{gl}(t) = \begin{bmatrix} 1 & 0 & 0 & p_{h,x}^{(g)}(t) \\ 0 & 1 & 0 & p_{\min,y}^{(g)}(0) \\ 0 & 0 & 0 & p_{h,z}^{(g)}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where $p_{h,x}^{(g)}(t)$ and $p_{h,z}^{(g)}(t)$ are $x$-coordinate and $z$-coordinate of the root joint at time $t$ respectively. The transformation matrices between local coordinate system and global coordinate system are represented as

$$\begin{aligned} M_{gl}(t) &= T_{gl}(t) R_{gl}(t) \\ M_{lg}(t) &= R_{gl}^T(t) T_{gl}^{-1}(t) \end{aligned} \tag{6}$$

and the local positions of joints at time $t$ is calculated as

$$\boldsymbol{p}_j^{(l)}(t) = M_{lg}(t) \boldsymbol{p}_j^{(g)}(t) \tag{7}$$

We use $y$-coordinate of the root joint, and $xyz$-coordinates of the other 18 joints. We also use velocity in the $xz$ plane and angular velocity around the $y$ axis, which are calculated from the matrix $\Delta M(t)$, which is

$$\Delta M(t) = M_{lg}(t-1) M_{gl}(t) \tag{8}$$

The motion data at each frame is a 58-dimensional vector.

We separate each sequence of frames into windows of 120 frames (about 4 seconds), overlapped by 60 frames. Finally we get 14,122 motions, and we subtract the mean from them and divide them by the standard deviation to standardize the data.

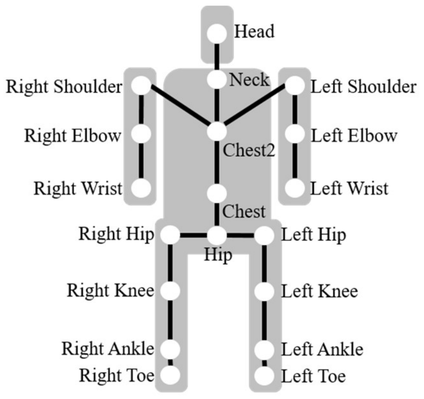

Fig. 2. Joint structure in motion data.

## 4. Human Motion Generative Model

### 4.1 Model Structure

Fig. 3. shows the structure of the GAN discriminator constructed in this study. As described in Chapter 3, the motion data used in this research is a $58 \times 120$ dimensional vector. In the first layer of the discriminator, 64 types of filters with sizes of 58 in the spatial direction and 15 in the temporal direction are applied. The stride in the spatial direction of the filter is 58, therefore all 58 dimensional joint positions are convolved. The stride in the time direction of the filter is 2, therefore the extracted features are half size of the input motion data in the temporal direction. In the second layer of the discriminator, 128 filters with sizes of 15 in the temporal direction are used, and the stride of the filters is 4. In the first and second layers, 25% of connections are dropped out, and LeakyReLU with a negative slope of 0.2 is used as the activation function. And the last layer, the extracted features are flattened and pass through a fully connected layer to get a scaler.

The structure of generator is shown in Fig.4. In this research, dimension of the latent space is 512, and latent variables are sampled from the 512 dimensional uniform distribution. The generator outputs motions from the latent variables through the network which performs the inverse transform with the discriminator. The first layer in the generator is a fully connected layer which reconstructs $15 \times 1 \times 128$ dimensional features from 512 dimensional latent variable. In the second layer, 64 filters with sizes of 15 are used, and upsampling is performed in the temporal direction, to get $60 \times 1 \times 64$ dimensional features. And in this layer batch normalization is applied and LeakyReLU with a negative slope of 0.2 is used as the activation function. In the third layer, 58 filters of with sizes of 15 are used, and upsampling is performed in the temporal direction, to reconstruct a motion which represents $58 \times 120$ dimensional vector. In this layer tanh is used as the activation function.

### 4.2 Loss Function

GAN proposed by Goodfellow et al. [8] uses Jensen-Shannon divergence between the real distribution and the generated distribution to optimize network parameters. Jensen-Shannon divergence leads us to mode collapse: during the training the generator always generate same outputs. It causes that the generator can trick a particular discriminator and learn generated distribution only fit to a small set of real distribution.

Arjovsky et al. [9] proposed Wasserstein GAN which uses Wasserstein divergence instead of Jensen-Shannon divergence to optimize network parameters. The Wasserstein divergence between real distribution $P_r$ and generated distribution $P_g$ is defined as

$$W(P_r, P_g) = \max_{w \in W} \mathbb{E}_{x \sim P_r}[D_w(x)] - \mathbb{E}_{z \sim p(z)}[D_w(G_\theta(z))] \quad (9)$$

The discriminative function is restricted to 1-Lipschitz and Arjovsky et al. used weight clipping to enforce this constraint. But it sometimes generates poor samples or fails to converge. To avoid these problems, Gullajani et al. [10] proposed gradient penalty (WGAN-GP) as an alternative way to enforce Lipschitz constraint.

In this paper, we used the loss function with gradient penalty shown in equation (10) to optimize the network parameters.

$$\begin{aligned} \mathrm{L} = {} & \mathbb{E}_{G(z) \sim p_g}[D(G(z))] - \mathbb{E}_{x \sim P_r}[D(x)] \\ & + 10\mathbb{E}_{G(z) \sim P_{G(z)}}\left[\left(\left\|\nabla_{G(z)} D(G(z))\right\|_2 - 1\right)^2\right] \quad (10) \end{aligned}$$

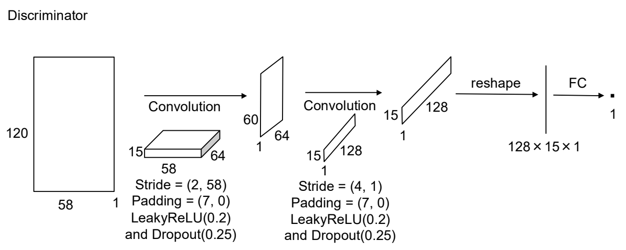

Fig. 3. Network structure of discriminator of Wasserstein GAN.

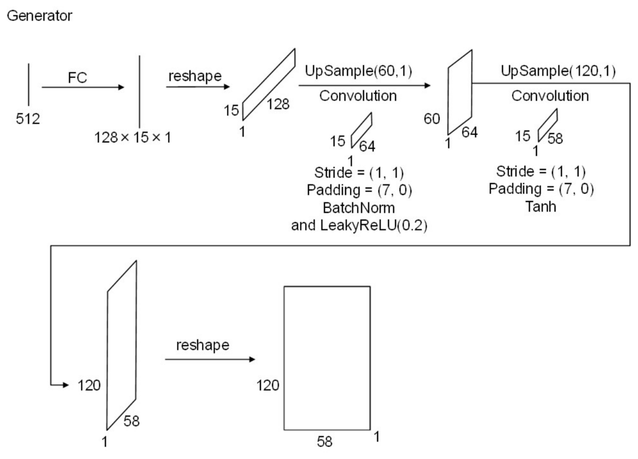

Fig. 4. Network structure of generator of Wasserstein GAN.

## 5. Experiment

### 5.1 Experiment 1

An experiment was conducted to evaluate how natural and how various motions our model can generate.

The number of training data is 12,720, and the number of validation data is 1,412. We use gradient descent with Adam optimization algorithm, batch size is 100, and the optimization is performed for 200 epochs.

Fig.5 shows losses of discriminator for training (D_loss) and validation data (D_val_loss), and losses of generator for training (G_loss) and validation data (G_val_loss) in each epoch. In the period between 25th epoch and 50th epoch, the losses of discriminator and generator are getting smaller. This shows that the network parameters of the discriminator and generator are estimated appropriately in the period.

And we confirmed whether the performance of discriminator and generator improved during training process. Fig.6 shows losses of discriminators trained in 25, 75, 125, 175, and 200 epochs against generators trained in 25, 75, 125, 175, and 200 epochs. The performance of discriminators is improved especially in the period between 25th and 50th epoch. Fig.7 shows losses of the generators against the discriminators. The performance of generators is improved slightly in the period between 25th and 50th epoch, but it is not changed in the last 125

In order to confirm whether the performance of the generators is improved during learning process, we generated motions using the generators trained in different epochs. Fig.8 shows 40 motions generated from eight randomly sampled latent variables using the five generators trained in 25, 75, 125, 175, and 200 epochs. The generators trained in 75, 125, 175, and 200 epochs can generate more various natural motion than the generator trained in 25 epochs.

Fig.9 shows 64 motions generated from 64 randomly sampled latent variables using the generator trained in 200 epochs. The proposed model can't generate natural human motions, but it can generate various motions.

### 5.2 Experiment 2

Another experiment was conducted in order to evaluate our proposed model in comparison with another generative model, which is Least Squares Generative Adversarial Networks (LSGAN) proposed by Mao[11] et al. It can penalize the fake samples away from the decision boundary to avoid the vanishing gradients problem.

In this experiment we used the least squares loss as shown in equation (11) to estimate the network parameters.

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}[(D(\boldsymbol{x})-1)^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}\left[\left(D\left(G(\boldsymbol{z})\right)\right)^2\right]$$
$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}\left[\left(D\left(G(\boldsymbol{z})\right)-1\right)^2\right] \qquad (11)$$

Fig.10 shows the structure of discriminator, which is the same as the structure of the discriminator used in experiment 1 except applying batch normalization in the second layer. And the structure of generator is exact the same as the structure of the generator shown in Fig.4.

The number of training data is 12,720, and the number of validation data is 1,412. We use gradient descent with Adam optimization algorithm, batch size is 100, and the optimization is performed for 200 epochs.

Fig.11 shows losses of discriminator for training (D_loss) and validation data (D_val_loss), and losses of generator for training (G_loss) and validation data (G_val_loss) in each epoch. In the first 25 epochs the losses of discriminator are getting smaller and the losses of generator are getting larger. This shows that the network parameters of the discriminator are estimated appropriately than the generator in the beginning of training process.

Fig.12 shows 64 motions generated from 64 randomly sampled latent variables using the generator trained using least square loss. Compared Fig.12 with Fig.9, the generator trained using Wasserstein distance can generate more various but less natural motions than the generator trained using least square loss. This shows the Wasserstein GAN can avoid mode collapse to generate various motions.

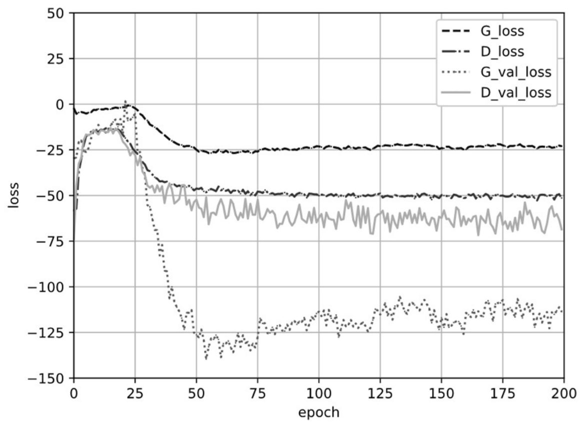

Fig. 5. Losses of discriminator for training (D_loss) and validation data (D_val_loss), and losses of generator for training (G_loss) and validation data (G_val_loss) in each epoch.

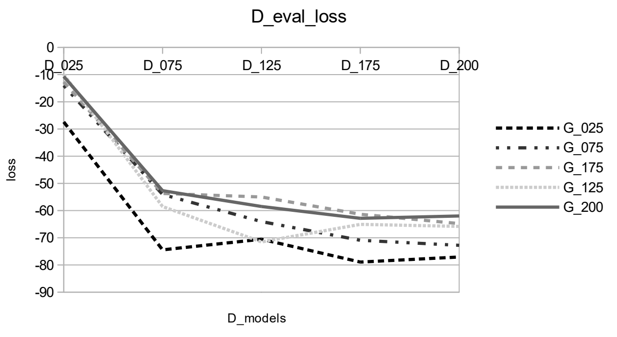

Fig. 6. Losses of discriminators trained in 25, 75, 125, 175, and 200 epochs against generators trained in 25, 75, 125, 175, and 200 epochs.

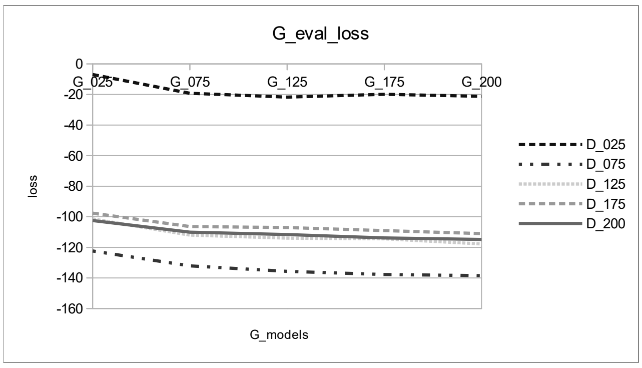

Fig. 7. Losses of generators trained in 25, 75, 125, 175, and 200 epochs against discriminators trained in 25, 75, 125, 175, and 200 epochs.

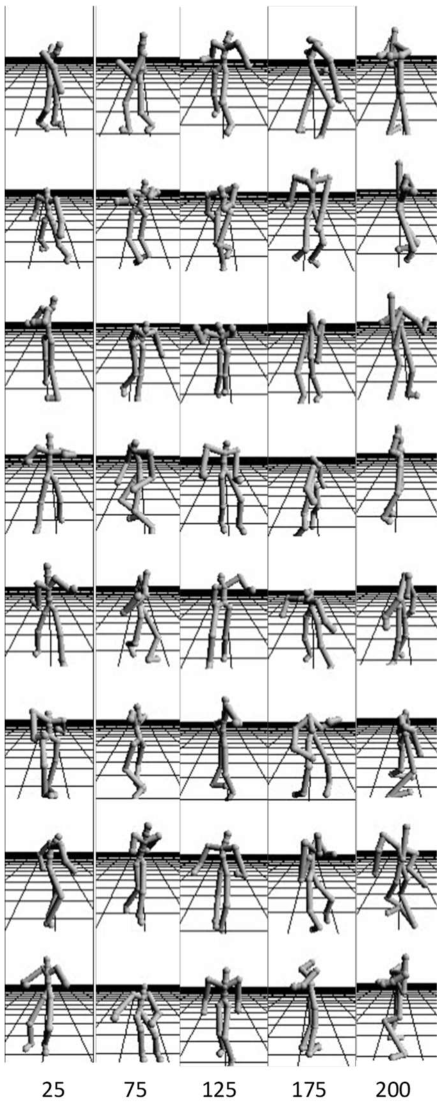

Fig. 8. 40 motions generated from eight randomly sampled latent variables using the five generators trained in 25, 75, 125, 175, and 200 epochs.

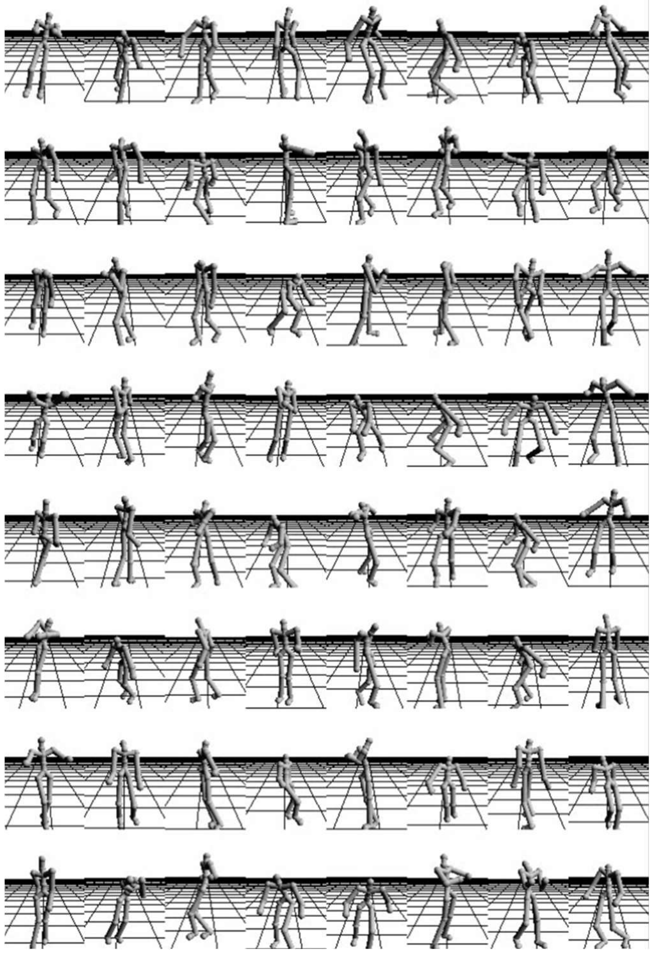

Fig. 9. 64 randomly generated motions using the generator of Wasserstein GAN.

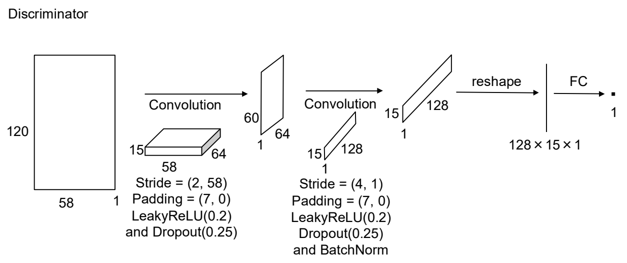

Fig. 10. Network structure of discriminator of LSGAN.

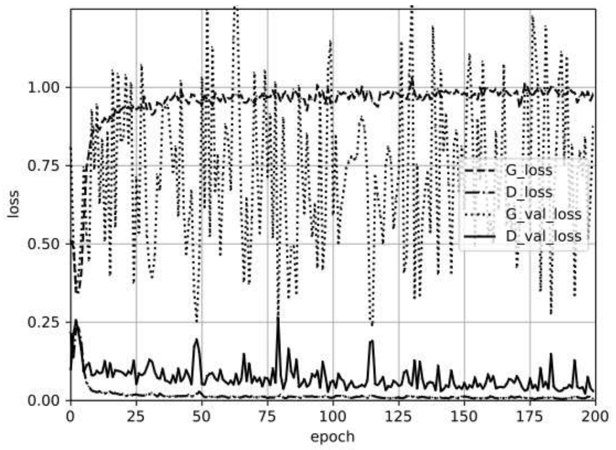

Fig. 11. Losses of discriminator for training (D_loss) and validation data (D_val_loss), and losses of generator for training (G_loss) and validation data (G_val_loss) in each epoch using LSGAN.

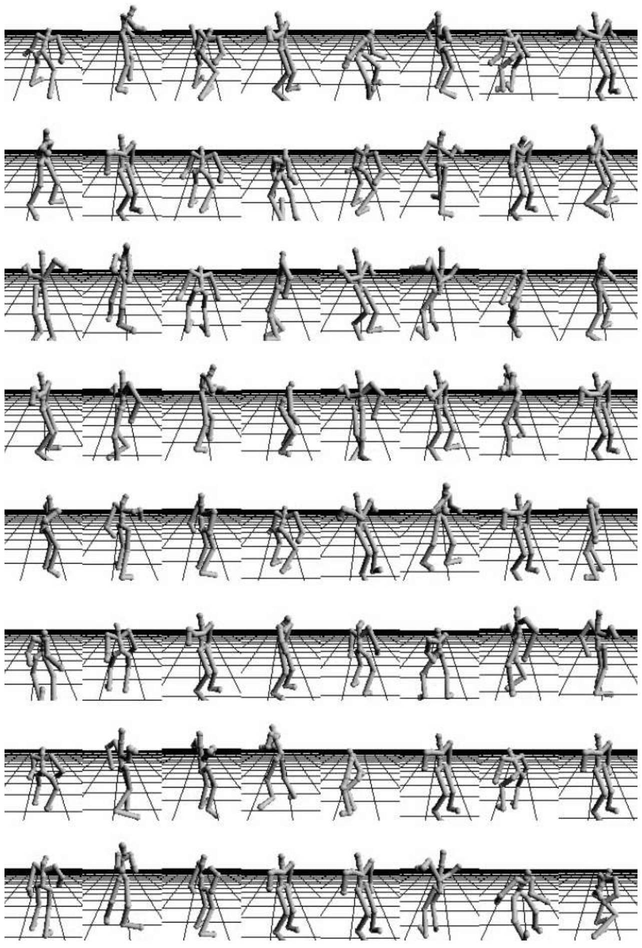

Fig. 12. 64 randomly generated motions using the generator of LSGAN.

## 6. Conclusion

We constructed a human motion generative model using Wasserstein GAN. The performance of discriminator and generator is improved in the beginning of training. And the performance of discriminator is gradually improved, but the performance of generator is not changed in the rest of the training process.

The generator of Wasserstein GAN can generate more various but less natural motions than the generator of LSGAN. This shows the Wasserstein GAN can avoid mode collapse to generate various motions.

We will improve performance of our model. Arjovsky et al. [9] proposes some techniques to enable stable learning such as not using dropout. We will use some of them and adjust some hyper parameters of netrowks.

## Reference

[1] Wang, X. Chen, Q. and Wang. W.. 3D human motion editing and synthesis : A survey. Computational and Mathematical Methods in Medicine. 2014.
[2] Holden, D. Saito, J. Komura, T. and Joyce, T.. Learning motion manifolds with convolutional autoencoders. SIGGRAPH Asia 2015 Technical Briefs. 2015.
[3] Kingma, D. P. and Welling, M.. Auto-encoding variational bayes. International Conference on Learning Representations. 2013.
[4] Kingma, D. P. Rezende, D. J. Mohamed, S. and Welling, M.. Semi-supervised learning with deep generative models. Neural Information Processing Systems. 2014.
[5] Sabaththe, R. Coutinho, E. and Schuller, B.. Deep recurrent music writer: Memory-enhanced variational autoencoder-based musical score composition and an objective measure. International Joint Conference on Neural Networks. 2017, pp. 3467-3474.
[6] Radford, A. Metz, L. and Chintala, S.. Unsupervised representation learning with deep convolutional generative adversarial networks. International Conference on Learning Representations. 2016.
[7] CMU. Carnegie Mellon University - CMU Graphics Lab - motion. http:// mocap.cs.cmu.edu.
[8] Goodfellow, I. J. Pouget-Abadie, J. Mirza, M. Xu, B. Warde-Farley, D. Ozair, S. Courville, A. C. and Bengio, Y.. Generative adversarial nets. NIPS. 2014.
[9] Arjovsky, M. Chintala, S. and Bottou, L.. Wasserstein GAN. ICML. 2017, pp. 214–223.
[10] Gulrajani, I. Ahmed, F. Arjovsky, M. Dumoulin, V. and Courville, A.. Improved training of wasserstein gans. NIPS. 2017, pp. 5767–5777.
[11] Mao, X. Li, Q. Xie, H. Lau, R. Y. K. Wang, Z. and Smolley, S. P.. Least Squares Generative Adversarial Networks. International Conference on Computer Vision (ICCV). 2017, pp. 2796- 2802.