Visual Storytelling with Stick Figures

Ziyi Gong, Xingchen Zhao, Haoyue Cui, Sijia Rong

University of Pittsburgh

CS1699 – Deep Learning

April 23rd, 2020

Code Repo: https://github.com/xingchenzhao/deep-learning-team-project

Review

We proposed a model to randomly generate a sequence of two or more stick figures that tell a story through their interactions.
The model consists of two steps:

1. Randomly generate the first stick figure sequence. Thus, the major task for the first sub-model is to learn the relations among the joints
2. Map the first stick figure to the second stick figure. The second sub-model therefore needs to learn interactions between two individuals

Step 1

$$\begin{tikzcd} x_1 \arrow[r, "\text{Encoder}"] & z \arrow[r, "\text{Decoder}"] & \hat{x}_1 \arrow[dr, dash] & \\ & & & \boldsymbol{D}_1 \\ & & x_1 \arrow[ur, dash] & \end{tikzcd}$$

$$\begin{tikzcd} x_1 \arrow[r] & \text{UNet} \arrow[r] & \hat{x}_2 \arrow[dr, dash] & \\ & & & \text{L}_2\ \text{Loss} \\ & & x_2 \arrow[ur, dash] & \end{tikzcd}$$

Step 2

$$\begin{tikzcd} z \arrow[r, "\text{Decoder}"] & \hat{x}_1 \arrow[r] & \text{UNet} \arrow[r] & \hat{x}_2 \arrow[r] & (\hat{x}_1, \hat{x}_2) \arrow[dr, dash] & \\ & & & & & \boldsymbol{D}_2 \\ & & & & (x_1, x_2) \arrow[ur, dash] & \end{tikzcd}$$

Review

In Step 1, we proposed to use a VAE-GAN, as we thought the reconstruction loss would be an assurance for the generation. On the other hand, the variational learning of the continuous latent variable distribution p(z) should enable us to get less "null" samples than vanilla autoencoder or GAN.

In Step 2, we proposed to train a UNet, because its skip connection protects the information in the first stick figure to some degree.

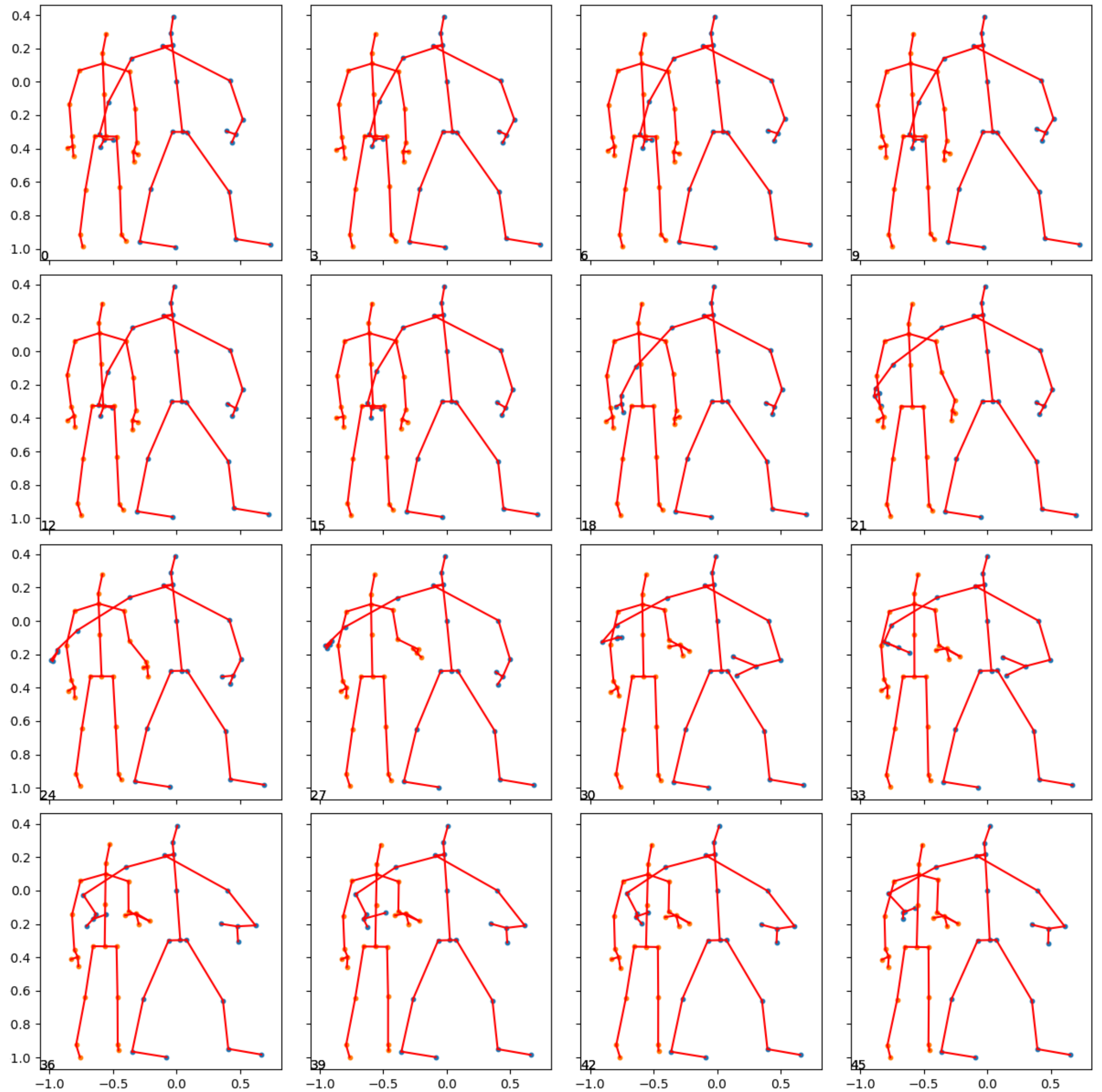
$$\begin{tikzcd}
 & x_1 \arrow[r, "\text{Encoder}"] & z \arrow[r, "\text{Decoder}"] & \hat{x}_1 \arrow[dr] & \\
 & & & x_1 \arrow[r] & \boldsymbol{D}_1 \\
\text{Step 1} & & & & \\
 & x_1 \arrow[r] & \text{UNet} \arrow[r] & \hat{x}_2 \arrow[dr] & \\
 & & & x_2 \arrow[r] & \text{L}_2\text{ Loss}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Step 2} & z \arrow[r, "\text{Decoder}"] & \hat{x}_1 \arrow[r] & \text{UNet} \arrow[r] & \hat{x}_2 \arrow[r] & (\hat{x}_1, \hat{x}_2) \arrow[dr] & \\
 & & & & & (x_1, x_2) \arrow[r] & \boldsymbol{D}_2
\end{tikzcd}$$

Data Preparation

We chose "NTU RGB+D" Dataset and "NTU RGB+D 120" Dataset which provide 26 categories of full-body, two-person interactions (see figure). We intended to perform curriculum learning so that 4 categories were selected for initial training.

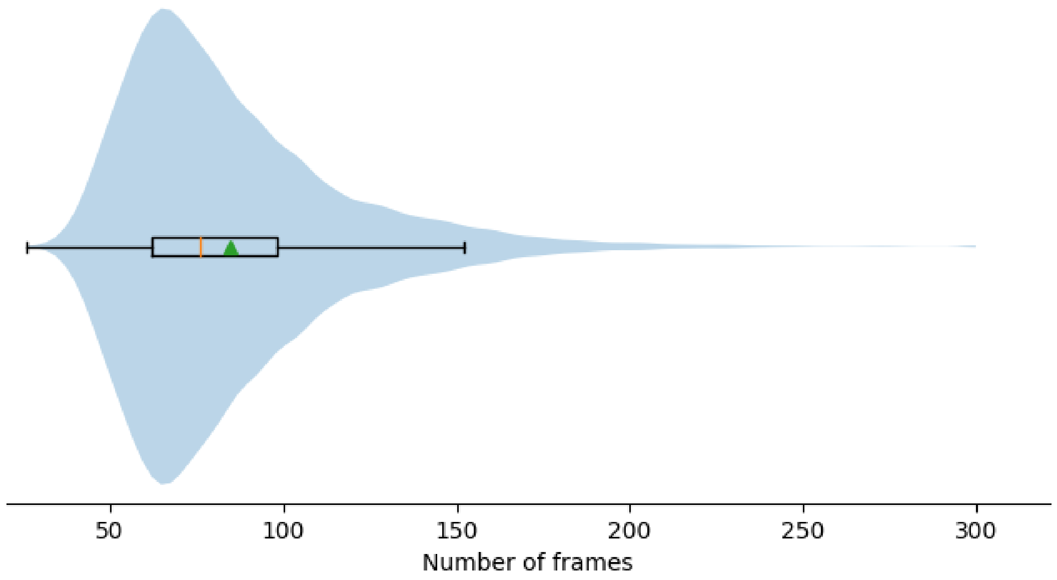
The two stick figures in a sequence were transformed so that one's bounding box is (-1, 1) for both x and y dimensions, and the other's scales and positions are relative to the first skeleton. Consequently, the data size was doubled, and the scale effect was removed.



Data Preparation

We also found the distribution of the number of frames approximates a relatively diffusive, positive skewed normal distribution (see figure). In case of aliasing, we, though not thoroughly considering about Nyquist-Shannon sampling theorem, picked 100 as the target number of frame such that less needed to be downsampled. The outliers were removed, and samples with lower numbers of frames were linearly interpolated.

The final dimension of a sample is
(number of stick figures = 2, number of frames = 100, number of joints = 25 * 2 = 50)



Development

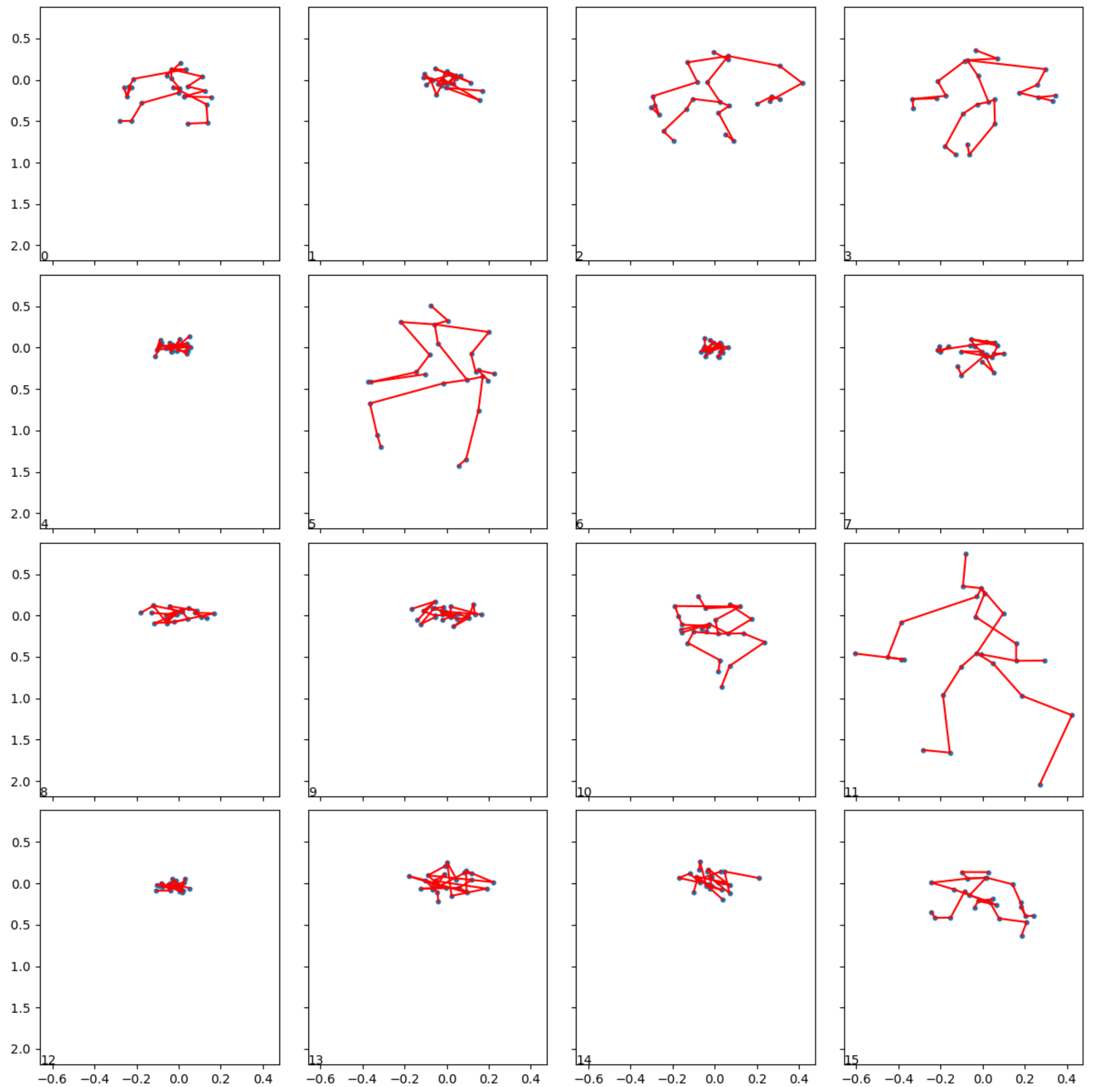
We started with a warm-up by fitting a vanilla GAN, however failing to achieve acceptable results. In these early failures, we identified the vanishing gradient problem in the GANs proposed. The generator, in many cases, did not evolve much as the discriminator inaccurately assigned the generated samples with high confidence scores (close to 1), probably because the skeletons were largely variable.

To avoid the problem in future development, we instead used Wasserstein GAN, where the discriminator performs regression instead of classification. WGAN can better approximate the distribution of data observed in the training dataset.

Development

The general structure of the decoder of the VAE was three Conv-BN-leakyReLU blocks, and the encoder was a mirror of that structure, followed by two parallel dropout-linear blocks that predicts the means and the log-variances.

A broad search over the hyperparameters of VAE-WGAN was performed. However, the model still failed to converge well and generated unrecognizable sequences (see figure, the bottom left digits represent frame numbers).



Development

We reasoned the task for a single convolutional VAE structure to generate a whole time series was too challenging. Hence, we separated Step 1 into two sub-tasks (two GANs): generate (GAN_seq) a sequence of latent variables and generate (GAN_0) stick figures based on the latent variables yielded*.

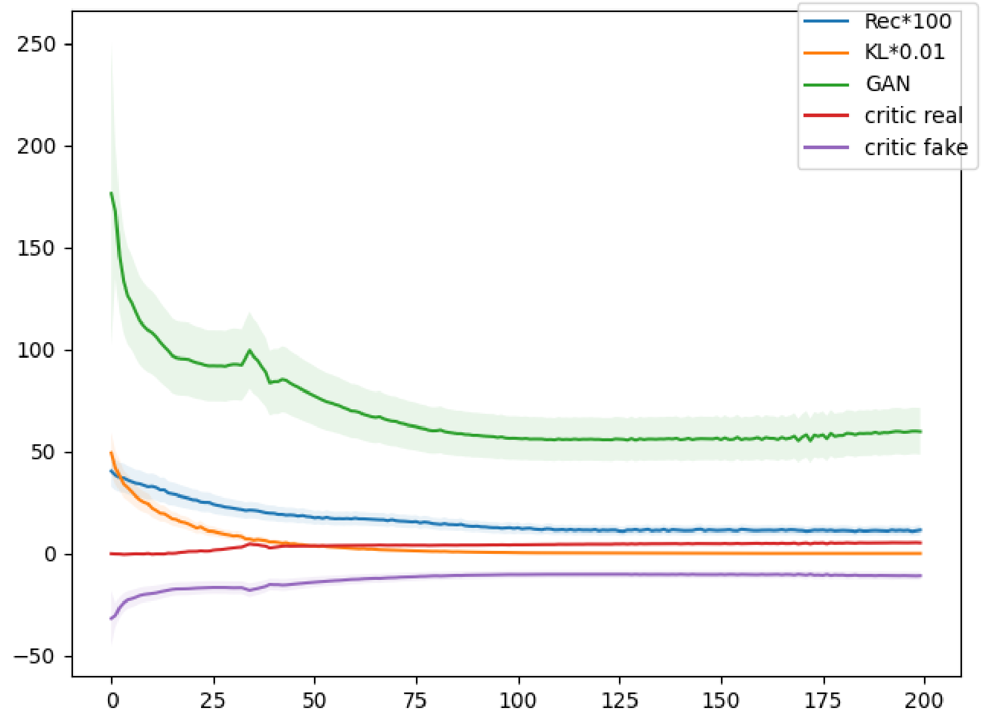
In addition, a typical WGAN was also in development in parallel, in case it may outperform VAE-WGAN.

*Idea adopted from Haoye Cai *et al.* (2018). Deep Video Generation, Prediction and Completion of Human Action Sequences. ECCV, 374–390. DOI: 10.1007/978-3-030-01216-8_23

Development

The structure of the decoder of the new VAE was two linear-BN-leakyReLU blocks, and the encoder's was also a mirror of that structure, followed by two parallel dropout-linear blocks that predicts the means and the log-variances.

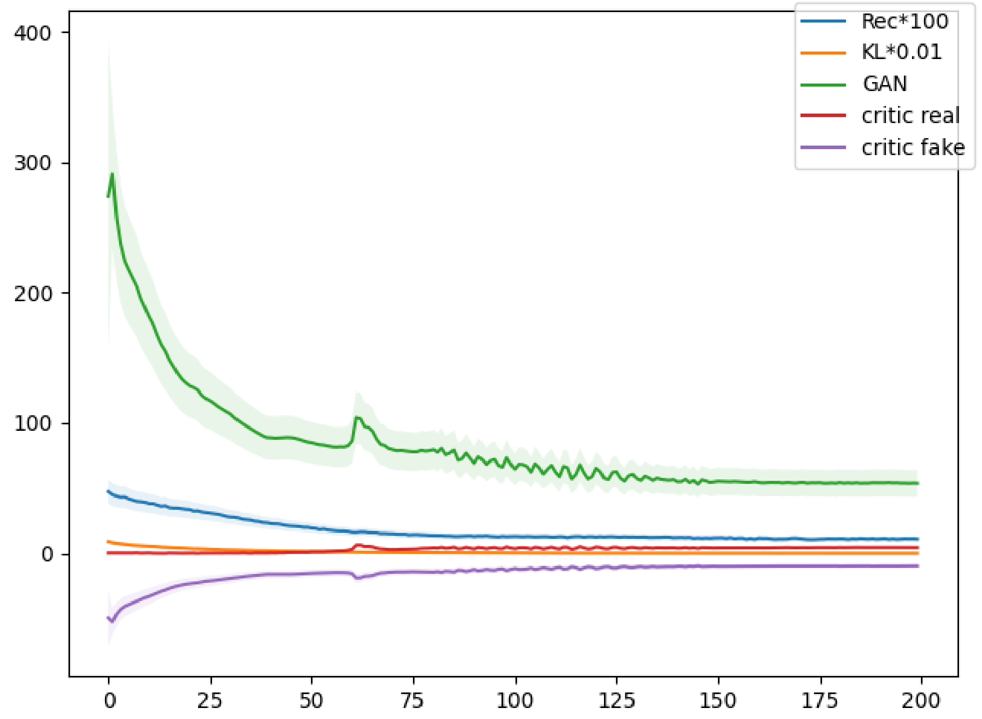
Again, a broad search over the hyperparameters of VAE-WGAN was performed. It was found that the KL loss of VAE should be weighted by a small coefficient, so it does not dominate the process. Two examples are shown:



Left: hidden layers dimension = 50 and 30, latent space dimension = 20.

Right: hidden layers dimensio n = 25 and 12, latent space dimension = 3.

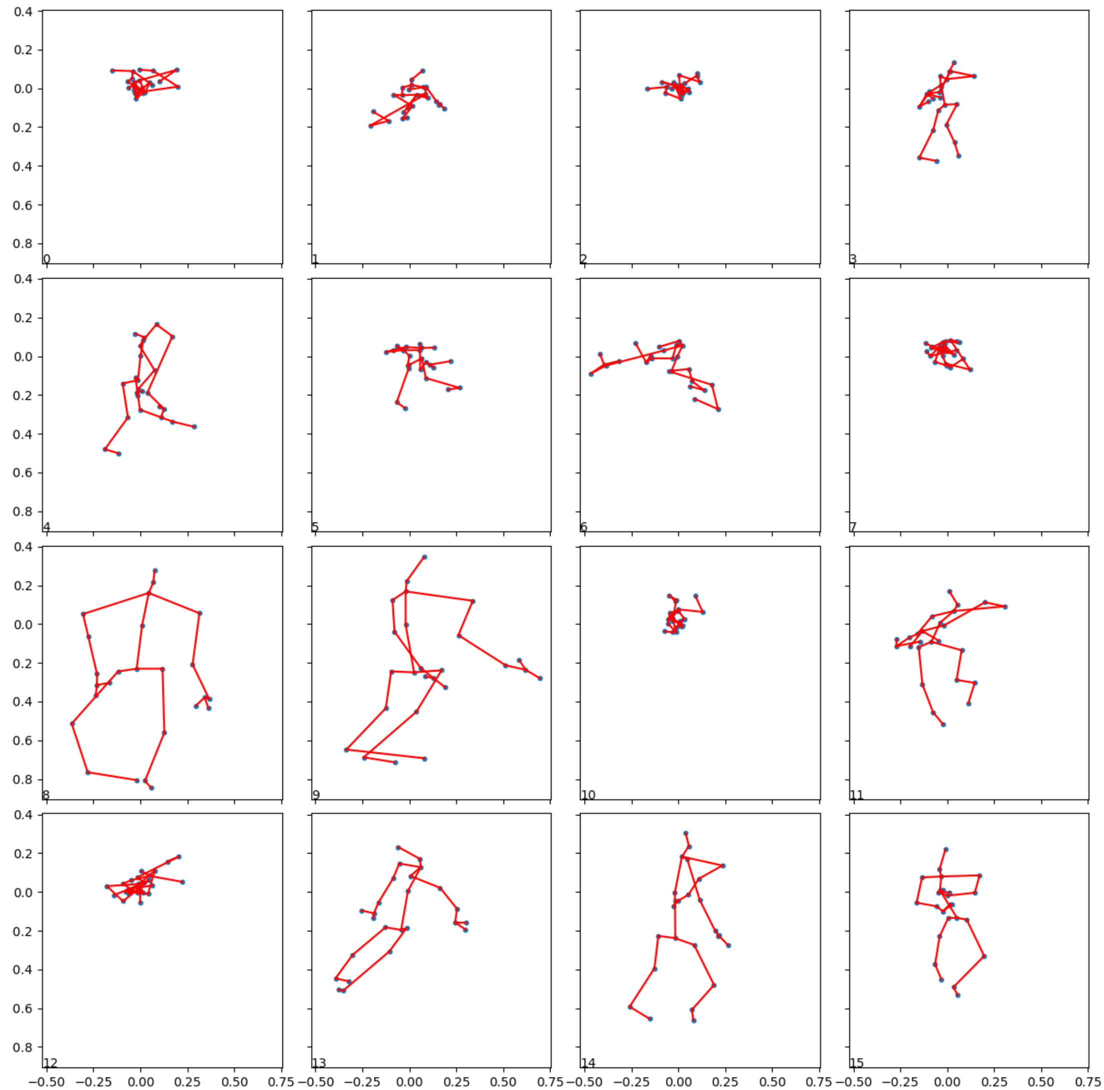
Shade areas stand for one stdev. of batch losses, and the curves stand for the mean trends. "rec" stands for reconstruction loss and "GAN" stands for the generator loss of two samples reconstructed from the estimated Gaussian and generated from z ~ *N*(0, 1).



Development

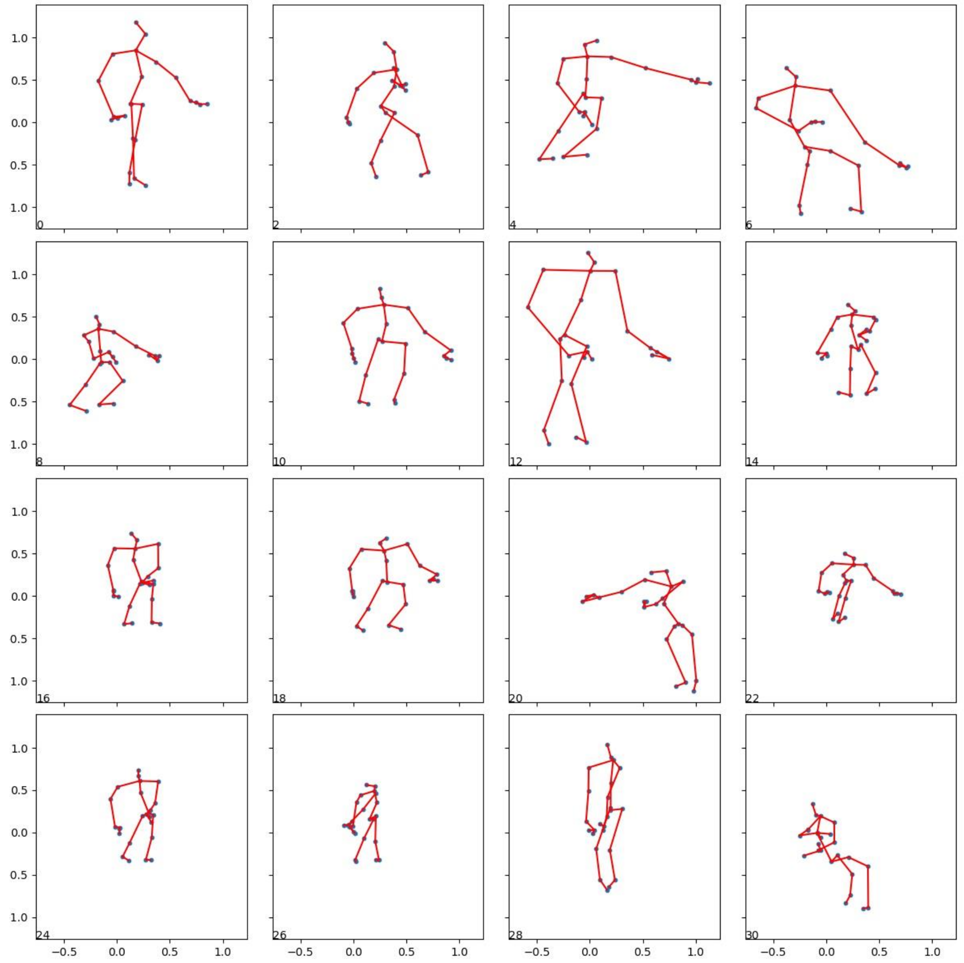
Right: generation from the VAE-WGAN with hidden layers dimension = 50 and 30, latent space dimension = 20. The bottom left digits are not frame number but simple indices of samples.

Note that the generation is by decoding a latent variable z ~ *N*(0, 1), while all "clusters" the VAE learned did not have to clump together tightly in one or two stdev of the standard normal distribution. A more effective method was to sample z in a Gaussian mixture model, where the means and variances were achieved from clustering the outputs of the trained encoder, and the weights were calculated from the percentages of cluster population.

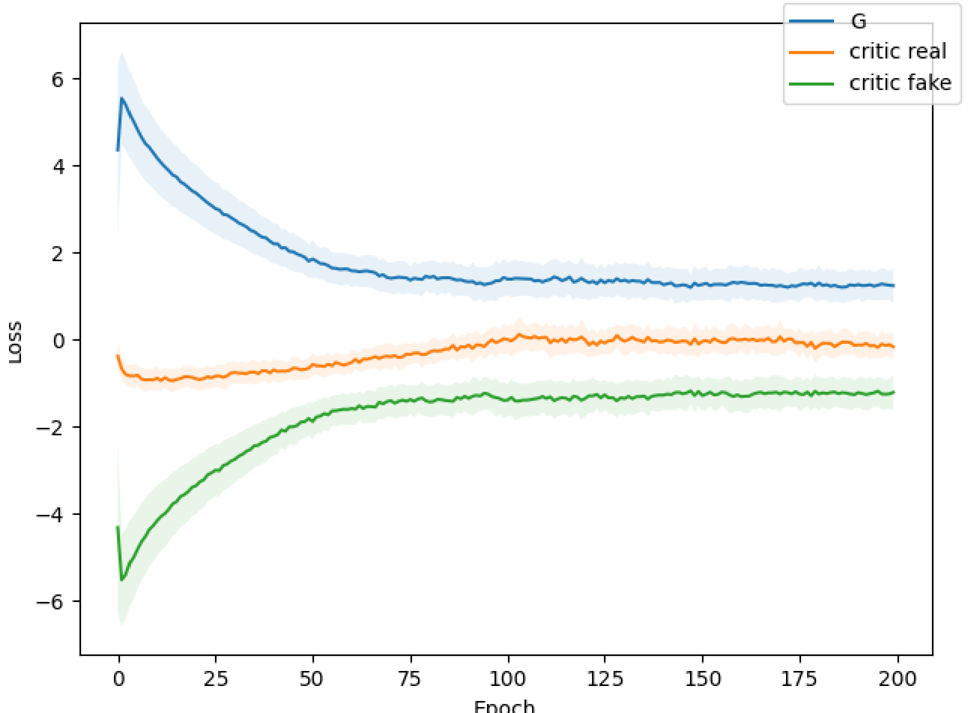


Development

We also tried typical WGAN for single frame generation. Even though we did not quantatively analyze the difference, the results from the typical WGAN generally seemed to be better than VAE-WGAN with z ~ *N*(0, 1).



Right: latent dimension = 50, three hidden linear-BN-ReLU blocks with dimension = 1024.



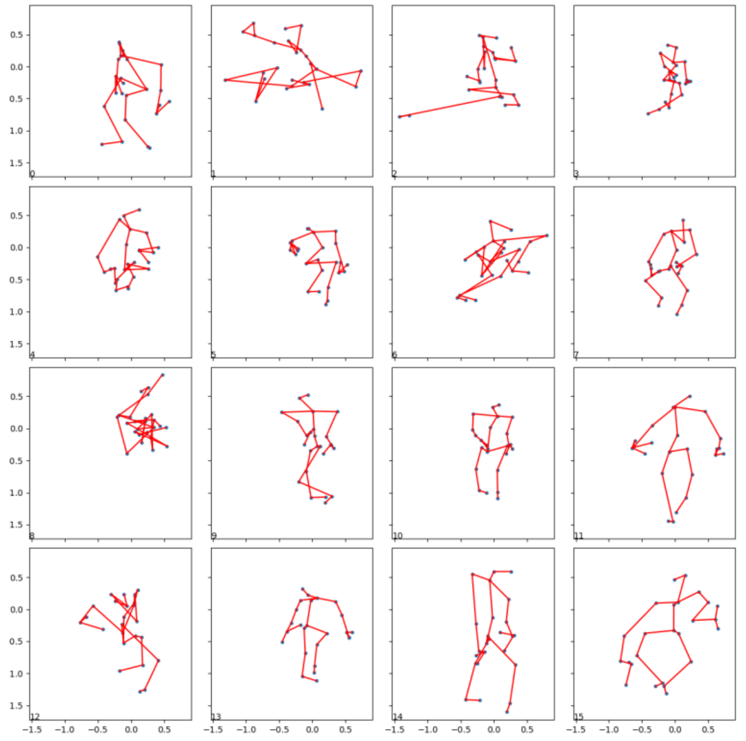
Bottom: training losses for the structure above.
"G" stands for generator loss.

Development

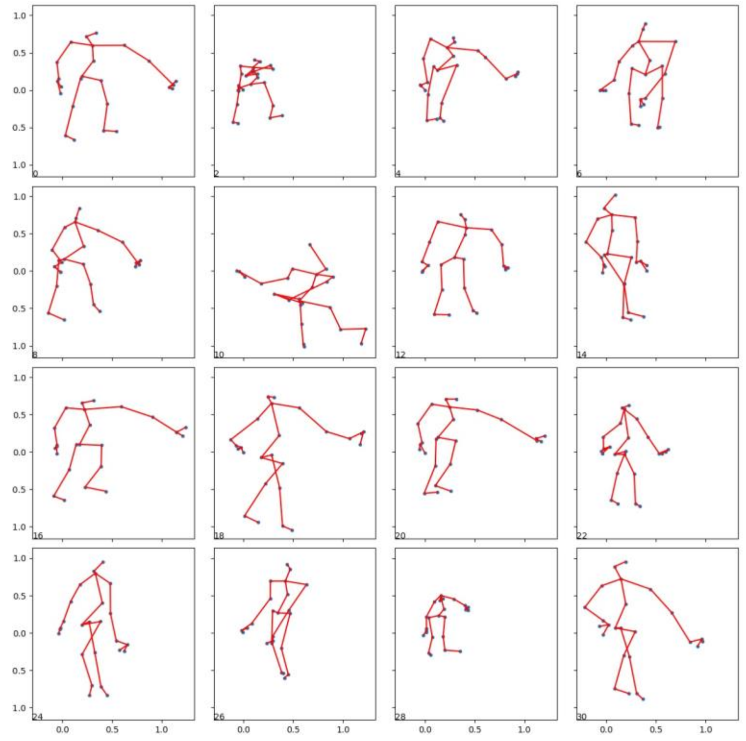
Initially, we thought that WGAN with conv layers may perform better than linear layers.

However, after several experiments, we found that simple WGAN model with linear layer did a better job to generate fake skeleton.

The reason is probably that the complexity of our skeleton data is much simpler than image data. Thus simple model works better in this situation.



Fake skeleton generated by wgan with conv layer



Fake skeleton generated by wgan with linear layer

New ideas

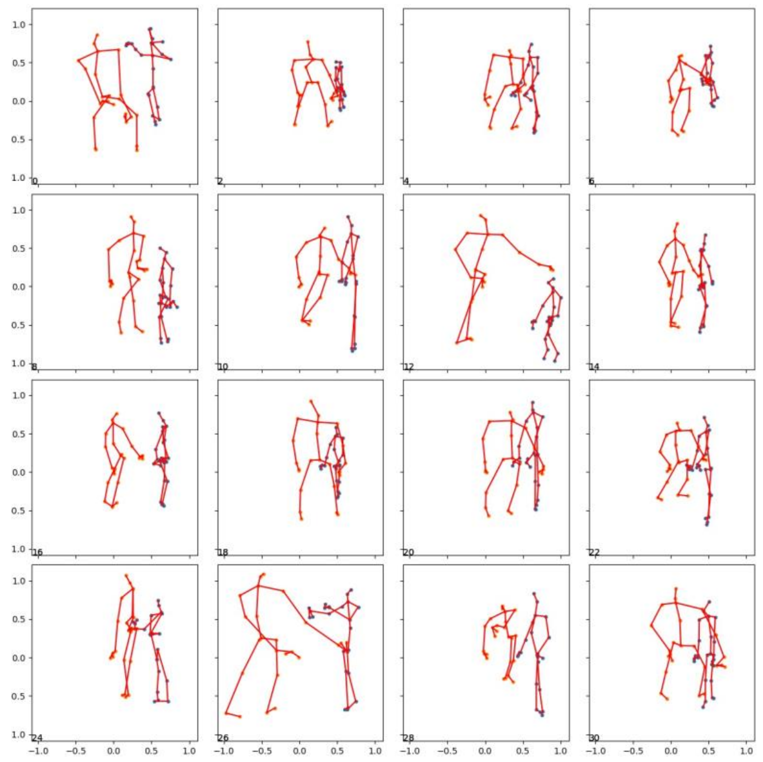
Previously, we mentioned that we can generate the second stick figure by mapping from the first stick figure such that it can learn the interactions between two individuals.

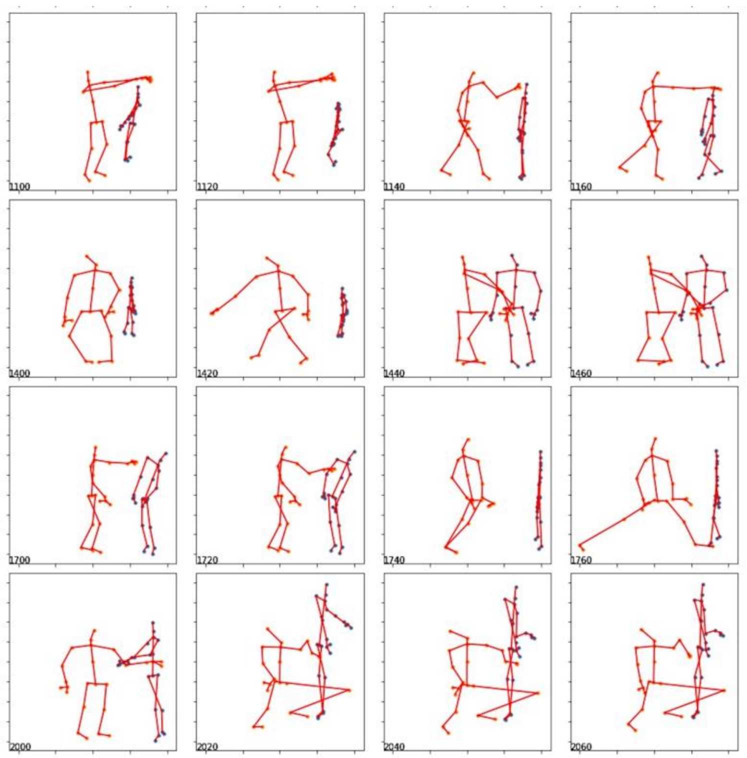
However, this may require additional training time and might unable to learn meaningful interaction.

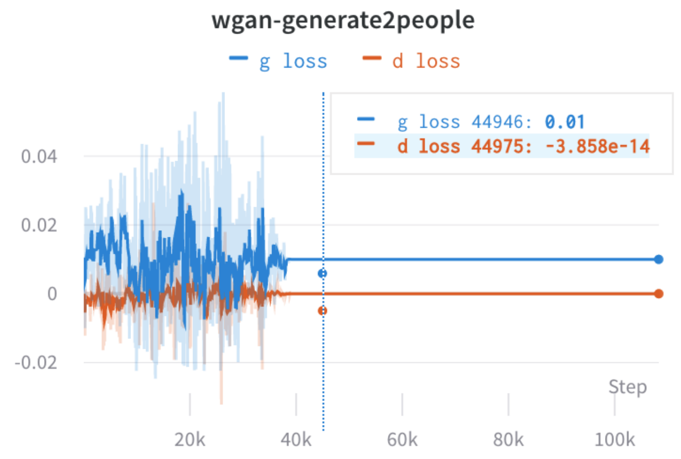
Why don't we just generate stick figure of two people simultaneously? We just need to change a few lines of code with same WGAN model mentioned before!

Our experiments shows that the new idea does a good job compared to the real figure. The two fake skeletons have good shapes, and their interactions make sense. You can even notice the relative position between two skeletons.

To understand the loss of generator and discriminator more clearly, we plot them in steps. From the plot, we can see that the g loss and d loss are converging to about 0 after 35k steps.


Generated figure


Real figure



Model Structure – Code Snippet

```
class Generator(nn.Module):
  def __init__(self):
    super(Generator, self).__init__()

    def block(in_feat, out_feat, normalize=True):
      layers = [nn.Linear(in_feat, out_feat)]
      if normalize:
        layers.append(nn.BatchNorm1d(out_feat, 0.8))
      layers.append(nn.ReLU())

      return layers

    self.model = nn.Sequential(
      *block(latent_dim, 128, normalize=True),
      *block(128, 256),
      *block(256, 512),
      *block(512, 1024),
      nn.Linear(1024, PERSON*KEYPOINTS*DIM),
    )

//skeleton_shape = (PERSON, KEYPOINTS, DIM)
  def forward(self, z):
    skeleton = self.model(z)
    skeleton = img.reshape(skeleton.shape[0],
*skeleton_shape)
    return skeleton
```

Generator (G)

```
class Discriminator(nn.Module):
  def __init__(self):
    super(Discriminator, self).__init__()

    self.model = nn.Sequential(
      nn.Linear(int(np.prod(skeleton_shape)), 512),
      nn.ReLU(),
      nn.Linear(512, 256),
      nn.ReLU(),
      nn.Linear(256, 128),
      nn.ReLU(),
      nn.Linear(128, 64),
      nn.ReLU(),
      nn.Linear(64, 1),
    )

//skeleton_shape = (PERSON, KEYPOINTS, DIM)
  def forward(self, skeleton):
    skeleton_flat = img.view(skeleton.shape[0], -1)
    validity = self.model(skeleton_flat)
    return validity
```

Discriminator (D)

After trying many experiments by ourselves and default hyperparameters provided by WGAN paper*, we found the following hyperparameters perform well.

Latent dim = 50
Learning Rate = 0.00005
Optimizers for G and D: RMSprop
Number of training steps for D per iter: 5
Lower and upper clip value for D weights: 0.01

*Martin Arjovsky *et al.* (2017). Wasserstein GAN arXiv:1701.07875

Evaluation

- Inception Score(IS)
    - Evaluates both the variety and quality of the generated samples
    - Measure the exponential of the KL divergence between the actual label distribution p(y) and the label distribution given the generated samples p(y|x)
- Frechet Inception Distance (FID)
    - Measures the distance between the feature spaces of generated samples and real samples using a feature extractor such as CNN
    - The distance is based on the means and covariances of the features

Compare the two scores with the other baseline models* on the same task.

* Step 1 can be compared with: Haoye Cai *et al.* (2018). Deep Video Generation, Prediction and Completion of Human Action Sequences. ECCV, 374–390. DOI: 10.1007/978-3-030-01216-8_23
* Step 2 can be compared with CGSN: Sijie Yan *et al.* (2019). Convolutional Sequence Generation for Skeleton-Based Action Synthesis. ICCV. DOI: 10.1109/ICCV.2019.00449

Future Work

- Finish the sequence and interaction generators
  - We learned LSTM and transformer. How does UNet perform, compared with these two?
- Try to improve the model
  - How much performance gain can we achieve if we were to use WGAN-GP or LSGAN?
- "Fill in the blank"
  - Watching joints only can be confusing sometimes. For example, it may be hard to identify if a skeleton is facing or turning its back towards the audience. This information, however, is important in storytelling. Can we train a model to fill the skeletons smartly, such that the audience can distinguish such difference?
- Composite actions
  - We only proposed to generate a series of two simple, correlated actions to tell stories. However, we did not consider about the gap between the actions.
  - Can we train the generator, or a separate generator, to connect two simple actions in a way that is more realistic than linear interpolation?
  - Can we generate more than two skeletons with meaningful interactions?
- Evaluation
  - After finishing our improved model, we will compare it with the baseline models mentioned in the previous slide by using IS and FID metrics.