Synthesizing Human Pose from Captions

ZHANG Yifei

Matriculation No. 383306

Master Thesis in Media Informatics

RWTH-Aachen University

Supervisors:

Rania Briq

Julian Tanke

Examiners:

Prof. Dr. Juergen Gall

Dr. Michael Weinmann

March 2020

Abstract

This thesis is about the implementation of a deep learning generative model that synthesizes single-person 2D poses from human-written captions. Previous research on related topics is well studied. The model is based on convolutional neural networks, generative adversarial networks and the fastText text encoder. It is trained and tested on the Microsoft COCO data set. Results of experiments with the different versions of the model are analyzed in detail. Qualitatively, the model can synthesize plausible poses matching the given captions. Several evaluation methods show the performance of the model: the model does well in the interpolation test, and it has promising performance in the quantitative measures and the subjective evaluation in the form of a user test. It is the first time that a generative adversarial network has been used in text-to-pose transfer.

Keywords: convolutional neural networks; generative adversarial networks; human poses

Contents

List of Figures

List of Tables

1. Introduction

The human pose is the configuration of the position of body parts. There are standing poses, sitting poses, playing tennis poses and so on. Human pose estimation from for example images or videos is a quite challenging task due to strong articulations, occlusions, unusual poses, etc. [28]. The emergence of convolutional neural networks (CNNs) and many deep learning techniques has brought on drastic improvement in solving pose estimation [28][29][30][31][7][8]. Despite the great advances in pose estimation in the scientific community, no one has tried to generate poses conditioned on text before.

That is where we would like to give a try. In this project, we will try to build a model that can automatically synthesize 2D single-person poses from human-written captions.

Figure 1 is an image of a person and one possible caption of this image may be "A snowboarder in mid air heading toward the ground." (This example is from the COCO data set, which we will use in this project). On the right is the pose of the person extracted from the image. The pose contains several "keypoints" of the body (e.g. shoulders, hips) and lines connection these keypoints (called the "skeleton" of the pose). Comparing the image and the pose, we can see that even though there is only a set of points and lines, the pose alone can still convey some information about the person in the image: the pose depicts a person probably doing a snowboard trick in the air.

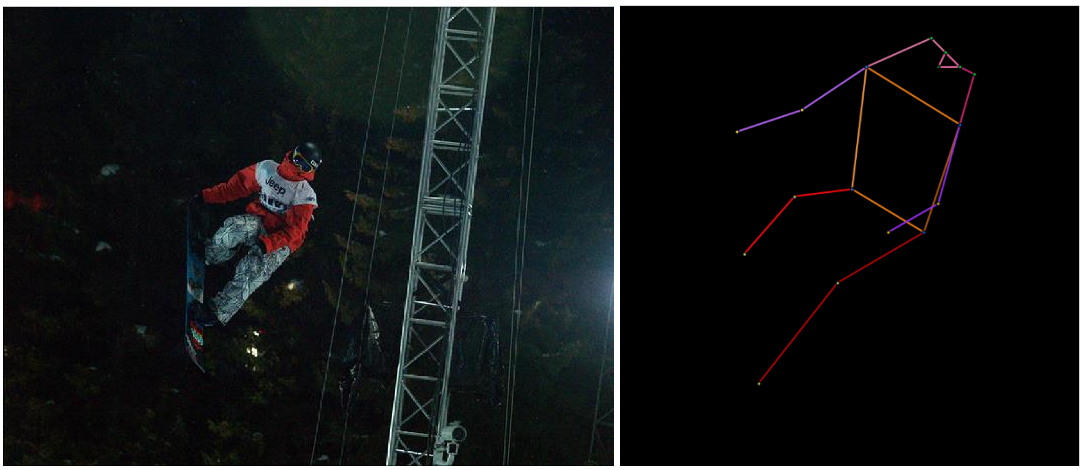


Figure 1 Left: a human image. The caption is "A snowboarder in mid air heading toward the ground.". Right: the pose of the person on the left image.

What we would like to achieve in this thesis is related to the poses and captions: from a caption describing a person doing something, such as the above "A snowboarder in mid air heading toward the ground.", the model can automatically synthesize a correct pose like the one in **Figure 1**. This is text-pose transfer and can also be described as pose generation conditioned on text. **Figure 2** shows what this is like.

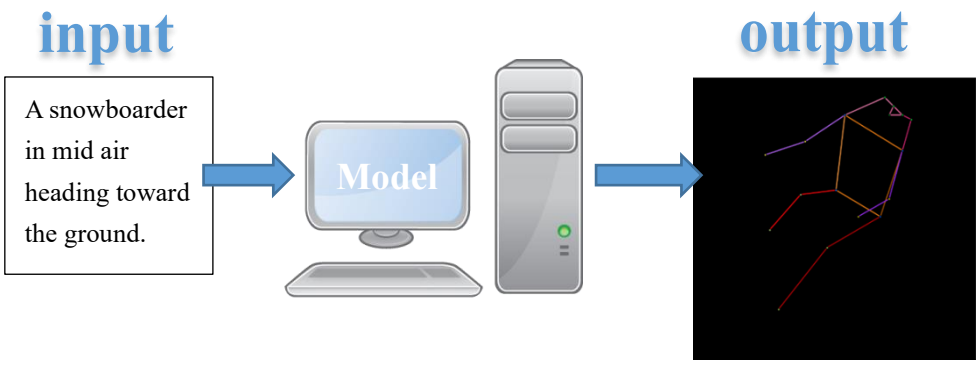


Figure 2 Model. Input: caption; output: human pose.

The motivation of our project originates from the recent attempts to synthesize images from text (text-image transfer). Reed et al. [3] succeed in synthesizing images of birds and flowers from human-written text (captions), but when it comes to synthesizing images of human beings, their outcome is far from ideal (**Figure 3** right): the synthesized human bodies are highly distorted. One possible explanation for this failure might be: the images of human beings are much more complicated than those of birds and flowers. Birds and flowers have simple shapes and rather fixed 'poses', while human beings composed of many articulated parts and their poses are highly changeable.

Our project might serve as a possible solution to the above problem. As it seems that human images are too difficult for models to generate directly, we can let them synthesize only the human poses as a first step. When realistic human poses can be synthesized, built upon the synthesized poses, complete human images are then synthesized, in the second step. We speculate that, in this hierarchical way, the difficulty of synthesizing human images can be diminished, and the results can be much more realistic.

Therefore, our strategy is to ignore the RGB (red, green and blue colors) information of the human images in our hand and focus only on the poses. The model then will be met with a problem with much clearer structure than the human image synthesis.

However, the above text-to-pose plus pose-to-image approach to better synthesize human images is only our hypothesis and, in this thesis, we will not really have a model to test this hypothesis. So this motivation is while promising, still speculative.

What's more interesting and more important about our project is that we would like to learn how well the GAN (generative adversarial network) can learn to generate human poses conditioned on text, given the complexity of human poses, and whether we can achieve semantic consistency between text encoding and pose features. The GAN is currently a hot research topic in the field of machine learning, and it has shown great competence in generating realistic images. But generating human poses conditioned on text has never been tried. This task is complex, because, as mentioned before, human poses are highly changeable. In this thesis we will use GAN to generate poses conditioned on text and examine whether it can achieve success in a task other than image generation.

Another highlight of this thesis is that the inputs to our pose synthesis model are just plain text (captions), but not class labels, categories, attributes or other values. This can give the users of the model much flexibility in specifying the desired poses and can also allow casual users to make use of our model. All that they need to do is give the model a written description about a human pose and then expect a plausible pose matching this description coming out of the model. In other words,

the model will be quite easy to use.

The model of pose synthesis conditioned on text might also have some interesting applications in artistic creation. For example, the poses synthesized from captions can be used in the creation of computer graphics (CG) of characters, including creating aminations from a sequence of continual synthesized poses. An ultimate goal might be that from a long paragraph of description a piece of realistic animation of the characters can be generated automatically.

In the following parts of the thesis, we will first discuss some other research work and background knowledge related to our project, with an emphasis on the GAN and related algorithms. Then we will explain in detail our methods and the architectures of our models. In the end, we will construct two versions of pose synthesis model: one unconditional model synthesizing human poses, and another model synthesizing human poses conditioned on captions. And we will conduct some experiments on those models to demonstrate how well they can perform. We will also evaluate the models qualitatively and quantitatively. In addition, we will do a ‘user test’, where we ask humans to distinguish between real and synthesized poses.

2. Related Work

2.1. Image Synthesis with GAN

In the field of image synthesis, the generative adversarial network (GAN) has been a power tool in recent years. It is first proposed by Goodfellow et al. [1]. As the term "generative adversarial" suggests, in this framework, two networks are simultaneously trained as a pair of adversaries: the generative network G to generate images, and the discriminative network D to discriminate samples from the training data and samples generated by G. G is trained to maximize the probability of D making a mistake, whereas D is trained to maximize the probability of making a correct decision. And after training, G can generate realistic images. They experiment on images of digits and human faces and the results demonstrate the potential of the GAN framework.

Besides plain image synthesis, GANs also have many other applications. For example, Radford et al. [2] use the GAN in unsupervised learning. Their GAN is trained on unlabeled photos. Both the discriminative and generative networks learn a hierarchy of representations from object parts to scenes in the images and can be used as feature extractors. Zhu et al. [5] use the GAN in image classification. The discriminative network is used as a classifier. The classification accuracy outperforms traditional methods, and the overfitting problem is also mitigated to a great extent. Liu et al. [4] use GAN in image-to-image transfer (generating images conditioned on images). They propose a model that automatically generates colorful cartoon images from black-and-white cartoon sketches. One of the highlights of the work is that a subjective test with volunteers is conducted to evaluate the image quality, and the test results show the good performance of the model. We will learn from their experience and also undertake such a test to evaluate our model.

Text-to-image synthesis (generating images conditioned on text) is another popular research topic about the GAN in recent years. Given a piece of human-written text, the model's job is to synthesize an image matching the text. Reed et al. [3] combine GAN with advanced text modeling (a deep symmetric structured text-image joint embedding) to develop a model to synthesize plausible images of birds and flowers from detailed text descriptions (**Figure 3** left). They also attempt to generalize their model to generating images with multiple types of objects and variable background, using the COCO data set, the same data set that we will use. However, the generated human images are not as successful as the birds and flowers (**Figure 3** right). Their approach does not directly address the human poses, and therefore the persons in the synthesized images seem to have strange poses. We will train our model on the same COCO data set but focus only on poses extracted from the images. Besides, we will apply the Wasserstein GAN algorithm [16][10] rather than the common GAN algorithm used in their work. We will see whether our model can generate more realistic poses, compared to this work.

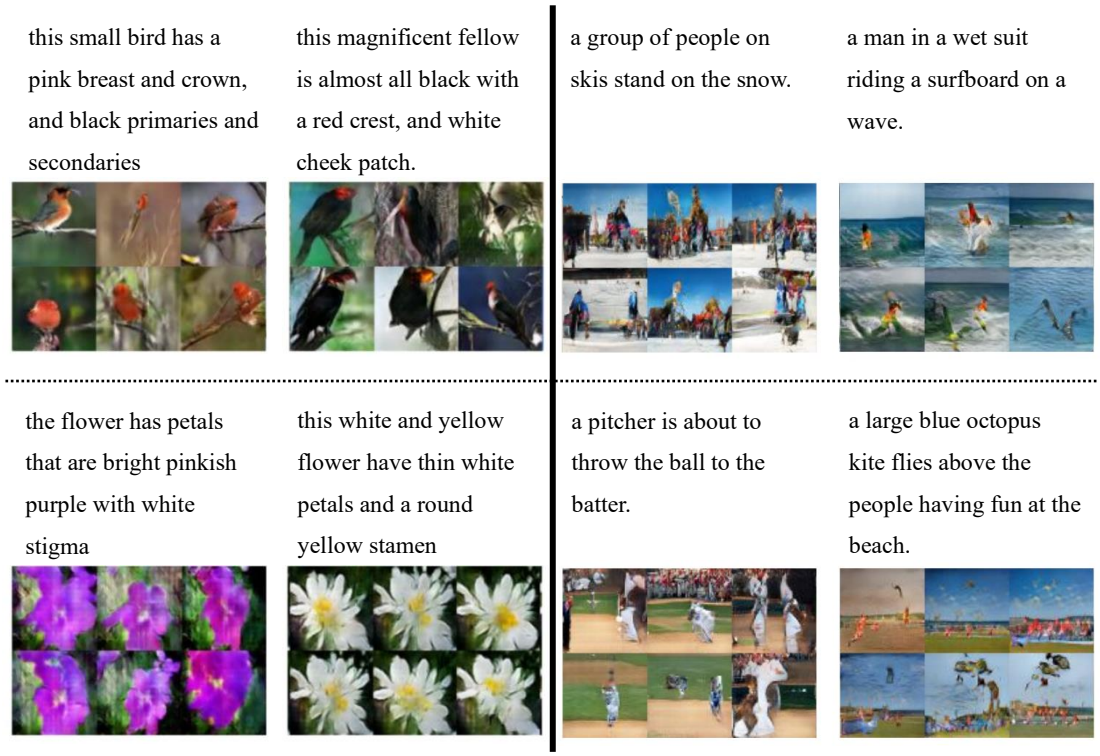


Figure 3 Examples of generated images from text descriptions, work from Reed et al. [3]. Left: birds and flowers. Right: people. Figure from Reed et al. [3].

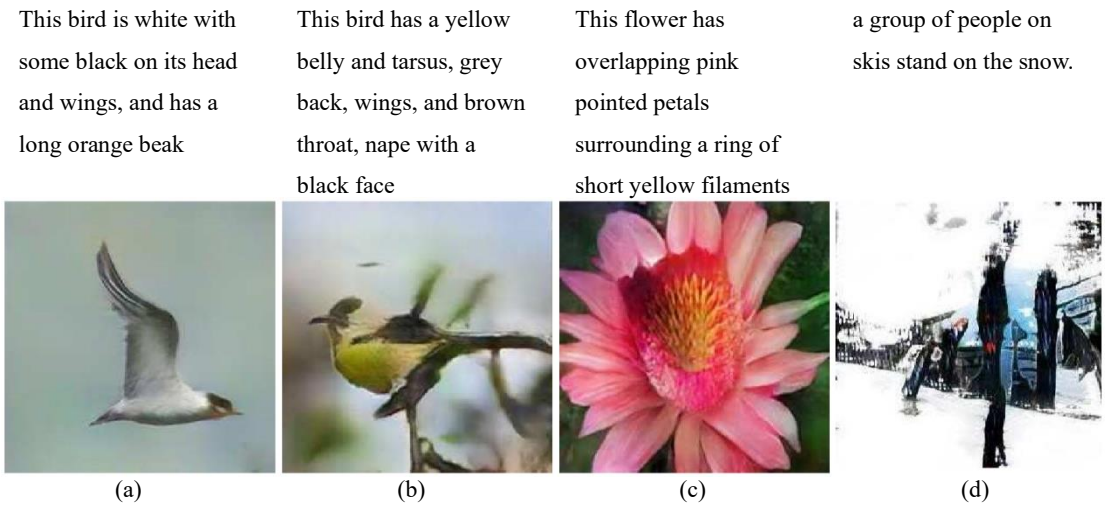


Figure 4 Examples of generated images from text descriptions, work from Zhang et al. [24]. (a) and (b): birds; (c): flower; (d): human beings. Figure from Zhang et al. [24].

Zhang et al. [24] make some improvement on the previous work of Reed et al. [3]. They propose the StackGAN, a GAN composed of two stages: the Stage-I GAN is conditioned on the text and generates low-resolution images with rough shapes and basic colors; the Stage-II GAN is conditioned on both the text and the result images of the Stage-I and generates high-resolution images, adding more details and correcting defects in the low-resolution images. They also add augmentation to the text conditioning to improve the GAN training. They train their model on the same bird and flower data sets as Reed et al. [3], as well as the COCO data set. Their bird and flower

results (**Figure 4** (a), (b) and (c)) are improved, compared with Reed et al. [3]. But the synthesized human images are still bad, as in **Figure 4** (d). The caption for this image is the same as the first set of synthesized human images in **Figure 3**, and we still cannot identify proper human poses from the image. This further confirms our argument: without explicitly handling the poses, even with more details, defect correction and text conditioning augmentation, it is still difficult to synthesize human images with plausible poses from captions. Nevertheless, this work is inspiring to us in that, in the future, for human image synthesis, we can adopt the stacked GAN architecture like theirs. Our text-to-pose model may play the role of the first stage, and the second stage may be a GAN generating RGB human images conditioned on the poses from the first stage.

2.2. Poses in Human Image Synthesis

Poses have been utilized in human image synthesis, especially in pose guided image transfer. That is, given a human image and a new pose, to synthesize an image of the human in the new pose. Most of the recent work of pose guided transfer also uses the GAN. In the work of Ma et al. [25], they use a 2-staged GAN to generate the image of a person in a novel pose from a condition image of the person and a target pose. Dong et al. [26] work on the same task. They propose a GAN with a more complex model structure. With this new structure their synthesized image quality is better than the previous work. Balakrishnan et al. [27] adopt another approach. They separate the condition image into foreground (the human) and background and then the two parts are processed separately. Their model is also capable of synthesizing coherent videos of actions given sequences of target poses, which is of great interest.

The pose synthesis model of our project can be combined with their image transfer models. The target pose can be replaced by a target caption. Then we can have an interesting model which takes an image of a person and synthesizes a new image of the same person according to a given caption.

2.3. Pose Estimation

Pose estimation, defined as the localization of human joints/keypoints, is an important problem in the field of computer vision [28]. There is 2D as well as 3D pose estimation. For 2D pose estimation, the input is often an image of human.

The first attempt in pose estimation based on CNNs is based on regression: Toshev and Szegedy [28] formulate 2D pose estimation as a regression problem. The independent variable is the image and the dependent variable is the coordinates of joints. They use a cascade of deep convolutional neural networks to do the regression and achieve unprecedent precision.

Wei et al. [29] adopt the second approach, formulating pose estimation as an image analysis problem. They propose another 2D pose estimation model also consisting of a cascade of convolutional neural networks but the output of each network is a “belief map” (we call it “heatmap” in this thesis) encoding the spatial probability of each keypoint. The networks operate directly on heatmaps, which provide rich information about spatial relationship between keypoints. Their model performs better than previous studies. In this project, we will also operate on heatmaps instead of keypoint coordinates.

Cao et al. [31] propose an efficient method for multi-person pose estimation. There can be two different approach to solve the multi-person pose estimation problem. The first approach is top-down, first detecting persons and then estimating the pose for each detected person [33]; the second one is bottom-up, first detecting the body keypoints and then associating them to individual persons [34]. The top-down approaches can leverage existing techniques for single-person pose estimation but the person detector sometimes fails due to occlusions, close proximity or partial visibility of people and the top-down approaches become less robust; the bottom-up approaches do not suffer from the above problem but they do not use global contextual cues and the final associating part is costly [31]. In the work of Cao et al. [31], they creatively take the bottom-up approach, using both location and orientation of body parts, and achieve high-quality results. He et al. [30] use a general framework for object instance segmentation in pose estimation. They regard each keypoint as an instance and the segmentation mask only has a single pixel as foreground. Their approach can also achieve high accuracy in pose estimation of multiple persons, even outperforming the previous work.

2.4. Human Pose Synthesis

There are also other works about synthesizing human pose. For example, Martinez et al. [23] work on a 3D pose prediction model. The input is a sequence of previous 3D poses and the output is a sequence of predicted 3D poses. This is a sequence-to-sequence transfer and they deal with it using recurrent neural networks (RNNs). Their model is not conditioned on text, rather on previous poses in the sequence. In our project, the outputs are static images (heatmaps) and there is no sequence, so we do not need to use RNNs. But their work might be important reference if, in the future, we want to synthesize sequences of poses from text to, for example, create animation. In that case, the RNN might be a useful tool.

3. Background

3.1. Convolutional Neural Network

Convolutional neural networks (CNNs) are widely applied in the field of computer vision. Nowadays they are very commonly used for both image analysis and image synthesis. Because we use heatmaps to represent poses and heatmaps are also images, our task is actually also image synthesis. At present, most GAN image synthesis models are based on CNNs and our models will also be based on CNNs.

The basis is the convolution operation. Convolution is a common operation applied on images in image processing. For an input image of dimension $w \times h \times c$ $(width \times height \times channels)$, a filter or kernel of the same number of channels and usually square shape $(m \times m \times c)$ slides over the image spatially and dot products are computed. The filter can have stride (the number of pixels of each step when sliding) larger than one and the input image can be padded (zeros added around the border of the image). The dot product results form an output map called convolution features. This operation is convolution, which is illustrated in **Figure 5**. More than one filter can be applied, and if there are c' filters, the output convolution features will have c' channels (feature dimension $w' \times h' \times c'$, w', h' usually smaller than the input w, h). The output of one convolution can be the input of another convolution. And a CNN is made up of multiple layers of convolutions stacked together with nonlinear activation layers (and sometimes other layers such as maxpooling for downsampling) inserted between them. In CNNs, the parameters of the filters are learnable. CNNs are very suitable for handling images because the filters make use of the spatial structure of the images. In our model, we will use such a CNN as the discriminative network D.

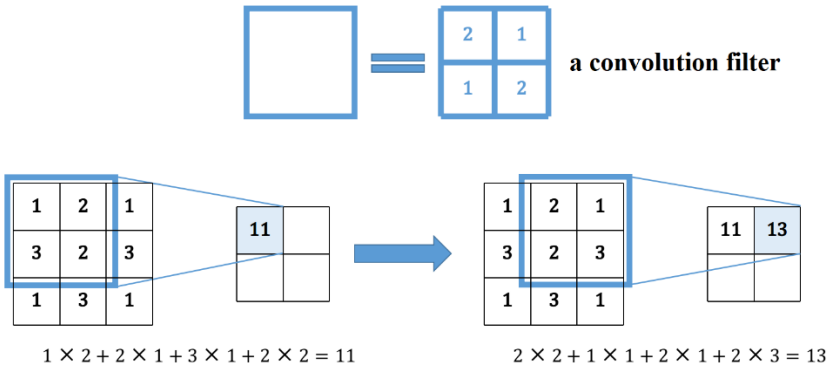


Figure 5 Illustration of convolution.

The transposed convolution [11] (also called fractionally-strided convolution) is just the 'opposite' operation of the convolution: as illustrated in **Figure 6**, the filter is multiplied by every value in the input and slides over the output, overlapping values summed up. Likewise, layers of transposed convolutions can form another type of CNN. In contrast to the first type of CNN, usually the output size (w', h') is larger than the input (w, h). We will use this type of CNN as our generative network G. This type of CNN is often used to generate images in the GAN framework, as it plays the role of upsampling and can produce images of certain resolution from some input noise (details will be explained later).

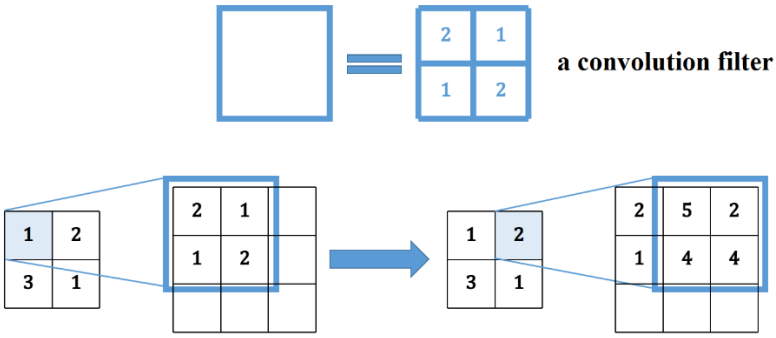


Figure 6 Illustration of transposed convolution.

3.2. Batch Normalization

The batch normalization is a technique to speed up the learning of deep neural networks such as CNNs, because it can normalize the distribution of the inputs to each layer so that the networks do not need to learn to adapt to the change of the input distributions [14]. In CNNs, it is usually inserted between a convolution layer and an activation layer. In the batch normalization, during training, each output $x_i, i = 1, \cdots, m$ of the previous layer over a minibatch of size m is first normalized by the mean and variance $\hat{x}_i = \frac{x_i - \mathrm{E}[x_i]}{\sqrt{\mathrm{Var}[x_i]}}$, then scaled and shifted $y_i = \gamma \hat{x}_i + \beta$, and then inputted to the next layer. The γ and β of each batch normalization are learnable parameters in the network and running averages of the means and variances of the x_i are recorded. And during test/evaluation, the recorded running averages are used for normalization.

3.3. Optimizers

The stochastic gradient descent (SGD) is the most basic algorithm to train neural networks. The idea is to get to the local minimum of the loss function by following the opposite direction of the gradient. Suppose that we have a network W_θ with learnable parameters θ, and x denotes a minabatch of data and L is the loss function. The SGD goes as follows.

Algorithm 1 SGD

Initialize parameters θ_0, learning rate η, number of iterations n

for $t = 1, \cdots, n$ **do**

Take a minibatch of data x

Calculate loss $L\left(W_{\theta_{t-1}}(x)\right)$

Calculate gradient $g_t = \nabla_{\theta_{t-1}} L\left(W_{\theta_{t-1}}(x)\right)$

Update parameters $\theta_t = \theta_{t-1} - \eta g_t$

end for

RMSProp (root mean square propagation) [18] and Adam (adaptive moment estimation) [15] are two of the more advanced optimizers extended on SGD. RMSProp adapts the learning rate of each parameter to its updating history. The parameters that have been changed more will be changed

less. The algorithm is:

Algorithm 2 RMSProp

Initialize parameters θ_0, learning rate η, number of iterations n, smoothing constant α, squared gradient $n_0 = \vec{0}$

for $t = 1, \cdots, n$ **do**

Take a minibatch of data x

Calculate loss $L\left(W_{\theta_{t-1}}(x)\right)$

Calculate gradient $g_t = \nabla_{\theta_{t-1}} L\left(W_{\theta_{t-1}}(x)\right)$

Update squared gradient $n_t = \alpha n_{t-1} + (1-\alpha) g_t \circ g_t$ ($\circ$ denotes elementwise product)

Update parameters $\theta_t = \theta_{t-1} - \eta \frac{g_t}{\sqrt{n_t}}$ (division and square root are done elementwise)

end for

Adam is like RMSProp but uses the momentum (running average of gradient) instead of just the gradient. **Algorithm 3** is the algorithm of Adam optimizer. Here the first momentum is the running average of the gradient, and the second momentum is the running average of the squared gradient, like in RMSProp.

Algorithm 3 Adam

Initialize parameters θ_0, learning rate η, number of iterations n, first momentum smoothing constant β_1, second momentum smoothing constant β_2, first momentum $m_0 = \vec{0}$, second momentum $n_0 = \vec{0}$

for $t = 1, \cdots, n$ **do**

Take a minibatch of data x

Calculate loss $L\left(W_{\theta_{t-1}}(x)\right)$

Calculate gradient $g_t = \nabla_{\theta_{t-1}} L\left(W_{\theta_{t-1}}(x)\right)$

Update first momentum $m_t = \frac{\beta_1 m_{t-1} + (1-\beta_1) g_t}{1-{\beta_1}^t}$

Update second momentum $n_t = \frac{\beta_2 n_{t-1} + (1-\beta_2) g_t \circ g_t}{1-{\beta_2}^t}$ ($\circ$ denotes elementwise product)

Update parameters $\theta_t = \theta_{t-1} - \eta \frac{m_t}{\sqrt{n_t}}$ (division and square root are done elementwise)

end for

3.4. GAN

3.4.1. Common GAN

The GAN (generative adversarial network) [1] is a framework of generative models. A generative model is a model whose outputs have a distribution P_g that estimates the distribution of real data P_r. For a perfect model, P_g and P_r should be identical and therefore nothing could tell an

output from the model and a sample of real data apart. The GAN is designed based on this idea. As mentioned before, a GAN consists of two parts: a generative network G (generator), and a discriminative network D (discriminator). Both G and D can be simple multilayer perceptrons, or the more complex CNNs or other types of networks. G is trained to generate 'fake' samples while D is trained to correctly discriminate between fake samples and samples of real data. The two networks are trained simultaneously. This is a two-player game and the theoretical training outcome is that G can generate such realistic fake samples that D cannot discriminate from real samples.

Mathematically, $G(z)$ represents an output fake sample from G, where z is a noise input to G. The noise $z \in \mathbb{R}^N$ is sampled from some simple noise distribution P_z, for example, normal distribution $N(0,1)$, and it is necessary so that the output samples of G can have some variance. $D(x) \in [0,1]$ is the discriminating output of D, representing the probability that the input sample x comes from the real data. In other words, $D(x)$ should be higher if x is a sample of real data and be lower if x is generated. So when D is being trained, the objective is to maximize both $\log\big(D(x)\big)$ with $x \sim P_r$ and $\log\Big(1 - D\big(G(z)\big)\Big)$ with $z \sim P_z$, that is, to correctly label both real samples and fake samples; when G is being trained, the objective is to minimize $\log\Big(1 - D\big(G(z)\big)\Big)$ with $z \sim P_z$, that is, to fool D with the generated fake samples. So the overall training objective is the following minimax game (here E means expected value):

$$\min_G \max_D \Big\{ \mathrm{E}_{x \sim P_r}\big[\log\big(D(x)\big)\big] + \mathrm{E}_{z \sim P_z}\Big[\log\Big(1 - D\big(G(z)\big)\Big)\Big] \Big\} \qquad \textbf{(1)}$$

Normally, gradient-based learning algorithms such as SGD are applied to train G and D. But early in the training, G is poor and D can discriminate samples generated by G from real samples with high confidence, meaning that $D\big(G(z)\big)$ is close to zero and the gradient signal of $\log\Big(1 - D\big(G(z)\big)\Big)$ is low, which is insufficient for G to learn [1]. Therefore, in practice, when G is trained, the objective is to maximize $\log\Big(D\big(G(z)\big)\Big)$ instead of minimizing $\log\Big(1 - D\big(G(z)\big)\Big)$, as $\log\Big(D\big(G(z)\big)\Big)$ will provide higher gradient signal early in the training when $D\big(G(z)\big)$ is close zero. It is obvious that both maximizing $\log\Big(D\big(G(z)\big)\Big)$ and minimizing $\log\Big(1 - D\big(G(z)\big)\Big)$ serve the same purpose – making G fool D with better samples. The above description can be summarized in the following GAN algorithm:

Algorithm 4 GAN

for number of training iterations **do**

for k steps **do**

Take a minibatch of m noise samples $\{z_1, \cdots, z_m\}$ sampled from noise distribution P_z

Take a minibatch of m real samples $\{x_1, \cdots, x_m\}$

Update D using the gradient:

$$-\nabla \frac{1}{m} \sum_{i=1}^{m} \left[\log\big(D(x_i)\big) + \log\Big(1 - D\big(G(z_i)\big)\Big) \right]$$

end for

Take a minibatch of m noise samples $\{z_1, \cdots, z_m\}$ sampled from noise distribution P_z

Update G using the gradient:

$$-\nabla \frac{1}{m} \sum_{i=1}^{m} \left[\log\Big(D\big(G(z_i)\big)\Big) \right]$$

end for

As shown, D and G are trained alternately: k iterations of updating D and then one iteration of updating G. Usually $k = 1$ is chosen, which is the least computationally expensive [1].

Theoretically, if D and G have infinite capacity, **Algorithm 4** GAN will eventually converge to the global optimum, where P_g and P_r are identical and the output of D is 0.5 for both real and fake samples [1]. In practice, since D and G have limited capacity, the global optimum cannot be reached. However, the algorithm can still produce reasonable results [1].

The idea behind the GAN is easy to comprehend and its architecture is not very complicated. But one of the challenging aspects of the GAN is the training or learning instability [2][10][16]. For example, one common failure of training GANs is called the "mode collapse", where the outputs of G collapse into only a few samples, because with these samples G can often successfully 'fool' the discriminator D [1][2]. Apart from the mode collapse, good training hyperparameters for stable learning are hard to find [16].

After the invention of the GAN, a lot of research has been done to improve the performance of the GAN. Radford et al. [2] do some experiments on GANs and propose a set of constraints on the architecture and training protocol of GANs to make them more stable to train. They put forward the Deep Convolutional GAN (DCGAN), a class of GAN with a specific architecture [2]. D and G of DCGAN contain layers of convolutions and transposed convolutions, respectively. In the generative network (**Figure 7**), transposed convolutions play the role of upsampling. And in the discriminative network, convolutions have strides greater than one for downsampling. This is different from usual CNNs, which use pooling functions (such as maxpooling) for downsampling. The nonlinear activation function used in G is ReLU (rectified linear unit: $y = \max(0, x)$) [12] while in D it is leaky ReLU ($y = \max(\alpha x, x)$, where $0 \leq \alpha < 1$ is a parameter) [13]. What's more, to spped up and stabilize training, batch normalization is applied after each layer of convolution/transposed convolution and before the nonlinear activation function, except for the first layer in D and the last layer in G. The learning of DCGAN with the Adam optimizer is relatively stable [2]. The models in our project are all based on DCGAN.

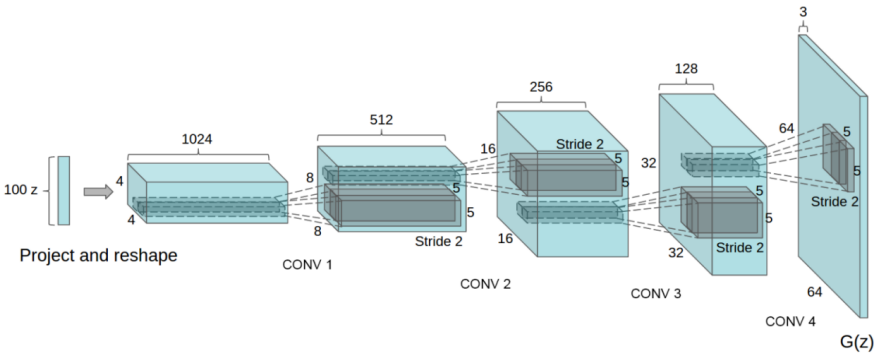


Figure 7 The generative network of the DCGAN. Figure from Radford et al. [2].

3.4.2. Wasserstein GAN

Arjovsky et al. [16] make extensive theoretical analysis and introduce a new GAN algorithm – the Wasserstein GAN (WGAN), different from the common GAN training in the last section. The WGAN algorithm further improves the stability of learning.

The WGAN is the improved GAN algorithm making use of the Wasserstein-1 distance to compare the two distributions P_r and P_g [16].

The optimization objective of GANs (**Expression (1)**) is actually related to $JSD(P_r \| P_g)$ the Jensen-Shannon divergence between P_r and P_g [1]:

$$\max_D \left\{ \mathrm{E}_{x\sim P_r}[\log(D(x))] + \mathrm{E}_{z\sim P_z}\left[\log\left(1 - D(G(z))\right)\right] \right\} = -\log 4 + 2 \cdot JSD(P_r \| P_g) \qquad \textbf{(2)}$$

where

$$JSD(P_r \| P_g) = \frac{1}{2} KL\left(P_r \middle\| \frac{P_r + P_g}{2}\right) + \frac{1}{2} KL\left(P_g \middle\| \frac{P_r + P_g}{2}\right) \qquad \textbf{(3)}$$

is the Jensen-Shannon divergence between P_r and P_g, where

$$KL(P \| Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right) \qquad \textbf{(4)}$$

is the Kullback–Leibler divergence of the distribution Q from the distribution P.

Both the Kullback–Leibler divergence and the Jensen-Shannon divergence measure how different two probability distributions are. Larger divergence means greater difference. It is easy to see that when $P_r = P_g$, $JSD(P_r \| P_g) = 0$, and when P_r and P_g are disjoint, $JSD(P_r \| P_g) = \log 2$. When P_r and P_g go from overlapping to disjoint or the other way around, $JSD(P_r \| P_g)$ will go through a discontinuous jump to or from $\log 2$. This is a cause for GANs' instability [16]. In the scenario of GAN, as images have very high dimensions, P_r and P_g are generally disjoint during the training process but this is just where the Jensen-Shannon divergence is discontinuous. So **Expression (1)** cannot provide very usable gradients for the network to learn.

The above phenomenon does not occur with the Wasserstein-1 distance $W(P_r, P_g)$, which is another measurement of two probability distributions' difference. When P_r and P_g go from overlapping to disjoint or the other way around, $W(P_r, P_g)$ will stay continuous. This can provide better gradients than the Jensen-Shannon divergence in GAN [16]. $W(P_r, P_g)$ can be calculated by the following equation [17]:

$$K \cdot W(P_r, P_g) = \sup_{\|f\|_L \leq K} \mathrm{E}_{x \sim P_r}[f(x)] - \mathrm{E}_{x \sim P_g}[f(x)] \quad \textbf{(5)}$$

Here K is some constant and $\|f\|_L \leq K$ means that f is a K-Lipschitz continuous function (f is K-Lipschitz continuous if there exists a constant $K \geq 0$ such that for all x, y from f's domain, $|f(x) - f(y)| \leq K|x - y|$. Intuitively, f's gradient cannot be larger than K anywhere). When f is realized by a neural network, clamping all the network's parameters to a fixed interval (such as $[-0.01, 0.01]$) can assure the K-Lipschitz requirement for some K [16]. In the context of GANs, the discriminator D can take the role of the above f to estimate the Wasserstein-1 distance between the real distribution P_r and fake distribution P_g. The output of D is no longer the probability that the input sample x comes from the real data, and there is no logarithm anywhere in the expression, so D's output need not be limited in the range $[0,1]$, i.e. $D(x) \in \mathbb{R}$. Meanwhile, the generator G's task is to reduce this distance by generating fake samples. So, the optimization objective becomes:

$$\min_G \max_D \{\mathrm{E}_{x \sim P_r}[D(x)] - \mathrm{E}_{z \sim P_z}[D(G(z))]\} \quad \textbf{(6)}$$

Here the maximization part corresponds to **Expression (5)**, estimating the Wasserstein-1 distance. The following is the WGAN algorithm:

Algorithm 5 WGAN

for number of training iterations **do**

  for k steps **do**

    Take a minibatch of m noise samples $\{z_1, \cdots, z_m\}$ sampled from noise distribution P_z

    Take a minibatch of m real samples $\{x_1, \cdots, x_m\}$

    Update D's parameters using the gradient:

$$-\nabla \frac{1}{m} \sum_{i=1}^{m} [D(x_i) - D(G(z_i))]$$

    Clamp D's parameters to $[-c, c]$

  end for

  Take a minibatch of m noise samples $\{z_1, \cdots, z_m\}$ sampled from noise distribution P_z

  Update G's parameters using the gradient:

$$-\nabla \frac{1}{m} \sum_{i=1}^{m} [D(G(z_i))]$$

end for

Algorithm 5 has a hyperparameter c for parameter clamping. As for k, usually a value much larger than 1 is chosen, because the more the discriminator is trained, the better the estimated Wasserstein-1 distance will be, and the more reliable gradients G can get [16]. Momentum-based optimizers such as Adam (**Algorithm 1** with first momentum smoothing constant $\beta_1 > 0$) will lead to unstable training, so RMSProp, which does not use momentum, is used for training WGAN [16].

The WGAN has two obvious advantages over the common GAN [16]: First, unlike the common GAN, the WGAN discriminator loss correlates well with the quality of the generated samples. During the training, the discriminator loss's absolute value gets smaller as the generated samples get better. This provides a convenient way to check if the training goes well. Second, the training stability improves, and the mode collapse problem does not occur.

On the other hand, the WGAN of **Algorithm 5** still suffers from some problems. The parameter clamping applied to enforce the Lipschitz constraint can lead to some undesired behaviors of the networks [10]: the learned distribution P_g may be very simple approximations to the real distribution P_r; during training the gradients propagated in farther back layers of the networks may either vanish or explode unless the clamping threshold c is carefully tuned.

Gulrajani et al. [10] propose an improved WGAN algorithm which performs even better than the standard WGAN. They suggest an alternative way to enforce the Lipschitz constraint – the gradient penalty. As a 1-Lipschitz continuous differentiable function has gradient of norm of at most 1, we can directly constrain the discriminator output's gradient with respect to the input. This is realized by adding a penalty term on the gradient norm with respect to some random samples $\hat{x}\sim P_{\hat{x}}$ to D's loss function. $\hat{x}$ is sampled uniformly along straight lines between pairs of points sampled from P_r and P_g. Constraining gradient norms at such $\hat{x}$ is a sufficient measure to enforce D's Lipschitz constraint in practice [10]. Therefore D's loss function is changed to:

$$\mathrm{E}_{z\sim P_z}\big[D\big(G(z)\big)\big] - \mathrm{E}_{x\sim P_r}[D(x)] + \lambda \mathrm{E}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \qquad \textbf{(7)}$$

Here the third term is the added gradient penalty term and λ is the penalty coefficient. Below is the new WGAN algorithm with gradient penalty (WGAN-GP):

Algorithm 6 WGAN-GP

for number of training iterations **do**

for k steps **do**

Take a minibatch of m noise samples $\{z_1, \cdots, z_m\}$ sampled from noise distribution P_z

Take a minibatch of m real samples $\{x_1, \cdots, x_m\}$

Take a minibatch of m random numbers $\{\epsilon_1, \cdots, \epsilon_m\}$ sampled from the uniform distribution $U[0,1]$

Calculate m samples $\{\hat{x}_i | \hat{x}_i = \epsilon_i x_i + (1-\epsilon_i)\cdot G(z_i), i = 1, \cdots, m\}$

Update D using the gradient:

$$\nabla \frac{1}{m}\sum_{i=1}^{m}\left[D\big(G(z_i)\big) - D(x_i) + \lambda\left(\left\|\nabla_{\hat{x}_i} D(\hat{x}_i)\right\|_2 - 1\right)^2\right]$$

end for

Take a minibatch of m noise samples $\{z_1, \cdots, z_m\}$ sampled from noise distribution P_z

Update G using the gradient:

$$-\nabla \frac{1}{m}\sum_{i=1}^{m}\big[D\big(G(z_i)\big)\big]$$

end for

For WGAN-GP, there should not be batch normalization in D, since the loss function (**Expression (7)**) contains gradient with respect to each input sample, not with respect to a batch [10]. The algorithm of WGAN-GP can overcome the above-mentioned problems of WGAN with parameter clamping: the generated images have higher quality and the hyperparameter tuning also becomes easier [10].

In our experiment, we will try different GAN algorithms (common GAN, WGAN, WGAN-GP) to train our model and compare the performance.

3.5. FastText Model

A text encoder is a function that maps a piece of text to a vector: $\varphi: \mathbb{T} \to \mathbb{R}^T$, where $\mathbb{T}$ is the set of all possible text and T is the dimension of the encoding vectors. On the word level, a word encoder is $\varphi: \mathbb{W} \to \mathbb{R}^T$, where $\mathbb{W}$ is the set of all possible words. Good text encoding vectors can be very useful in many natural language processing tasks. In our project, we need to get viable representations of the captions, too, which can be used as input features to our model.

"fastText" [21] is a model to encode words into vectors (a library is also available: https://fasttext.cc/). Words' vector representations (also called encodings and embeddings) are derived from large language corpora. The basic idea is that each word is represented as a bag of character n-grams. Special boundary symbols < and > are added at the beginning and end of words, allowing to distinguish prefixes and suffixes. The word itself is also included in the set of its n-grams. Taking the word "where" and n = 3 as an example, it will be represented by these character n-grams: <wh, whe, her, ere, re> and <where>. The vector representation of a word is the sum of the vector representations of its n-grams. The vector representations of n-grams are trained such that the scalar products (indicating similarity) between a word's vector and its context words' vectors are large while the scalar products between the word's vector and random words' vector are small [21].

More specifically, given a word w, G_w is the set of n-grams of w and z_g is the vector representation of an n-gram g. Then the vector representation of w is $u_w = \sum_{g \in G_w} z_g$. Given another word c with vector representation v_c, the scoring function s of w and c is $s(w, c) = u_w^\top v_c$. During the training of the model, there is a dictionary storing all the n-grams existing in the language corpus with their vector representations, and the training corpus contains T words. For the word at position t, the context C_t is the set of positions surrounding t (considered as positive examples), and N_t is a set of words randomly from the corpus (considered as negative examples). The probability of the presence of a positive example word w_c ($c \in C_t$) given w_t is modeled as a logistic function $p(w_c|w_t) = \frac{1}{1+e^{-s(w_t,w_c)}}$, while probability of the absence of a negative example word n ($n \in N_t$) given w_t is also modeled as a logistic function $q(n|w_t) = \frac{1}{1+e^{s(w_t,n)}}$. And the minimization objective is the negative log-likelihood:

$$\sum_{t=1}^{T} \left[\sum_{c \in C_t} \log\left(1 + e^{-s(w_t,w_c)}\right) + \sum_{n \in N_t} \log\left(1 + e^{s(w_t,n)}\right) \right] \quad \textbf{(8)}$$

This objective makes the scores with context words (positive examples) large and the scores with random words (negative examples) small.

The fastText model can do well in tasks such as word similarity and word analogies (A is to B what C is to D) [21]. Besides being simple and fast to train, this model has the advantage that it considers not only language structures but also internal structures of words due to the n-grams, and it can also properly represent words that have not appeared in the training corpus [21].

The above is about encodings of individual words, from which encodings of captions (sentences) can be derived. Although the fastText model itself is a word encoder, sentence representation based on the word encoder is also provided in the fastText library. A sentence is represented by the average of normalized vectors of all its words: Suppose that we have a trained

word model φ, given a sentence s containing n words $\{w_1, \cdots, w_n\}$. The encoding of s will be $\varphi(s) = \frac{1}{n}\sum_{i=1}^{n} \frac{\varphi(w_i)}{\|\varphi(w_i)\|}$. This sentence representation is word based, and the word order does not count here. However, we still choose to use fastText to encode the captions in our project, as it is easy and fast, and there are many pre-trained models available online, which we can simply download and use directly.

4. Method

4.1. Pose and Caption Processing

In the model, the information of a pose (the location of each keypoint) can be used directly as point coordinates in a regression task, as in the work of Toshev and Szegedy [28]. But transforming it into a "heatmap" is a better way, since heatmaps, as images in the general sense, preserve more information about the spatial relationship between the keypoints in poses than coordinates do. Also, it will be more convenient to process them like images with our models based on DCGAN. We will process it in this way: Each keypoint corresponds to one channel of the image. In each channel, if the corresponding keypoint is not visible, all the values will be zero (this is because some poses in the training data set only have part of keypoints visible due to occlusion); if the keypoint is visible, the values will have a bell-shaped profile, with the center in the place of the keypoint and the maximum value one. More specifically, for a visible keypoint, the value of the point (u, v) in this channel will be $e^{-\frac{(u-u_0)^2+(v-v_0)^2}{2\sigma^2}}$, where (u_0, v_0) are the coordinates of the keypoint and σ controls the width of the bell shape. Such kind of image representing pose keypoints is called a "heatmap" $x \in [0,1]^{w\times h\times c}$, where w and h are the width and height of heatmap and v is the number of channels in the heatmap. The underlying principle of the heatmap is that, the higher the value of a point in the heatmap is, the more likely the pose keypoint is to lie on this point.

To improve training, we will also perform some augmentation on the training heatmaps: they are randomly horizontally flipped and randomly rotated by $-10°\sim10°$ around the center.

The outputs of our models will also be heatmaps and we take the maximum point in each channel as the location of the corresponding keypoint. When the maximum value of one channel is below the threshold (we take 0.2), we then decide that the corresponding keypoint does not appear (is not visible) in the output pose.

The captions will be encoded by a pre-trained fastText model from fastText's website. This model is trained on the English Wikipedia and encodes a piece of text (the caption) into a vector of 300 dimensions: $\varphi: \mathbb{T} \to \mathbb{R}^{300}$. We expect it to work well because it has been trained on a very large text corpus, much larger than the quantity of captions in the data set that we will use for training our pose synthesis model.

4.2. Two Models

In this thesis, we will set up two models, one unconditional model and one conditional model.

The first model, the unconditional model merely synthesizes human poses without considering any captions. We build this model to verify if the GAN framework, which can successfully synthesize realistic images in previous work, is also able to synthesize realistic human poses in general, no matter what the captions for the poses would be.

The second model, the conditional mode will synthesize poses conditioned on captions. This is the main model of the thesis. We do this on the basis of the simpler unconditional model because our strategy is starting with simple things before going on with more complicated things. In our

opinion, only when we have succeeded in synthesizing general human poses, can we synthesize caption-specified human poses. This strategy has another advantage: we can make sure that the conditional model can indeed make use of the text encodings, compared to the unconditional model, since, for example, if the unconditional model succeeds and the conditional model fails, then the model cannot really make sense of the caption encodings.

4.3. Conditional GAN Algorithm

As for the conditional model, we will use GANs to synthesize human poses from captions. But in this scenario, the GAN is not the simple unconditional GAN described in **Section 3.4** because we have a condition – the caption text t. Both G and D take t as an additional condition input: G generates fake samples from t and the input noise z, while D gets data samples paired with t and gives outputs. That is the conditional GAN.

Reed et al. [3] give a good example work of conditional GANs. Their task is synthesizing images of birds and flowers from captions. This is very similar to our task, if we, as mentioned in **Section 2.3**, also take poses as images (heatmaps), although unlike us Reed et al. [3] do not use WGAN. We will follow their steps and refer to Bodnar [6] as well, who uses WGAN for text to image synthesis.

First, captions need encoding into vectors in order to synthesize poses. We will use the fastText model (**Section 3.5**). A caption t will be encoded into a vector $h \in \mathbb{R}^T$ by the text model φ: $h = \varphi(t)$. Then we can formulate our conditional GAN mathematically: $G: \mathbb{R}^N \times \mathbb{R}^T \to \mathbb{R}^{w \times h \times c}$ and $D: \mathbb{R}^{w \times h \times c} \times \mathbb{R}^T \to \mathbb{R}$, where N is the dimension of the noise input to G, T is the dimension of the caption vector and $w \times h \times c$ is the dimension of the image (heatmap). In contrast, the unconditional GAN model is simpler, formulated as: $G: \mathbb{R}^N \to \mathbb{R}^{w \times h \times c}$ and $D: \mathbb{R}^{w \times h \times c} \to \mathbb{R}$.

Because our conditional GAN has conditional input, the training process is a little different from that of the unconditional GAN. As the discriminator encounters two types of errors: first, unrealistic images and second, realistic images with mismatching captions, separating them explicitly during training can help the discriminator learn better [3]. So in WGAN the Wasserstein-1 distance estimation is changed to:

$$\max_D \{ \mathrm{E}_{(x,h)\sim P_r, z\sim P_z}[D(x,h) - D(G(z,h),h)] - \alpha \mathrm{E}_{(x,h)\sim P_r, \hat{h}\sim P_h}[D(x,h) - D(x,\hat{h})] \} \quad \textbf{(9)}$$

Here in the conditional GAN, both D and G have an additional input: the caption encoding h or $\hat{h}$. $(x,h)\sim P_r$ is a pair of matching image and caption encoding from the training data set, while $\hat{h}\sim P_h$ is a random caption encoding from the training data set, independent of x. In **Expression (9)**, the two expectation terms represent the above-mentioned two types of errors, respectively, and α is a coefficient to control the level of text-image matching [6].

Another modification is adding interpolated caption encodings in the training of the generator. Another term is added to G's maximization objective:

$$\mathrm{E}_{z\sim P_z, h\sim P_h}[D(G(z,h),h)] + \mathrm{E}_{z\sim P_z, \tilde{h}\sim P_{\tilde{h}}}[D(G(z,\tilde{h}),\tilde{h})] \quad \textbf{(10)}$$

where

$$\tilde{h} = \beta h_1 + (1-\beta) h_2 \quad \textbf{(11)}$$

Here h, h_1 and h_2 are caption encodings from the training data set; β is the interpolation coefficient. This $\tilde{h}$ is an interpolated encoding, not corresponding to any real text. However, this

interpolation adds many more training encodings and can help the generator learn better [3].

In WGAN, D must be Lipschitz continuous. In the conditional scenario, D has two inputs: the data sample x (image) and the condition h (caption encoding). So if the Lipschitz constraint is enforced by limiting the gradient, the gradient $\nabla_{\hat{x},h} D(\hat{x}, h)$ is with respect to $\hat{x}$ and h together. $\hat{x}$, as previously, is sampled between a real data sample x and a generated data sample $G(z, h)$ conditioned on x's matching caption's encoding h. More specifically,

$$\hat{x} = \epsilon x + (1 - \epsilon) \cdot G(z, h) \quad \textbf{(12)}$$

where ϵ is a random number sampled from $U[0,1]$ the uniform distribution from 0 to 1.

Moreover, in the WGAN-GP algorithm, the Lipschitz constraint is enforced by adding a penalty term with coefficient λ in D's loss function (**Expression (7)**). However, in the conditional GAN, as D's loss can be much larger because there are two errors – unrealistic and mismatching , λ should also be set larger, but WGAN-GP algorithm is not very robust to the change of λ [6][20]. Petzka et al. [20] proposes a new strategy called the Lipschitz penalty (LP), which is robust to λ. It just changes the penalty to be one-sided, meaning encouraging the norm of the gradient to stay below 1. The penalty term is therefore:

$$\mathrm{E}_{(\hat{x},h)\sim P_{\hat{r}}} \left[\left(\max\left(0, \left\| \nabla_{\hat{x},h} D(\hat{x}, h) \right\|_2 - 1\right)\right)^2 \right] \quad \textbf{(13)}$$

where h is a caption encoding from the training set with the real image x, and $\hat{x}$ is sampled uniformly along straight lines between the real image x and the generated image $G(z, h)$.

In contrast, in WGAN-GP, the penalty is two-sided (**Expression (7)**), meaning encouraging the norm of the gradient to go towards 1 in either direction [10].

To sum up, we will use the following conditional WGAN-LP algorithm (for comparison, here we also put down WGAN-GP, which does not have the maximum in the penalty term) to train our conditional model:

Algorithm 7 Conditional WGAN-LP (and WGAN-GP)

for number of training iterations **do**

for k steps **do**

Take a minibatch of m noise samples $\{z_1, \cdots, z_m\}$ sampled from noise distribution P_z

Take a minibatch of m real heatmap samples with matching and mismatching captions: $\{(x_1, t_1, \hat{t}_1), \cdots, (x_m, t_m, \hat{t}_m)\}$

Encode m matching captions $\{h_i | h_i = \varphi(t_i), i = 1, \cdots, m\}$

Encode m mismatching captions $\{\hat{h}_i | \hat{h}_i = \varphi(\hat{t}_i), i = 1, \cdots, m\}$

Take a minibatch of m random numbers $\{\epsilon_1, \cdots, \epsilon_m\}$ sampled from the uniform distribution $U[0,1]$

Calculate m samples $\{\hat{x}_i | \hat{x}_i = \epsilon_i x_i + (1 - \epsilon_i) \cdot G(z_i, h_i), i = 1, \cdots, m\}$

Update D using the gradient:

$$\nabla \frac{1}{m} \sum_{i=1}^{m} \Big\{ D(G(z_i, h_i), h_i) + \alpha D\big(x_i, \hat{h}_i\big) - (1 + \alpha) D(x_i, h_i) + \lambda \left[\Big(\max\Big(0, \big\| \nabla_{\hat{x}_i, h_i} D(\hat{x}_i, h_i) \big\|_2 - 1 \Big) \Big)^2 \right] \Big\}$$

(for WGAN-GP it is:

$$\nabla \frac{1}{m} \sum_{i=1}^{m} \Big\{ D(G(z_i, h_i), h_i) + \alpha D\big(x_i, \hat{h}_i\big) - (1 + \alpha) D(x_i, h_i) + \lambda \left[\Big(\big\| \nabla_{\hat{x}_i, h_i} D(\hat{x}_i, h_i) \big\|_2 - 1 \Big)^2 \right] \Big\}$$

instead)

end for

Take two minibatch of m noise samples $\{z_1, \cdots, z_{2m}\}$ sampled from noise distribution P_z

Take another two minibatches of m random captions $\{t_{m+1}, \cdots, t_{3m}\}$

Encode these two minibatches of m captions $\{h_i | h_i = \varphi(t_i), i = m + 1, \cdots, 3m\}$

Interpolate these two minibatches of m captions encodings: $\{\tilde{h}_i | \tilde{h}_i = \beta h_{i+m} + (1 - \beta) h_{i+2m}, i = 1, \cdots, m\}$

Update G using the gradient:

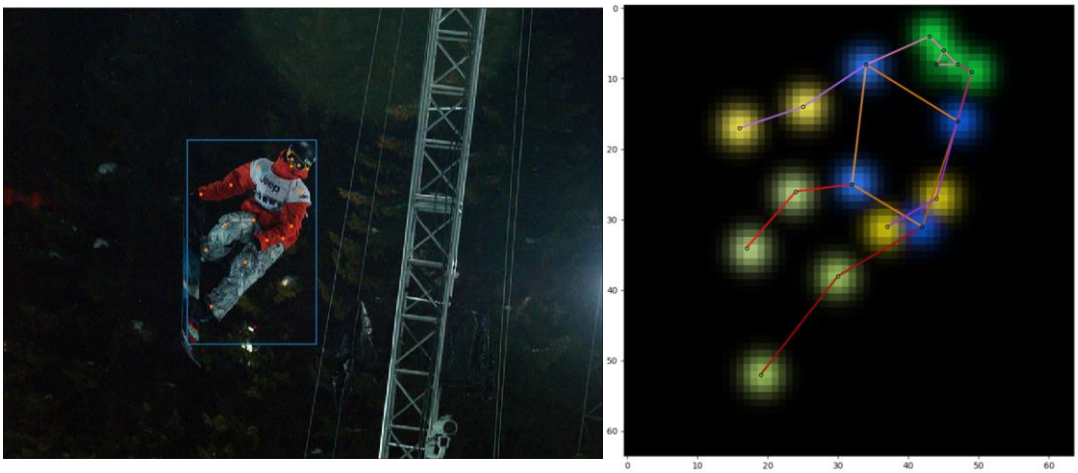
$$-\nabla \frac{1}{m} \sum_{i=1}^{m} \big[D(G(z_i, h_i), h_i) + D\big(G\big(z_{i+m}, \tilde{h}_i\big), \tilde{h}_i\big) \big]$$

end for

5. Implementation

5.1. Data Set

The data set that we are using for training and evaluating our model is the Microsoft COCO (Common Objects in Context) [22] (http://cocodataset.org/). This data set contains more than 100 thousand annotated images of everyday scenes. In the images, objects of different categories (person, bicycle, etc.) are labeled. A bounding box (for example, the blue rectangle surrounding the person shown in **Figure 8**) is given for each labeled object. If a labeled person is clear enough in the image, the locations of visible pose keypoints (for example, again in **Figure 8**, the orange dots on the person) are also given. There are in total 17 keypoints: they are the nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle and right ankle. In the example of **Figure 8**, all the 17 keypoints are visible but in most cases only some of them are visible due to occlusion. There are some poses of persons in the data set that have too few keypoints visible, and thus they do not provide much pose information for our models. So we will only include persons with 8 or more out of the 17 keypoints visible in the training of our models. In addition, there are five captions accompanying each image. For example, the five captions for **Figure 8** are shown below the image. We will use the keypoint information (poses) and captions from the COCO data set in our work.



Snowboarder in full orange and white gear is airborne.
A skateboarder negotiates a steep slope next to a crane.
A guy flies through the air on his snow board.
a snowboard high up in the air while doing a trick
A snowboarder in mid air heading toward the ground.

Figure 8 An example from the COCO data set. There is a person in this image on the upper left. The blue rectangle is the bounding box and the orange dots indicate where the keypoints are. Below are the five captions for this image. On the upper right is the heatmap of the pose of the person.

The poses will be processed as heatmaps and the captions will be encoded into vectors by the fastText encoder (**Section 4.1**).

The height and width of the images processed in our models are all 64 and there are 17 keypoints, so our heatmap size is $64 \times 64 \times 17$. However, the images from the COCO data set have varying sizes. We deal with this with the help of the bounding box: we perform an affine transformation on the keypoints of a person, such that the bounding box fits in the center of our $64 \times 64 \times 17$ heatmap. And we set the bell shape width $\sigma = 2$ when generating the heatmaps. In **Figure 8** on the upper right is an illustration of an example heatmap. Here, to display all the 17 channels in one single colored image, we use different colors to show different channels, and the brighter the colors are, the higher the values are. Adjacent keypoints can be connected to form a "skeleton" of this pose. The skeleton is also marked by different colors.

5.2. Model Architecture

5.2.1. Unconditional Model

The unconditional model is just a plain unconditional DCGAN, with the size of image $64 \times 64 \times 17$. The input of the generator G is a noise $z \in \mathbb{R}^{128}$. There are 5 transposed convolution layers. In each layer, a transposed convolution is followed by batch normalization and then ReLU activation, except for the last layer. In the last layer there is no batch normalization, and the activation is a $tanh$ function. **Table 1** is a summary of G's architecture. The output of G is a heatmap $x \in [-1,1]^{64\times64\times17}$, and we need to scale it to $[0,1]$ to obtain the normal heatmap.

layer	input	transposed convolution	batch normali-zation	activation	output
1	$1 \times 1 \times 128$	128,512,4,1,0	yes	ReLU	$4 \times 4 \times 512$
2	$4 \times 4 \times 512$	512,256,4,2,1	yes	ReLU	$8 \times 8 \times 256$
3	$8 \times 8 \times 256$	256,128,4,2,1	yes	ReLU	$16 \times 16 \times 128$
4	$16 \times 16 \times 128$	128,64,4,2,1	yes	ReLU	$32 \times 32 \times 64$
5	$32 \times 32 \times 64$	64,17,4,2,1	no	tanh	$64 \times 64 \times 17$

Table 1 The architecture of the unconditional model's generator. In the column "transposed convolution", the parameters are input channels, output channels, kernel size, stride, padding.

We add one extra convolution layer to our discriminator (compared to the DCGAN of Radford et al. [2]) so as to make it easier to incorporate caption encodings later on in the conditional model. The input of the discriminator D is a heatmap $x \in [-1,1]^{64\times64\times17}$, so again we need to scale the heatmap before inputting it to D. However, when we directly input an output from G to D, the scaling is not necessary on either side. There are 6 convolution layers. In each layer, a convolution is followed by batch normalization and then leaky ReLU activation, except for the last layer, where the activation is a sigmoid function ($y = \frac{1}{1+e^{-x}}$). **Table 2** is a summary of D's architecture. The output of D is a scalar $D(x) \in [0,1]$. So we have for the common GAN $G: \mathbb{R}^{128} \to [-1,1]^{64\times64\times17}$ and $D: [-1,1]^{64\times64\times17} \to [0,1]$.

layer	input	convolution	batch normali-zation	activation	output
1	$64 \times 64 \times 17$	17,64,4,2,1	yes	leaky ReLU	$32 \times 32 \times 64$
2	$32 \times 32 \times 64$	64,128,4,2,1	yes	leaky ReLU	$16 \times 16 \times 128$
3	$16 \times 16 \times 128$	128,256,4,2,1	yes	leaky ReLU	$8 \times 8 \times 256$
4	$8 \times 8 \times 256$	256,512,4,2,1	yes	leaky ReLU	$4 \times 4 \times 512$
5	$4 \times 4 \times 512$	512,512,1,1,0	yes	leaky ReLU	$4 \times 4 \times 512$
6	$4 \times 4 \times 512$	512,1,4,1,0	no	sigmoid	$1 \times 1 \times 1$

Table 2 The architecture of the unconditional model's discriminator. In the column "convolution", the parameters are input channels, output channels, kernel size, stride, padding. All leaky ReLUs have the negative slope $\alpha = 0.2$.

The above model architecture is only suitable for the common GAN algorithm (**Algorithm 4**). When applying the WGAN algorithm (**Algorithm 5**), we should remove the sigmoid activation in the last layer of D and its output $D(x) \in \mathbb{R}$ is no longer bounded in $[0,1]$. If we further want to apply the WGAN-GP algorithm (**Algorithm 6**), we should also remove all the batch normalization in D. The architecture of G can remain the same. So we have for the WGAN $G: \mathbb{R}^{128} \to [-1,1]^{64\times64\times17}$ and $D: [-1,1]^{64\times64\times17} \to \mathbb{R}$. **Figure 9** is a diagram of the architecture.

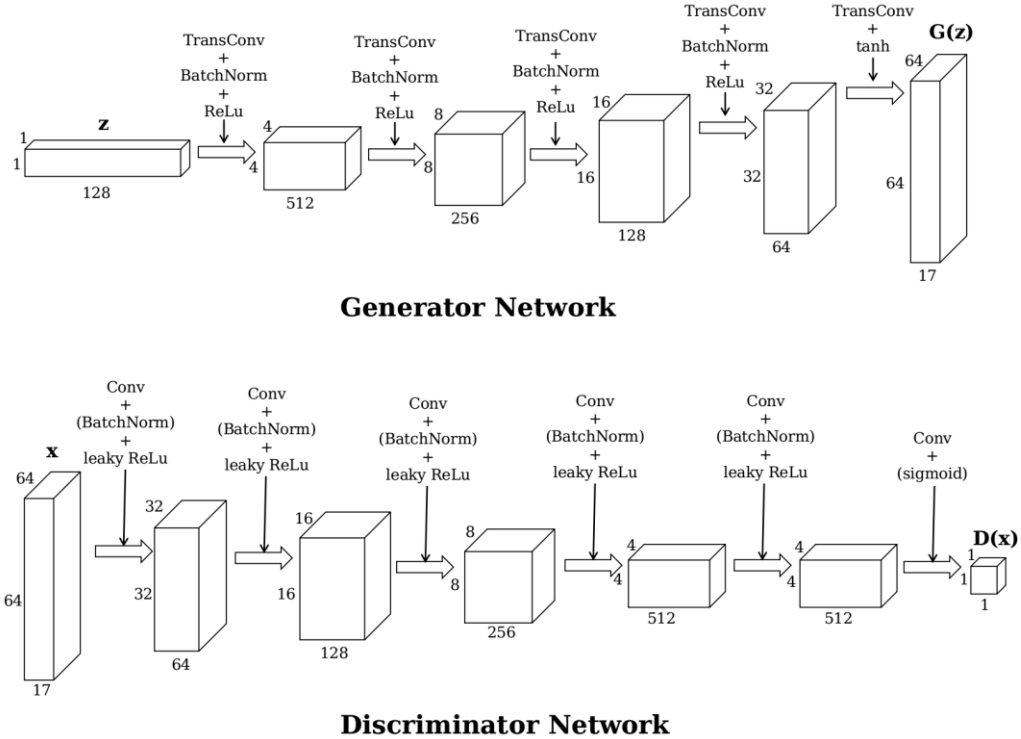


Figure 9 Architecture of the unconditional GAN.

5.2.2. Conditional Model

The conditional model is the tool to synthesize human poses from captions. We adopt the conditional model presented by Reed et al. [3] (with some small changes), which is based on DCGAN [2]. The conditional model is the unconditional model with the condition (caption

encoding) added. Because the number of training samples with captions is much smaller than the number of training samples without captions (details in **Section 6.1 and 6.2**), in order to avoid overfitting (the networks' capacity being too large for the number of training data), we will reduce the capacity of the networks by reducing the number of convolutional features in each convolution or transposed convolution layer in the G and D networks by half.

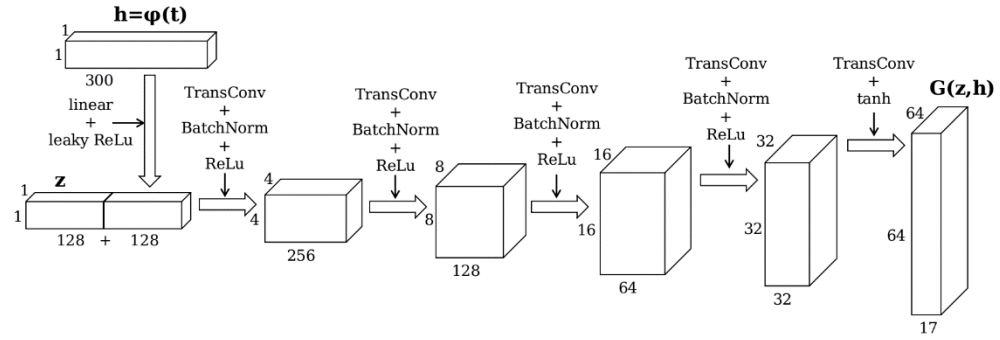
The condition is the encoding of the caption $h \in \mathbb{R}^{300}$. In the generator G, the encoding is compressed to 128 dimensions by a linear transformation followed by leaky ReLU activation. The compressed encoding is then concatenated to the input noise $z \in \mathbb{R}^{128}$ to get a 256-dimension input to the G. The other parts of G remain the same as in the unconditional model, except for the reduced number of convolutional features. In the discriminator D, the encoding $h \in \mathbb{R}^{300}$ is also compressed to 128 dimensions by another linear transformation followed by leaky ReLU activation. The first four layers remain the same, except for the reduced number of convolutional features. After the fourth layer, the compressed encoding is replicated sixteen times and then concatenated to the layer's $4 \times 4 \times 256$ output along the feature dimension, so the output becomes $4 \times 4 \times 384$. And then the last two layers continue. As we will use the WGAN-GP and WGAN-LP algorithm, there is no batch normalization in the discriminator D. **Table 3** and **Table 4** are the summaries of the architectures of the G and D, respectively, and **Figure 10** is the diagram. So we have for the conditional WGAN $G: \mathbb{R}^{128} \times \mathbb{R}^{300} \to [-1,1]^{64\times64\times17}$ and $D: [-1,1]^{64\times64\times17} \times \mathbb{R}^{300} \to \mathbb{R}$,

layer	input	transposed convolution	batch normalization	activation	output
	300	linear	no	leaky ReLU	128
1	$1 \times 1 \times 256$	256,256,4,1,0	yes	ReLU	$4 \times 4 \times 256$
2	$4 \times 4 \times 256$	256,138,4,2,1	yes	ReLU	$8 \times 8 \times 128$
3	$8 \times 8 \times 128$	128,64,4,2,1	yes	ReLU	$16 \times 16 \times 64$
4	$16 \times 16 \times 64$	64,32,4,2,1	yes	ReLU	$32 \times 32 \times 32$
5	$32 \times 32 \times 32$	32,17,4,2,1	no	tanh	$64 \times 64 \times 17$

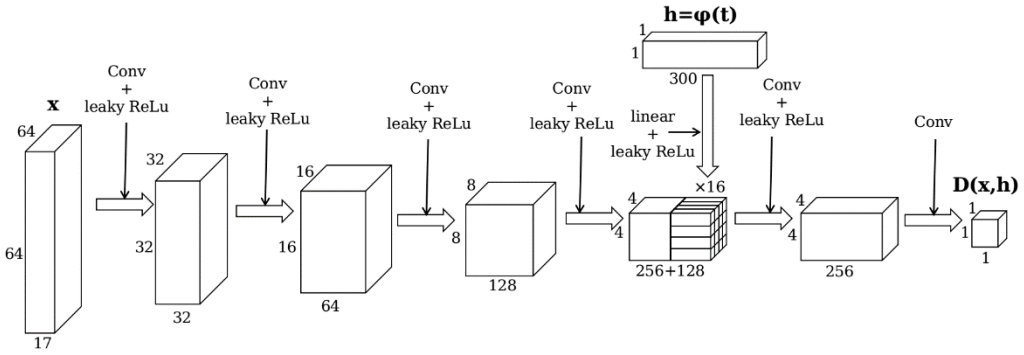
Table 3 The architecture of the conditional model's generator. There is a linear transformation to compress the caption encoding (the first row). In the column "transposed convolution", the parameters are input channels, output channels, kernel size, stride, padding. The ReLU has the negative slope $\alpha = 0.2$.

layer	input	convolution	activation	output
1	$64 \times 64 \times 17$	17,32,4,2,1	leaky ReLU	$32 \times 32 \times 32$
2	$32 \times 32 \times 32$	32,64,4,2,1	leaky ReLU	$16 \times 16 \times 64$
3	$16 \times 16 \times 64$	64,128,4,2,1	leaky ReLU	$8 \times 8 \times 128$
4	$8 \times 8 \times 128$	128,256,4,2,1	leaky ReLU	$4 \times 4 \times 256$
	300	linear	leaky ReLU	128
5	$4 \times 4 \times 384$	385,256,1,1,0	leaky ReLU	$4 \times 4 \times 256$
6	$4 \times 4 \times 256$	256,1,4,1,0	no	$1 \times 1 \times 1$

Table 4 The architecture of the conditional model's discriminator. There is a linear transformation to compress the caption encoding (the fifth row). There is no batch normalization as we are using the WGAN-LP/-GP algorithm. In the column "convolution", the parameters are input channels, output channels, kernel size, stride, padding. All leaky ReLUs have the negative slope $\alpha = 0.2$.



Conditional Generator Network



Conditional Discriminator Network

Figure 10 Architecture of the conditional GAN.

6. Experiments

6.1. Unconditional Model

For our unconditional model, we use all the eligible human poses (≥ 8 visible keypoints) in the COCO data set. Some images contain more than one eligible human pose. There are in total 116021 poses in the training set. The COCO data set also provides an official separate validation set. There are 4812 poses in the validation set. During the training, we use the validation poses mainly to calculate the D loss and G loss. After training, the validation set is used for evaluation of the model.

In the experiments, we will try training the unconditional GAN with **Algorithm 4** the common GAN algorithm (**Section 7.1.1**), **Algorithm 5** the WGAN algorithm with parameter clamping (**Section 7.1.2**) and **Algorithm 6** the WGAN-GP algorithm (**Section 7.1.3**).

6.2. Conditional Model

For the conditional model, we first need to obtain pose-caption pairs from the data set for training. But in the COCO data set, in images with more than one person, it is quite a difficult task to decide which part of the caption is describing which person. We restrict our task to images containing a single person: we only use the eligible human poses (≥ 8 visible keypoints) in images containing one person in the COCO data set. Now the numbers of poses in the training set and the validation set decrease to 17326 and 714, respectively. During the training, each time a pose is taken from the data set, a caption is selected randomly from the five corresponding captions to pair with the pose. On the other hand, mismatching captions are selected randomly from all captions in the training data set.

Considering the drastic decrease in training samples, we fear that the training set is not large enough for the model to learn to synthesize poses as well as in the unconditional scenario, where we train the model without captions and we simply take all eligible poses, no matter how many persons there are in one image. We come up with a strategy. We will initialize the conditional model's network parameters in the following way. As the conditional model is just a lightly modified version of the unconditional model with reduced capacity, we first train the corresponding unconditional model using the WGAN-GP algorithm on all training poses (larger number), where in images with more than one person, we take the bounding boxes. Then in the conditional model, for the parameters that already appear in the unconditional model, we assign them the trained values at initialization; and for all the other parameters, we randomly initialize them. After this initialization, we start to train the conditional model on training poses with captions (smaller number). In this way, we believe that our conditional model can synthesize realistic human poses from the beginning of training, and later training is just to adjust the poses to match the captions.

One important issue to note is that, the magnitude of a caption vector h encoded by the fastText model is much smaller than the magnitude of G's noise input z. So before being fed into the networks G and D, the caption vector h is multiplied by 30, otherwise the effect of the caption encoding h will be submerged by the noise z or other convolutional features in the networks.

In the experiments, we will pretrain an unconditional GAN with WGAN-GP for the initialization (**Section 7.2.1**) and try training the conditional GAN with **Algorithm 7** the WGAN-LP (**Section 7.2.2**) and WGAN-GP (**Section 7.2.4**) algorithms. It is worthwhile to compare the model's performance with these two algorithms. Why is this important? In **Section 4.3**, we say that for the conditional model the WGAN-LP algorithm will work better than the WGAN-GP algorithm. However, whether this is really the case can only be tested through real experiments.

In addition to that, we will do other two trials: training the conditional GAN without pretraining an unconditional model (**Section 7.2.5**) and training the conditional GAN with multi-person poses (**Section 7.2.6**). First, we would like to see how our model preforms when we train it with all parameters randomly initialized, without using the unconditional model's training result. Second, we would like to try and see whether our model can still work when the training samples are not limited to single-person images, because synthesizing multi-person poses will be a very interesting piece of future work following this project.

For multi-person poses, we convert images from the data set with at least one eligible human pose into pose heatmaps of size $64 \times 64 \times 17$ like **Figure 11**. This time we have 49431 samples in the training set and 2038 in the validation set.

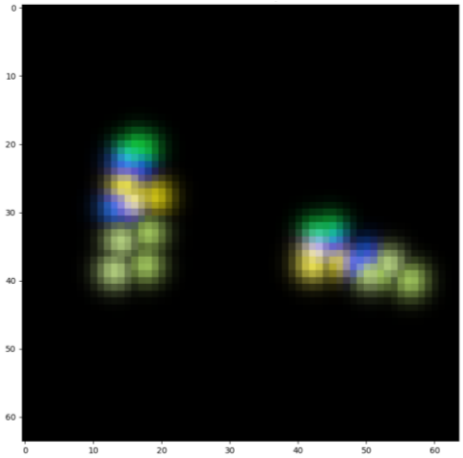


Figure 11 The pose heatmap of an image containing two persons.

We will also evaluate our conditional model (trained with WGAN-LP) with the interpolation test (**Section 7.2.7.1**), quantitative measures (**Section 7.2.7.2**) and a subjective test (**Section 7.2.7.3**).

7. Results

7.1. Unconditional Model

7.1.1. GAN

We first implement **Algorithm 4** – the common GAN algorithm with $k = 1$, i.e. updating the discriminator for one iteration and then updating the generator for one iteration. All the weights of the convolutions and transposed convolutions are initialized randomly with a normal distribution with 0 mean and 0.02 standard deviation $N(0, 0.02^2)$; for batch normalization the weights are initialized randomly with a normal distribution with 1 mean and 0.02 standard deviation $N(1, 0.02^2)$, the biases initialized with a normal distribution with 0 mean and 0.02 standard deviation $N(0, 0.02^2)$. The input noises of G are sampled from the standard normal distribution $N(0,1)$. Both D and G are trained with the Adam optimizer with first momentum smoothing constant $\beta_1 = 0.5$, second momentum smoothing constant $\beta_2 = 0.999$ and learning rate 0.0005. We use minibatch size $m = 128$ and train them for 50 epochs (going through all 116021 poses 50 times).

In **Figure 12** we plot the loss curves of the discriminator and generator. Training losses are calculated after each iteration, while validation losses are only calculated after each epoch. Therefore, data points for validation losses are much sparser than those for training losses. The discriminator loss is very close to zero while the generator loss is not. From **Expression (1)** we know that this means that the discriminator is learning better than the generator because it can correctly discriminate the fake samples. Other than that, as mentioned before, the GAN's loss curves cannot give much information about the training status and synthesized sample quality. So we might as well look directly into some heatmaps outputted by the generator (**Figure 13**), with input noises randomly sampled from $N(0,1)$.

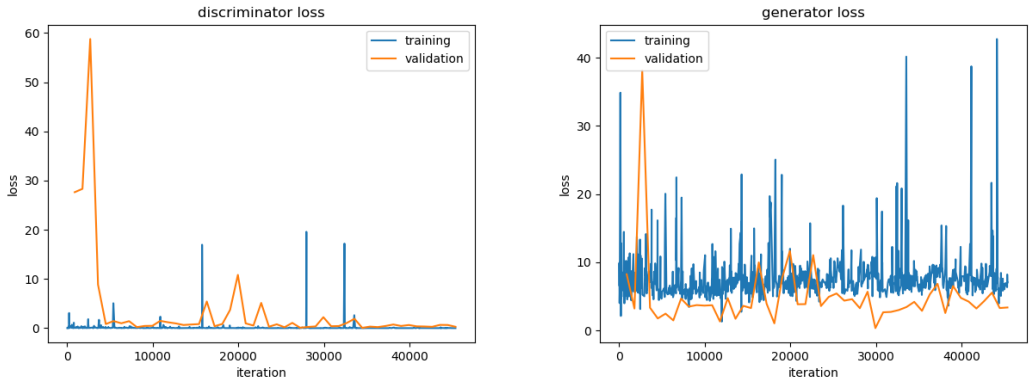


Figure 12 Loss curves of the unconditional model with common GAN algorithm.

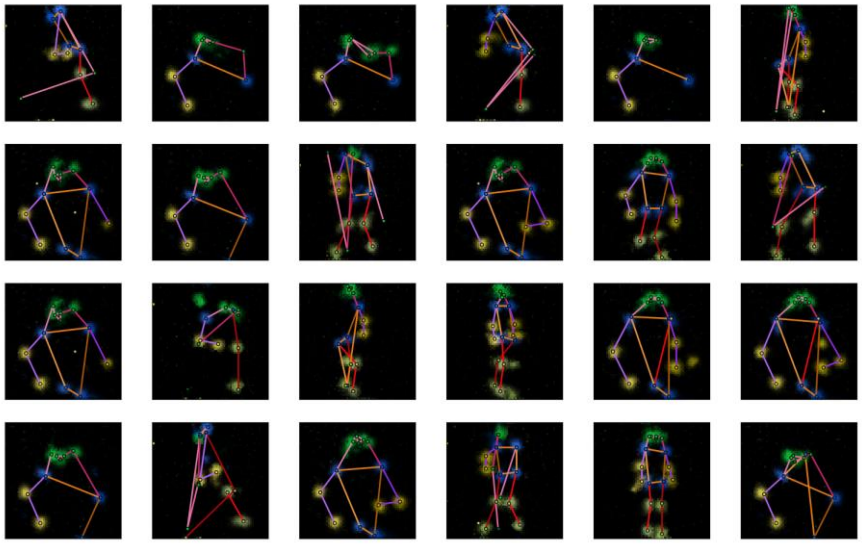


Figure 13 Some sample outputs of the generator of the unconditional model trained with common GAN algorithm.

First, the generator learns to generate human pose, as we can see that in most of the heamaps in **Figure 13** the combinations of the keypoints look like valid human poses. Second, the outputs have certain diversity. There are different poses, including full-body poses as well as half-body poses. Third, the mode collapse problem is obvious: many samples are very similar. And fourth, some poses are still not very realistic.

7.1.2. WGAN

We then try implementing **Algorithm 5** – the WGAN algorithm, with $k = 5$ and $c = 0.01$, i.e. updating the discriminator for five iterations and then updating the generator for one iteration and clamping D's parameters to $[-0.01, 0.01]$. We switch to the RMSProp optimizer with smoothing constant $\alpha = 0.99$ and learning rate 0.0001 for training both D and G. Because G is updated only every five iterations, we increase the number of training epochs to 200. The network parameters are randomly initialized as before and the minibatch size $m = 128$. **Figure 14** is the loss curves of the discriminator and generator. Data points for generator's training loss are sparser than those for the discriminator's, as the generator is not updated in every iteration. **Figure 15** shows some sample outputs from the generator. In theory [16], the absolute value of the discriminator loss should be related with the generator outputs' quality (the smaller, the better), but in our experiment, the discriminator loss does not change much. However, unexpectedly, our generator outputs do not look very bad. One obvious improvement compared to the last experiment is that the mode collapse seems to be relieved. At least there are no repeated patterns in these 24 samples. But the proportion of 'failed' outputs increases. This may be because of the inherent drawback of the parameter clamping in **Algorithm 5**: the learned distribution is a too simple approximation to the real data distribution. And such drawback can be overcome by **Algorithm 6** (WGAN-GP) [10].

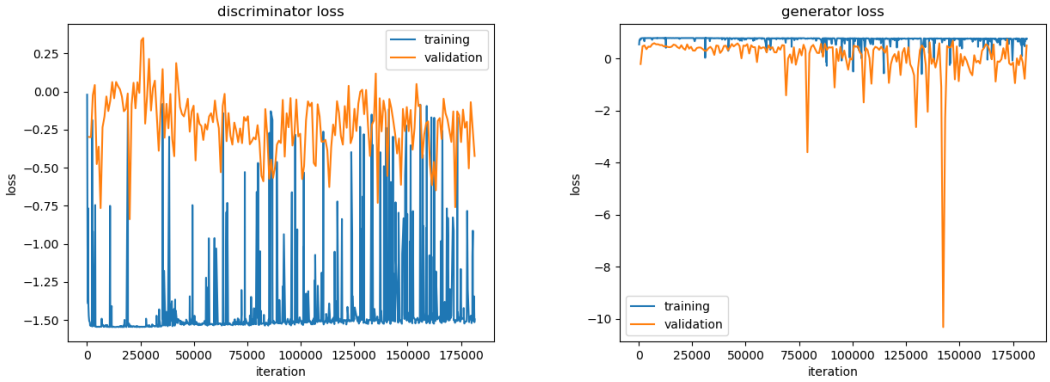


Figure 14 Loss curves of the unconditional model with WGAN algorithm.

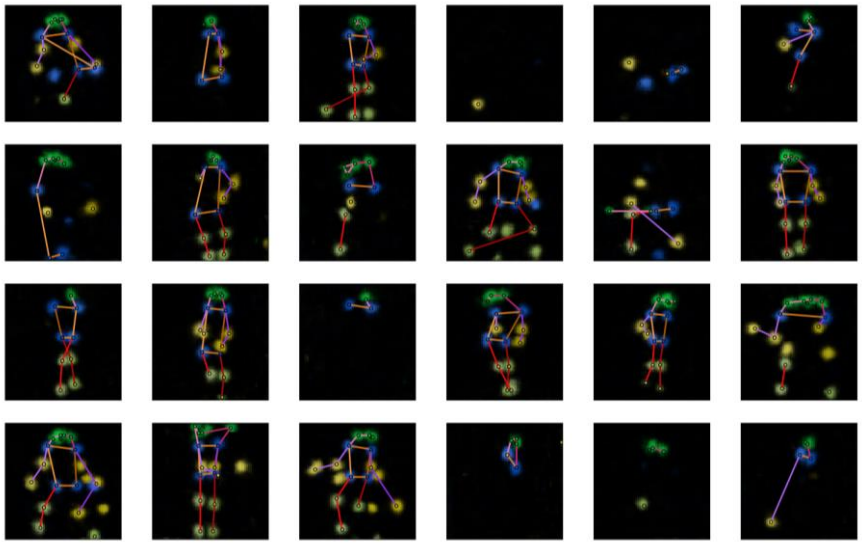


Figure 15 Some sample outputs of the generator of the unconditional model trained with WGAN algorithm.

7.1.3. WGAN-GP

We now train our model using **Algorithm 6** WGAN-GP and see whether can really improve the results. We set $k = 5$ and $\lambda = 10$, i.e. updating the discriminator for five iterations and then updating the generator for one iteration and penalizing the gradients with coefficient 10. The optimizer for both D and G is Adam with first momentum smoothing constant $\beta_1 = 0$, second momentum smoothing constant $\beta_2 = 0.9$ and learning rate 0.0001. The network parameters are randomly initialized as before and the minibatch size $m = 128$. During the training, we find that even after only 50 epochs the outputs are much better than in the previous experiment (after the same number of epochs) (**Figure 16**), which indicates that our model can learn faster with the WGAN-GP algorithm. **Figure 17** and **Figure 18** are the results after 200 epochs.

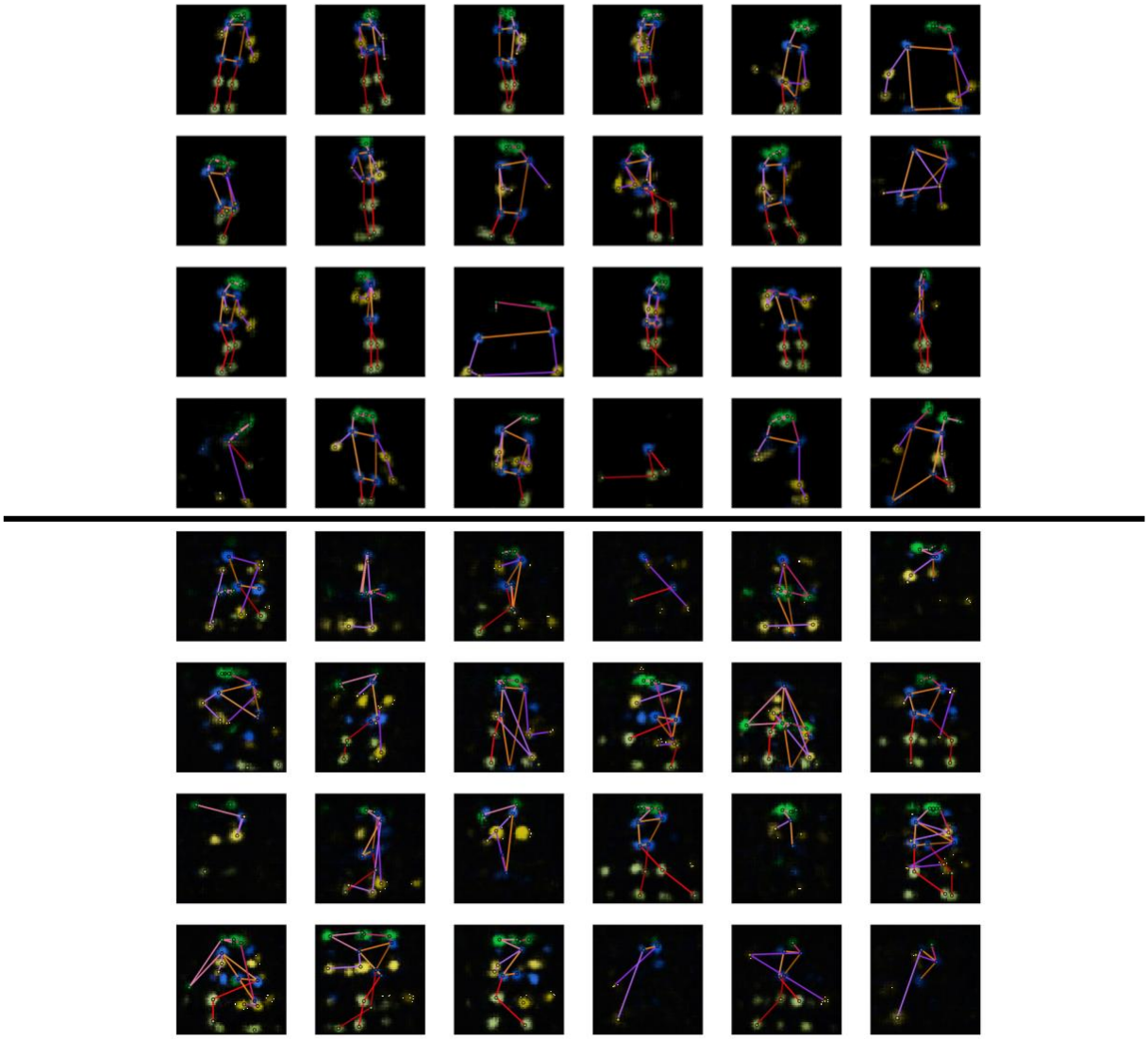


Figure 16 Some sample outputs of the generator of the unconditional model trained for 50 epochs with **Algorithm 6** WGAN-GP (upper) and **Algorithm 5** WGAN (lower).

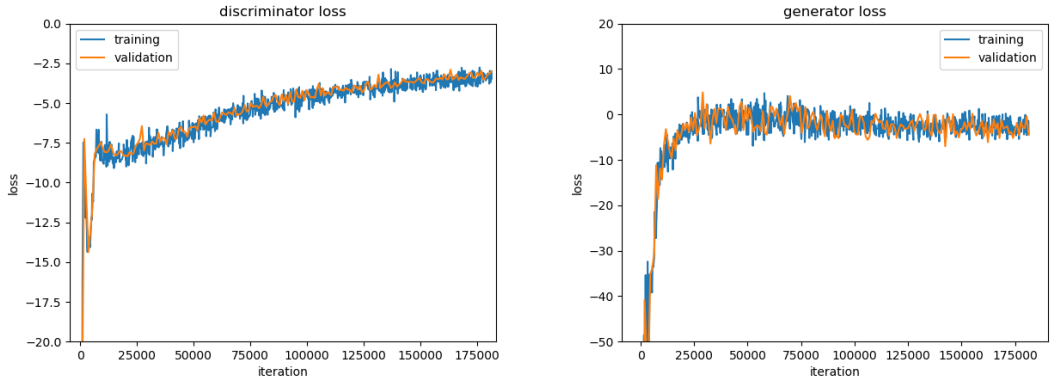


Figure 17 Loss curves of the unconditional model with WGAN-GP algorithm. Losses are very large in absolute values at the beginning of the training and the curves are cut off for visualization purposes.

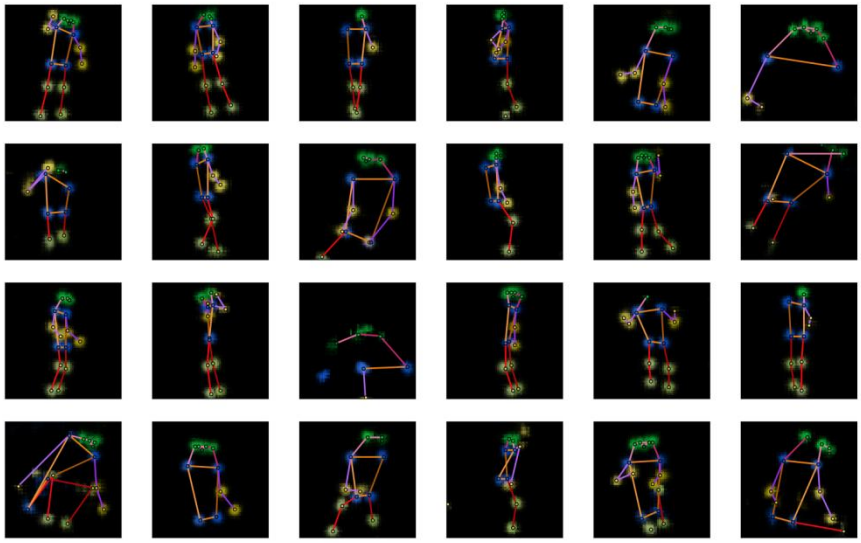


Figure 18 Some sample outputs of the generator of the unconditional model trained with WGAN-GP algorithm (after 200 training epochs).

From the results, we can see that all the sample outputs are realistic human poses. The previous problems (unrealistic poses and low diversity) are solved by the WGAN-GP algorithm. This is a good start for us, and we can go on to include captions in our model and synthesize poses from captions.

7.2. Conditional Model

7.2.1. Pretraining Unconditional Model

First, we train an unconditional model with reduced capacity for the initialization of the conditional model. **Figure 19** and **Figure 20** are the training results of the unconditional model for parameter initialization, using the same training hyperparameters as in **Section 7.1.3** for the WGAN-GP algorithm ($k = 5, \lambda = 10$, Adam optimizer with $\beta_1 = 0, \beta_2 = 0.9$ and learning rate 0.0001, minibatch size $m = 128$, 200 epochs, network parameters randomly initialized). The reduced model's sample outputs seem to be worse than the full unconditional model (**Figure 18**), maybe because the network capacity is reduced, but it is OK since we only use it to initialize our conditional model.

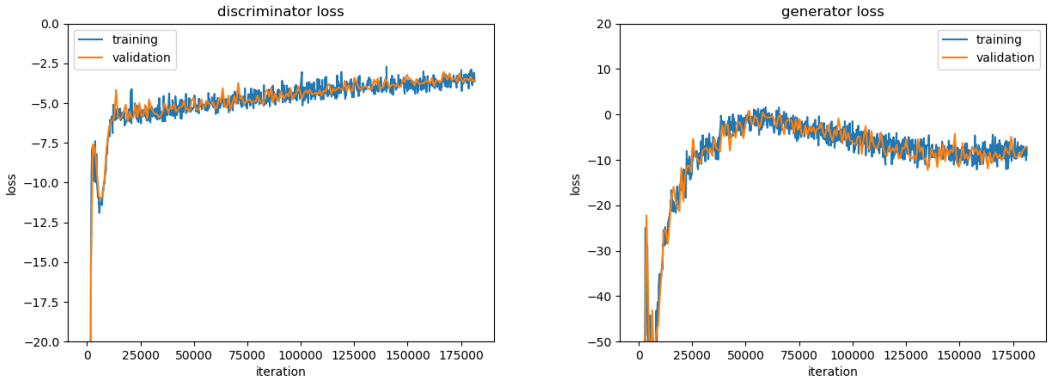


Figure 19 Loss curves of the reduced unconditional model with WGAN-GP algorithm. This model is trained for the initialization of the conditional model. Losses are very large in absolute values at the beginning of the training and the curves are cut off for visualization purposes.

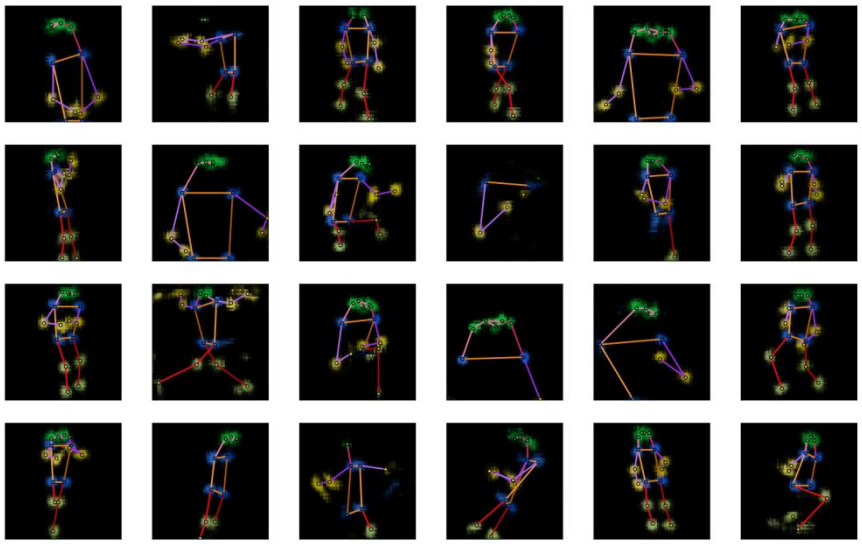


Figure 20 Some sample outputs of the generator of the reduced unconditional model trained with WGAN-GP algorithm. This model is trained for the initialization of the conditional model.

7.2.2. WGAN-LP

We set the hyperparameters $k = 5, \alpha = 1, \lambda = 150, \beta = 0.5$ (k: iterations of updating D before updating G, α: image-text matching coefficient, λ: penalty coefficient, β: encoding interpolation coefficient) in **Algorithm 7** WGAN-LP. The network parameters are initialized using the unconditional model's parameters, as described in **Section 6.2**. For the parameters of the linear transformations used for encoding compression, the weights are initialized randomly with a normal distribution with 0 mean and $\sqrt{\frac{2}{1+0.2^2}}$ standard deviation $N\left(0, \frac{\frac{2}{1+0.2^2}}{300}\right)$ (here we adopt the initialization method recommended by He et al. [32]), the biases initialized with a normal distribution with 0 mean and 0.02 standard deviation $N(0, 0.02^2)$. The input noises of G are again sampled from the standard normal distribution $N(0,1)$. Both D and G are trained with Adam optimizer with first momentum smoothing constant $\beta_1 = 0$, second momentum smoothing constant

$\beta_2 = 0.9$ and learning rate 0.0004. We use minibatch size of $m = 128$ and train the networks for 1000 epochs. The number of epochs is increased because the number of iterations in one epoch is reduced due to fewer training data. **Figure 21** is the loss curves. The discriminator loss gradually gets closer to zero, meaning that the model is learning better and better. **Figure 22** shows some synthesized poses. The captions used here are randomly selected from the validation set, which means that the model has never seen these captions in the training process.

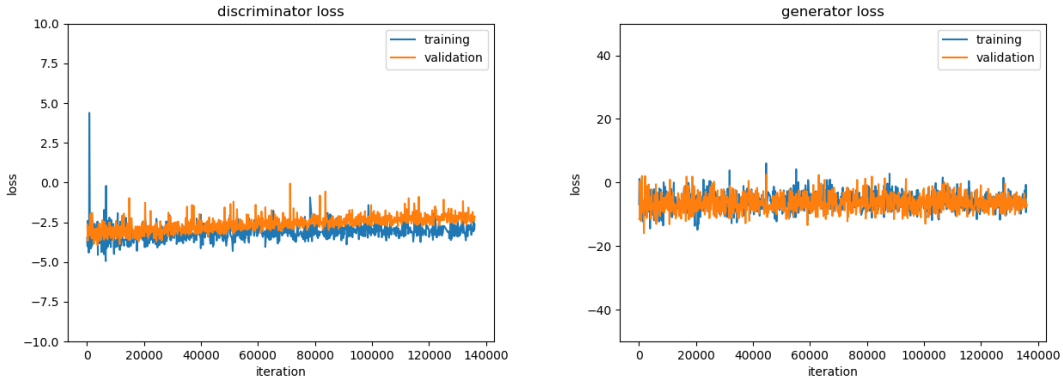


Figure 21 Loss curves of the conditional model with WGAN-LP algorithm.

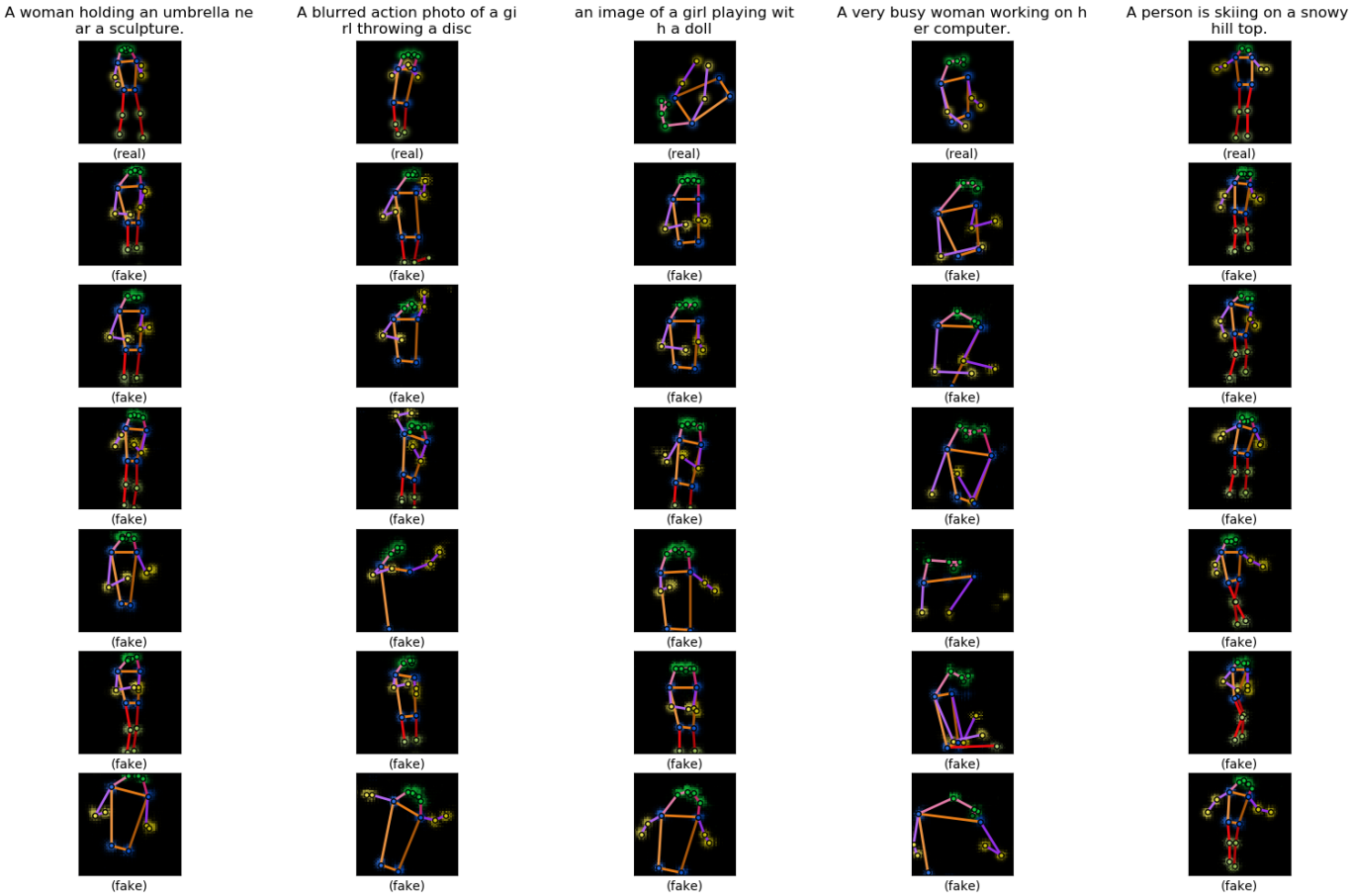


Figure 22 Some sample outputs of the generator of the conditional model trained with WGAN-LP algorithm. The first row is five sample poses from the validation set. The text on the top is the sample poses' accompanying captions. The six heatmaps below each sample pose are synthesized by the model from the caption on the top. Within the same row, the noise inputs to the generator are the same.

7.2.3. Analysis of the Training Results

On the one hand, in **Figure 22**, when we look along the row, we can see that although the noise to the generator z within a row is fixed, the generated poses are rather different. Therefore, the caption encodings are indeed effectively guiding the synthesis. On the other hand, when we look along the column, we can see that though quite diverse, the fake poses within a column share some common 'theme', and more importantly, it looks as if they resemble the real pose above and they can somewhat reflect the given caption. In other word, semantic consistency between text and poses is achieved in our model to some degree.

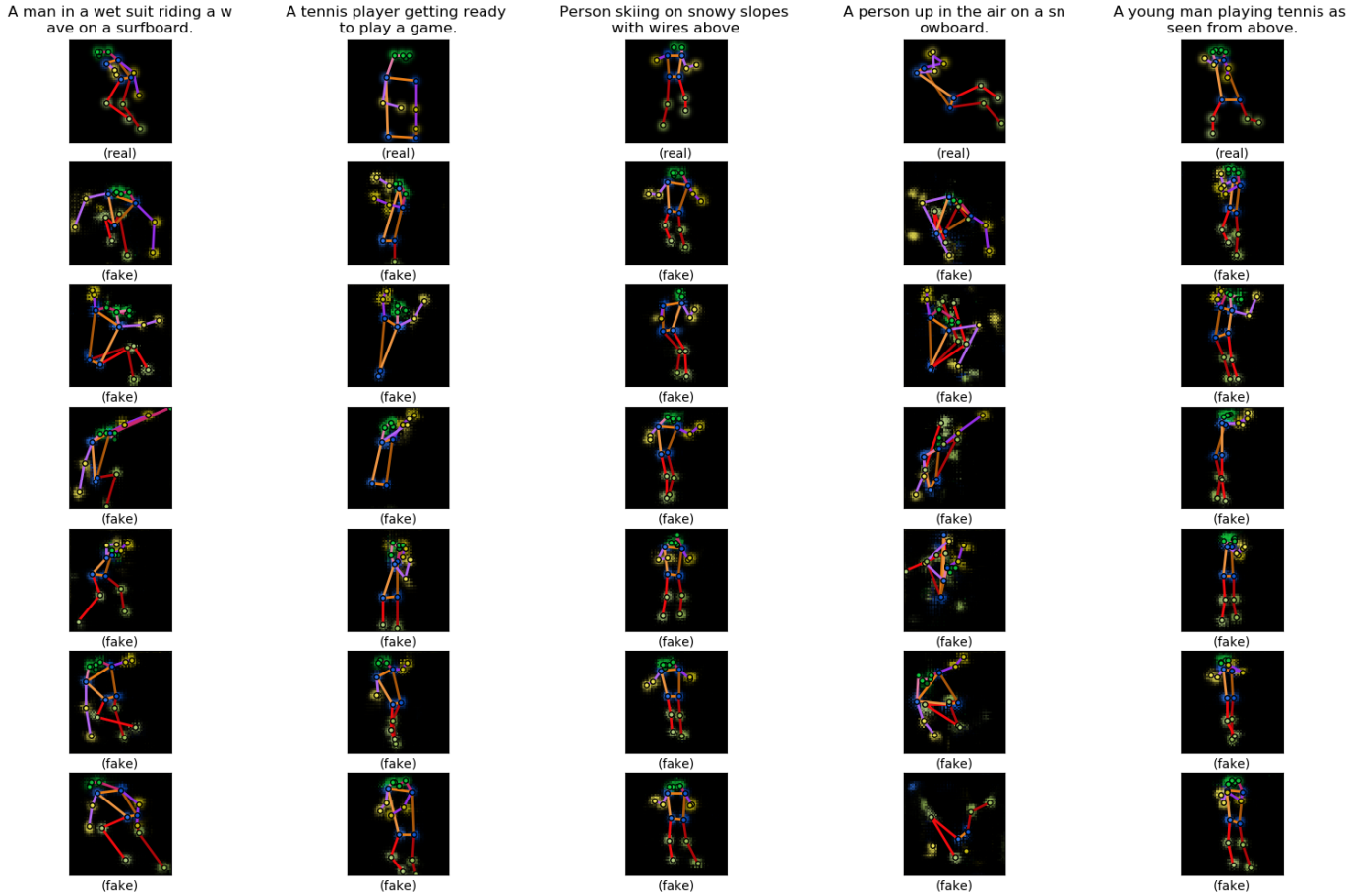


Figure 23 More sample outputs of the generator of the conditional model trained with WGAN-LP algorithm. The first row is five sample poses from the validation set. The text on the top is the sample poses' accompanying captions. The six heatmaps below each sample pose are synthesized by the model from the caption on the top. Within the same row, the noise inputs to the generator are the same.

The results are promising. Of course, we still have some aspects to improve. First, the synthesized poses can match the captions in terms of category but sometimes details may be missed. For example, in the second column counted from left in **Figure 23** the synthesized poses look like related to tennis, but they are not 'getting ready' but 'already playing', like the rightmost column. Perhaps this is because our text encoder is based on words in the sentences but does not encode the order of words in a sentence, as introduced in **Section 3.5**, and therefore sentence vectors may not reflect the meanings of the whole sentences very faithfully. We would expect a whole sentence

encoding to reflect the sentence's meaning but maybe it just would not capture the fine-grained meanings of the words. Maybe we should try to combine both word encodings and whole-sentence encodings in order to capture everything in a sentence (both fine-grained and coarse-grained meanings). Second, some of the synthesized poses look wrong or even very strange, such as the second column counted from right in **Figure 23**. Besides the text encoder, this may also be partly due to the data set, because we can only extract 2D poses from it and sometimes the poses are not very straightforward because of the shooting angle or blocking objects or going out of the range of the image, which can add to the difficulty of the model's learning. Also, the captions in the COCO data set are not specially designed for human poses, and therefore the captions are not very specific in describing the human pose and rather focus on the whole scene of the image. What's more, the number of different types of activities is not so balanced (for example, among single person images there are more sports images than other activities).

As the captions from the COCO data set are not specifically designed for poses, to better demonstrate the model's capacity, we feed it with some simple captions (**Figure 24**). These captions are specially made up by us and describe activities of one person, so it is rather clear what the target poses would look like. For those captions, our model can perform very well.

Figure 24 Poses synthesized from some simple captions.

We are also interested in how the generated pose changes based on gender. So it would be a nice experiment to generate some pairs of poses for the same caption with opposite subject genders while keeping the noise fixed. **Figure 25** and **Figure 26** are two sets of experiment results. The generated poses show subtle difference between the two genders. But if we observe more closely, we can find that the boy/girl poses are slightly smaller than the man/woman poses, which reflects

the reality. This experiment shows that in our model, the subject of the caption does not matter very much, and what really matters is the action.

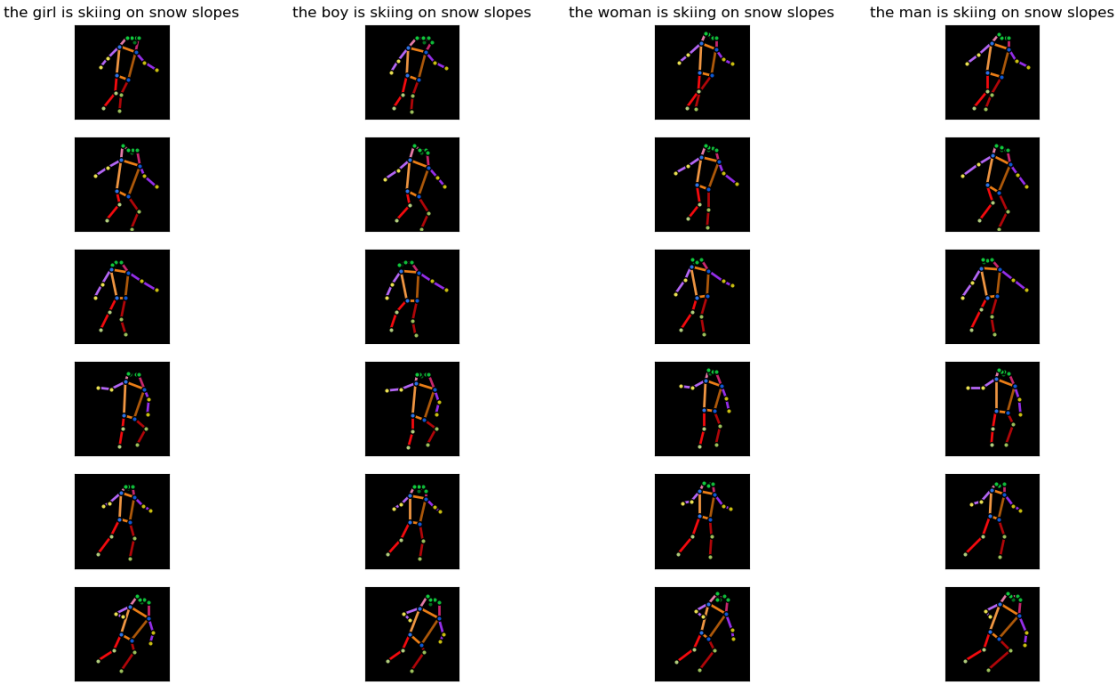


Figure 25 Poses synthesized from captions with opposite subject genders (ski). The caption to synthesize each column of poses is on the top. The noise input of the same row is the same.

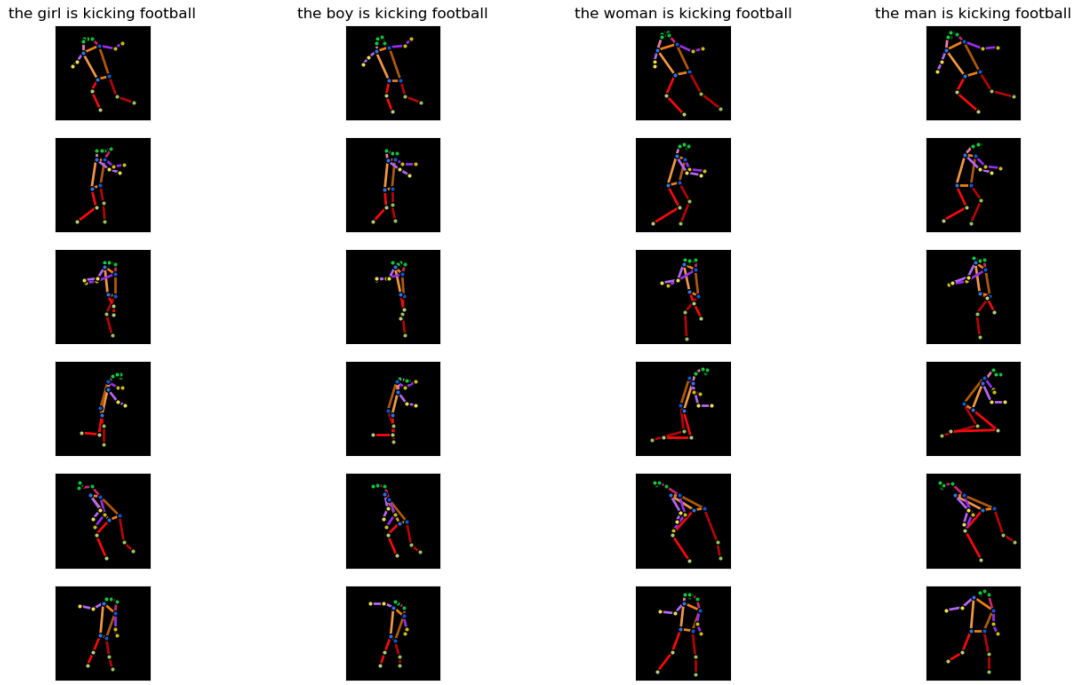


Figure 26 Poses synthesized from captions with opposite subject genders (football). The caption to synthesize each column of poses is on the top. The noise input of the same row is the same.

7.2.4. WGAN-GP

We train our conditional model with the conditional WGAN-GP algorithm (**Algorithm 7** in the parentheses). The only difference from the WGAN-LP is that the penalty is two-sided in D's loss function in the WGAN-GP (**Algorithm 7** in the parentheses). We train the model with $k = 5, \alpha = 1, \lambda = 20, \beta = 0.5$ (k: iterations of updating D before updating G, α: image-text matching coefficient, λ: penalty coefficient, β: encoding interpolation coefficient) with Adam optimizer with first momentum smoothing constant $\beta_1 = 0$, second momentum smoothing constant $\beta_2 = 0.9$ and learning rate 0.0004. The network parameters are initialized using the unconditional model's parameters, as before. We use minibatch size of $m = 128$ and train the networks for 1000 epochs. Actually, all the training hyperparameters are the same as WGAN-LP (**Section 7.2.2**), except for λ. **Figure 27** and **Figure 28** are the results. For comparison, the sample outputs of WGAN-LP and WGAN-GP are displayed side by side in **Figure 29**.

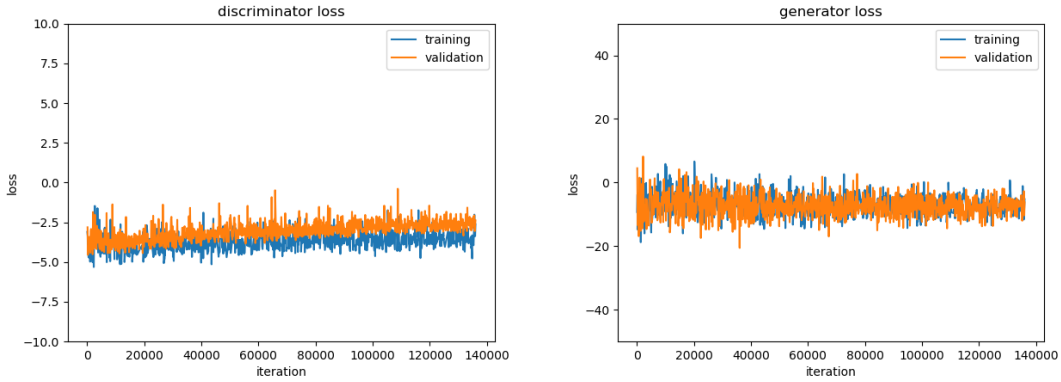


Figure 27 Loss curves of the conditional model with WGAN-GP algorithm.

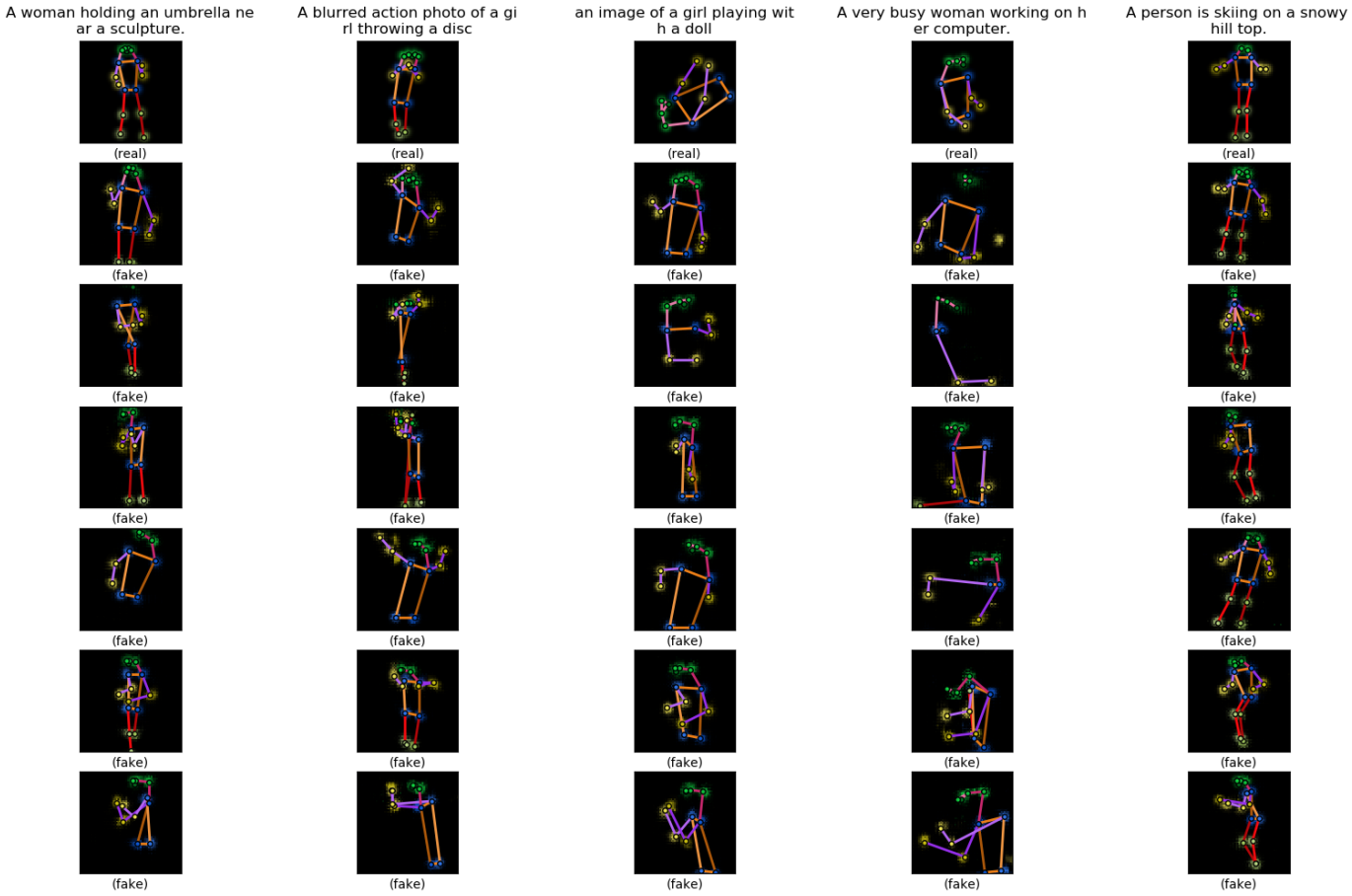


Figure 28 Some sample outputs of the generator of the conditional model trained with WGAN-GP algorithm. The first row is five sample poses from the validation set. The text on the top is the sample poses' accompanying captions. The six heatmaps below each sample pose are synthesized by the model from the caption on the top. Within the same row, the noise inputs to the generator are the same.

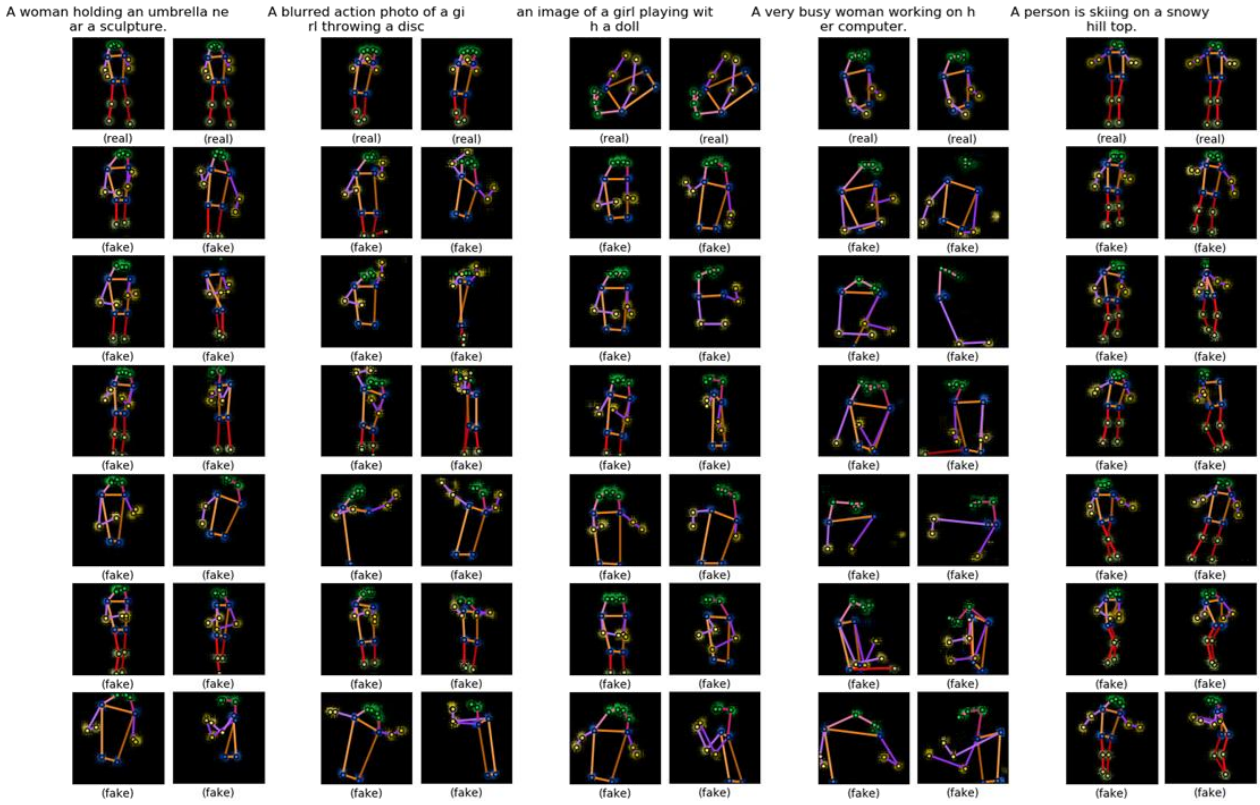


Figure 29 The heatmaps in **Figure 22** (WGAN-LP) and **Figure 28** (WGAN-GP) put together for comparison. The heatmaps in **Figure 28** are shown to the right of the heatmaps in **Figure 22**.

The loss curves are similar to that of the WGAN-LP (**Figure 21**). As for the sample outputs, visionally, it is hard to tell which algorithm is better. We can say that for our model, there is no significant improvement of the WGAN-LP over WGAN-GP, which is a little contrary to our expectation and to the statement of Petzka et al. [20].

7.2.5. No Pretraining Unconditional Model

We train the conditional model again using the WGAN-LP algorithm for 1000 epoch, all training hyperparameters kept the same as before ($k = 5, \alpha = 1, \lambda = 150, \beta = 0.5, \beta_1 = 0, \beta_2 = 0.9$, minibatch size $m = 128$ and learning rate 0.0004) but all network parameters are randomly initialized. **Figure 30** and **Figure 31** are the results. For comparison, the sample outputs of the specially initialized model (pretraining an unconditional model for initialization) and the randomly initialized model are displayed side by side in **Figure 32**. Except for the beginning part, the loss curves are almost the same as when the unconditional model's training result is used for initialization and the generator's outputs are not worse. This means that the model can learn as well without using the unconditional model's trained parameters. It seems that our fear before is unnecessary. Nonetheless, we will stick with the model trained with WGAN-LP and initialized with the unconditional model's trained parameters, as we think that that model makes more use of the information provided by the data set. The evaluations in **Section 7.2.7** are also all done upon that model.

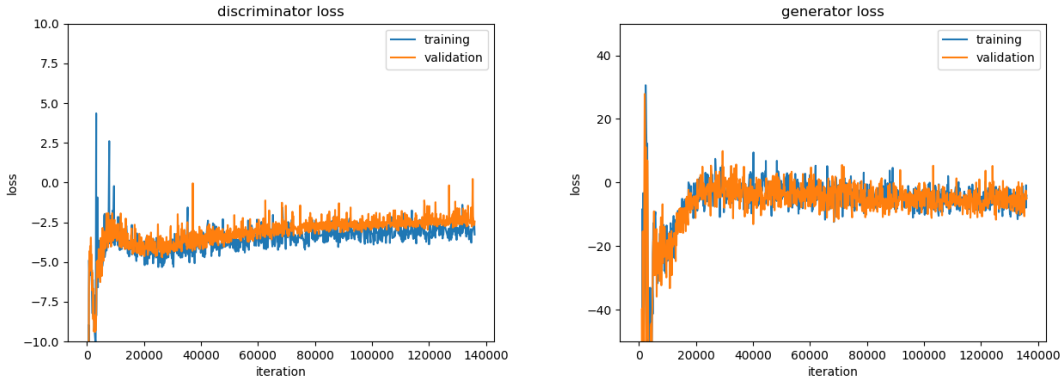


Figure 30 Loss curves of the conditional model with WGAN-LP algorithm. The network parameters are randomly initialized.

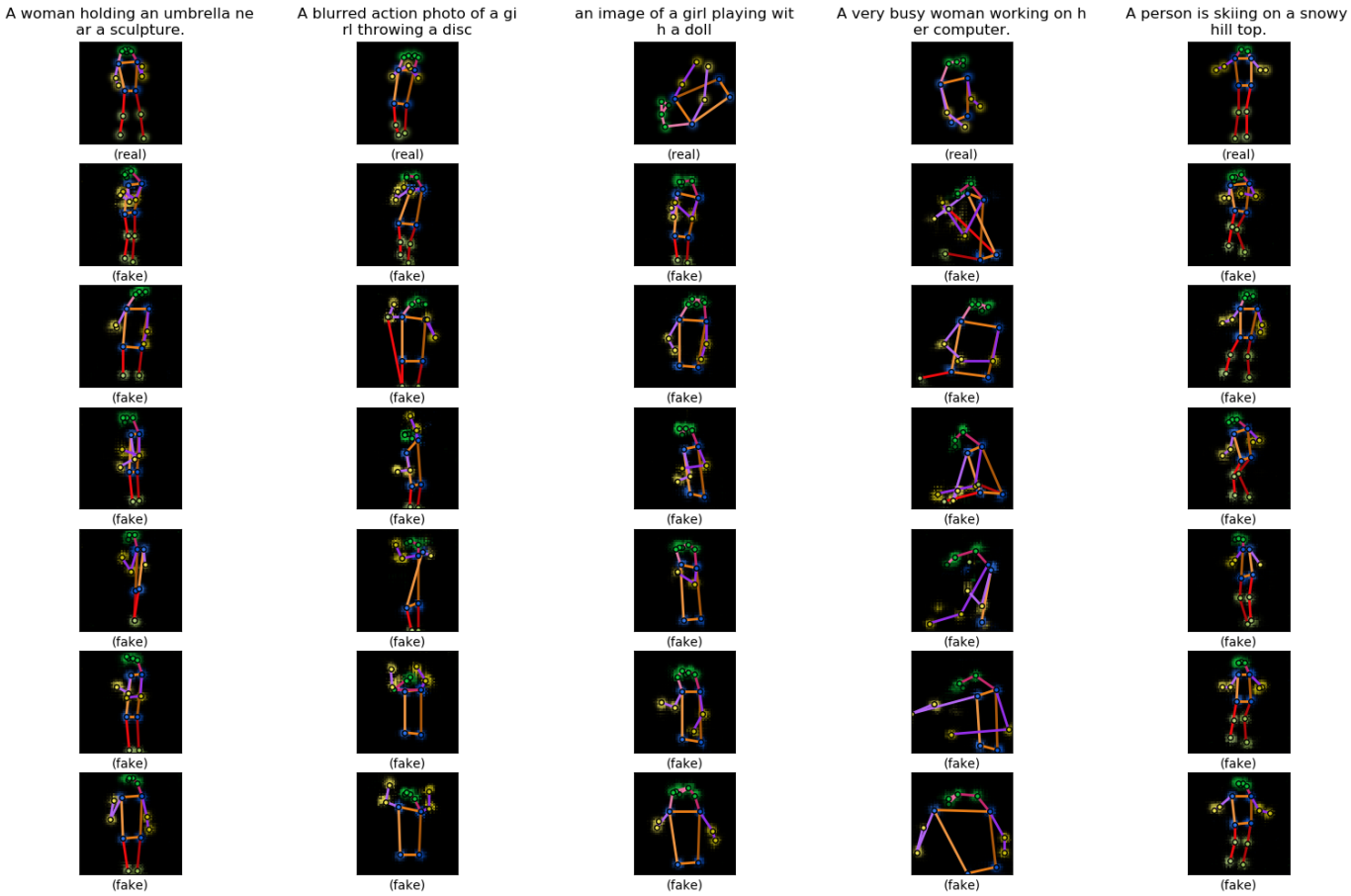


Figure 31 Some sample outputs of the generator of the conditional model trained with WGAN-LP algorithm. The network parameters are randomly initialized. The first row is five sample poses from the validation set. The text on the top is the sample poses' accompanying captions. The six heatmaps below each sample pose are synthesized by the model from the caption on the top. Within the same row, the noise inputs to the generator are the same.

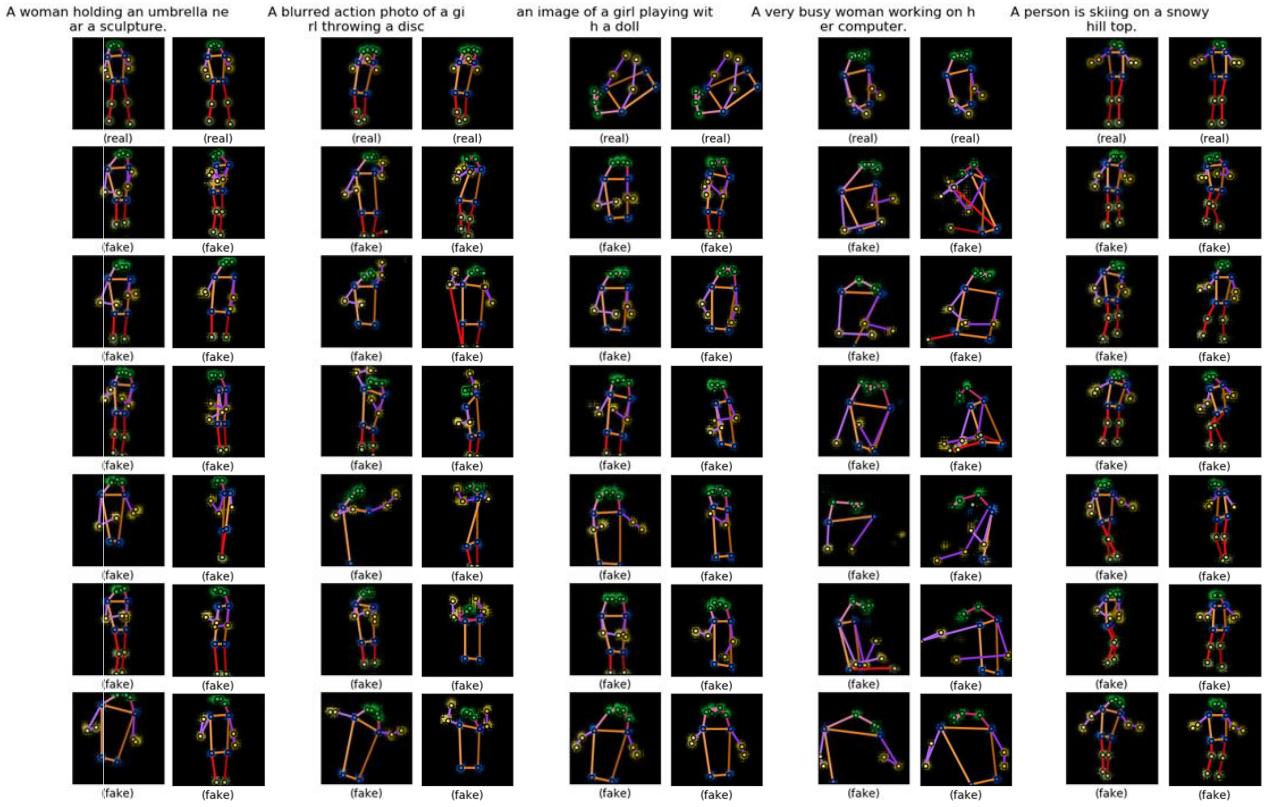


Figure 32 The heatmaps in **Figure 22** (specially initialized model) and **Figure 31** (randomly initialized model) put together for comparison. The heatmaps in **Figure 31** are shown to the right of the heatmaps in **Figure 22**.

7.2.6. Multi-Person Pose Model

We train the conditional model with multi-person pose training data. We keep the model the same as in the single-person case. The training algorithm is still WGAN-LP with the same parameters $k = 5, \alpha = 1, \lambda = 150, \beta = 0.5$. All the network parameters are randomly initialized. The optimizer is Adam with $\beta_1 = 0, \beta_2 = 0.9$, minibatch size $m = 128$ and learning rate 0.0004. **Figure 33** and **Figure 34** are the training results.

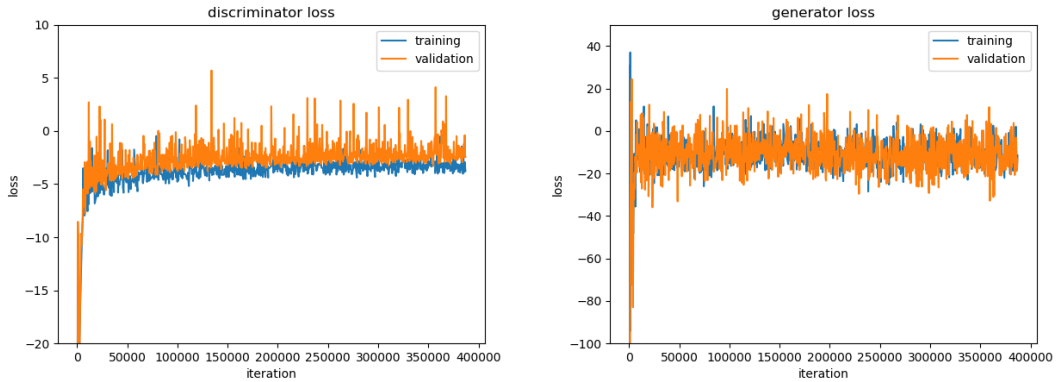


Figure 33 Loss curves of the conditional model with WGAN-LP algorithm trained on multi-person poses.

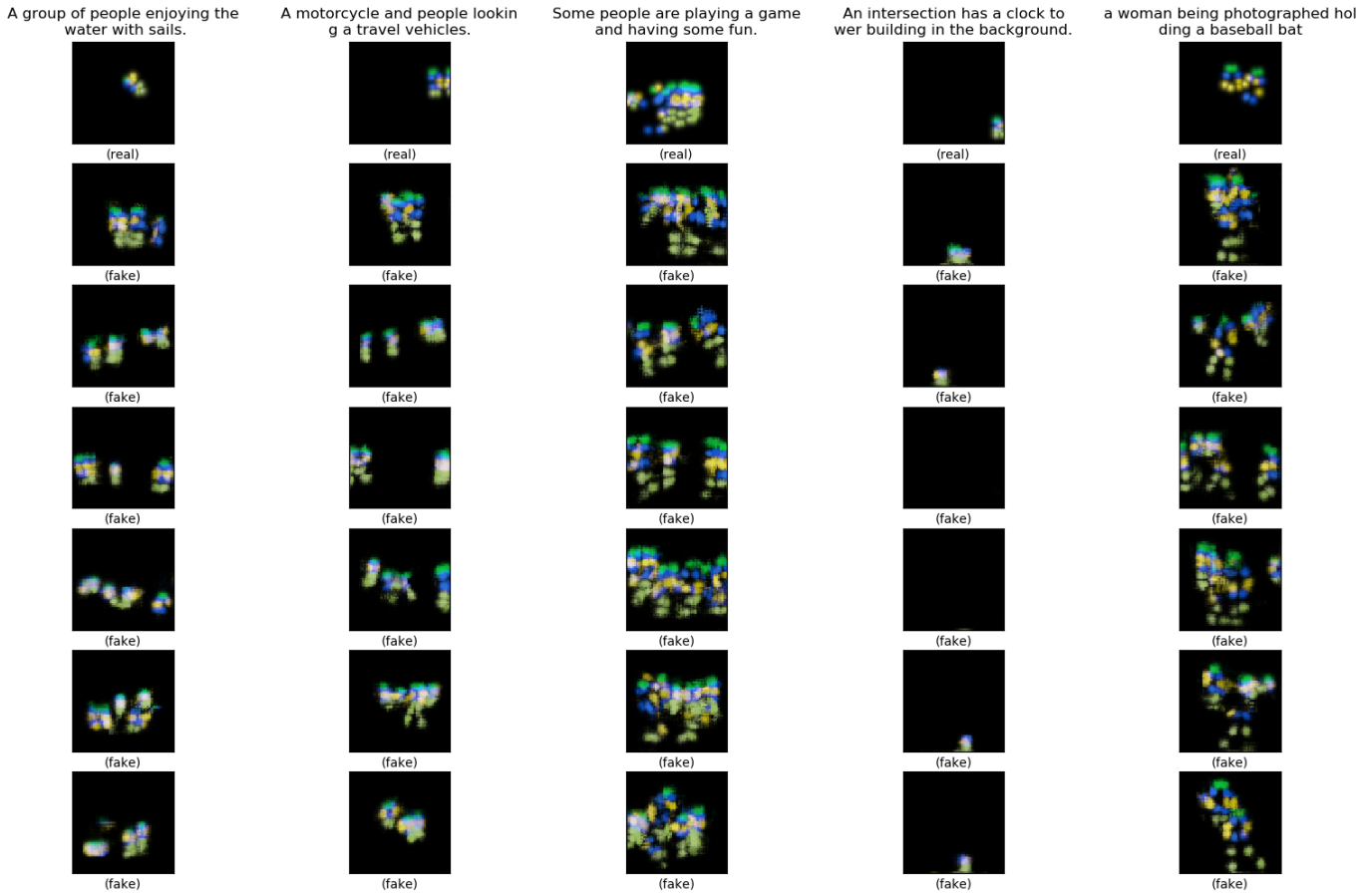


Figure 34 Some sample outputs of the generator of the conditional model with WGAN-LP algorithm trained on multi-person poses. The first row is five sample poses from the validation set. The text on the top is the sample poses' accompanying captions. The six heatmaps below each sample pose are synthesized by the model from the caption on the top. Within the same row, the noise inputs to the generator are the same.

From the sample outputs, we can see that the model is able to synthesize poses of more than one person when given captions describing activities involving more than one person. In the fourth column counted from left in **Figure 34**, the caption is not about person, and in the generated heatmap, the human pose almost disappears. This is quite intelligent. Our attempt to synthesize multi-person poses seems promising. But the synthesized poses are relatively too small, sometimes not located in the center place. This is because our heatmap resolution is low, and the persons in the training images are sometimes small or not located in the center place. And it may not be easy to extract separate poses from the heatmaps, let alone overlapping poses. Besides, the loss curves look less 'stable' than those for training a single-person model, which might indicate that maybe the learning of the model is not as good.

Also, multi-person captions can describe several poses, and there is no special handling in our model of such as complicated case, which possibly requires relating different parts of text to different poses. Our model capacity is designed to handle a single pose at a time. We leave the improvement of multi-person pose models to future projects.

7.2.7. Evaluation of the Conditional WGAN-LP Model

7.2.7.1. Interpolations

Interpolating the noise and condition inputs (z and h) to the generator and comparing the outputs is a popular way to check if the model has learned the data distribution well. For noise interpolation, we keep the condition (text encoding) h fixed, take two different noises z_1 and z_2, and input the condition h and the interpolated noise $z = \beta z_1 + (1-\beta)z_2$ to the generator, with β increasing from 0 to 1; for condition interpolation, we keep the noise z fixed, take two different caption encodings h_1 and h_2, and input the noise z and the interpolated text encoding $h = \beta h_1 + (1-\beta)h_2$ to the generator, with β increasing from 0 to 1. If the outputs show smooth transition between the interpolations, then we can say that the model learns a proper distribution instead of just memorizing the training samples [6][9].

We first interpolate the noise z. **Figure 35** is the results. The captions are from the validation set. There is smooth transition between the poses along the row.

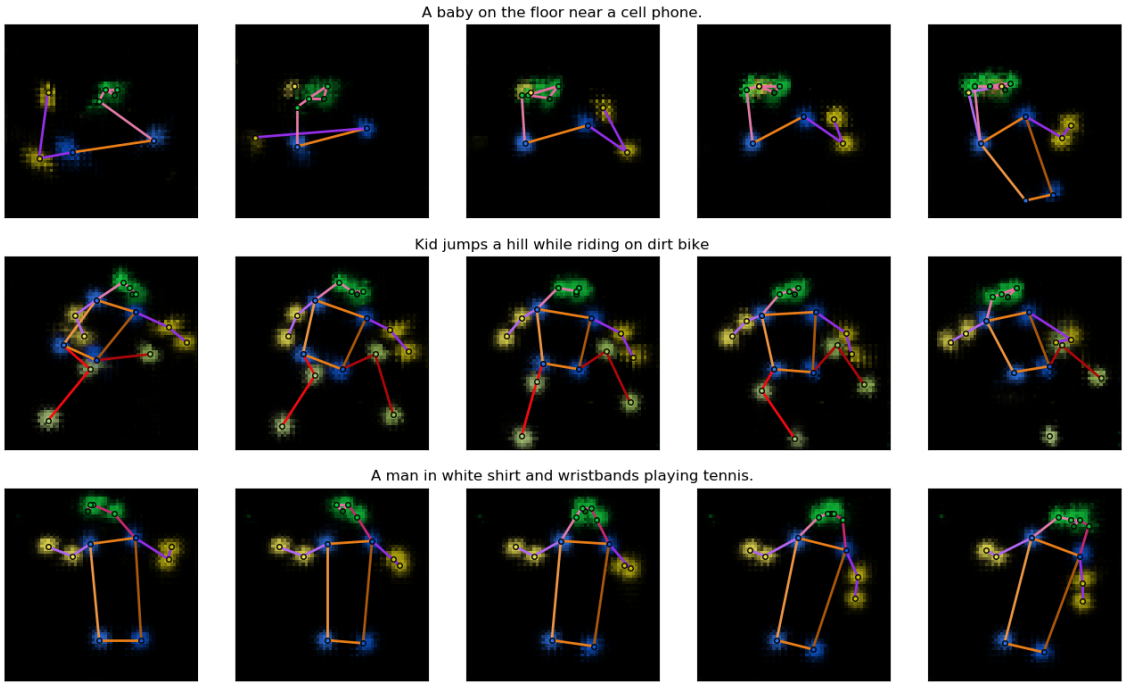


Figure 35 Noise interpolation results. In each row, the five poses are synthesized from the caption on the top. The noise inputs of the three poses in the middle are interpolated between the noise inputs of the leftmost and rightmost poses.

We then interpolate the caption encoding h. **Figure 36** is the results. We make up these captions ourselves to better observe the transition. Each pair of the captions are different but also related. There is smooth transition between the poses along the row, even though the interpolated encodings do not encode any real caption.

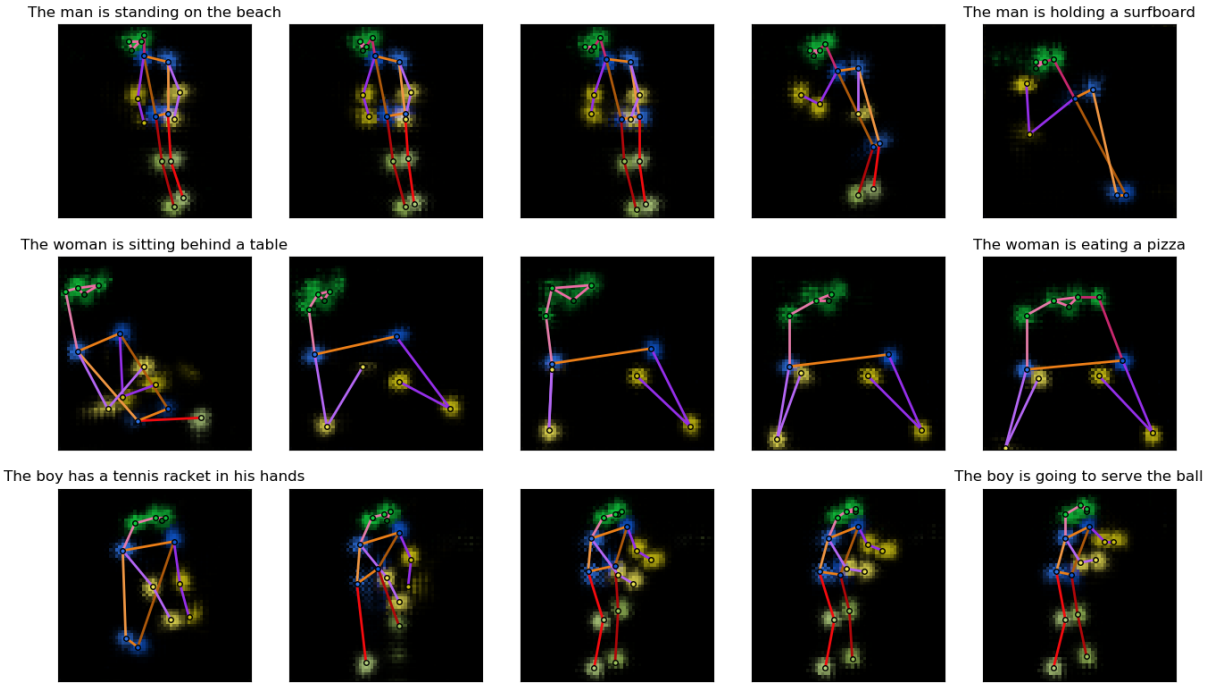


Figure 36 Caption encoding interpolation results. In each row, the leftmost and rightmost poses are synthesized from the captions on top of them. The three poses in the middle are synthesized from interpolations of the encodings of the two captions. The noise inputs within each row is fixed.

Our model can pass the interpolation test. It can represent the real data conditional distribution well.

7.2.7.2. Quantitative Measures

Many quantitative measures are targeted at unconditional generative models or models conditioned on class labels [9]. Our model is conditioned on text instead, but we can still use the quantitative measures to evaluate our model as a reference. The measures mainly compare two distributions, namely the distribution of the fake data generated by the model P_g and the distribution of real data P_r, and the closer those two distributions are, the better the model is. We will use the training set to represent P_r. And we change our fake data from conditional ones to unconditional ones by synthesizing one fake pose for each real pose from the training set using one of its five captions, and therefore we will have equal number of fake data samples (set S_g) and real data samples (set S_r) at hand and we do not care about the condition (the captions).

The first measure is called the "classifier two-sample tests" [9]. Here we construct a 1-nearest-neighbor (1-NN) classifier from the fake sample set S_g and real sample set S_r: given a sample, if its nearest neighbor is from S_g, then it is classified as fake; if its nearest neighbor is from S_r, then it is classified as real. And then we calculate the leave-one-out (LOO) accuracy of the 1-NN classifier on our fake and real samples in S_g and S_r, meaning that when classifying a fake sample we leave this sample out from S_g to build the classifier, and when classifying a real sample we leave this sample out from S_r to build the classifier. Obviously, if the model just memorizes all real samples, the accuracy is 0%; if the model learns a completely wrong distribution, the accuracy is

100%; if the model is perfect, the accuracy should be 50%. And the closer to 50% the LOO accuracy is, the better the model is. In our case, an issue is how to define the distance between our samples. As our samples are poses, a natural approach is to sum up the Euclidean distances between corresponding keypoints of the two poses. When a keypoint does not exist in a pose, we just pretend that it is located at the center of the heatmap (at $[32,32]$), which can give out a reasonable distance. The LOO accuracy of our model $\approx 66.7\%$, which is good.

The second measure is the "image retrieval performance" [9]. This time we will use our validation set as well. For each pose x_i from the validation set, we calculate d_i^r, the distance to its nearest neighbor in S_r, and d_i^g, the distance to its nearest neighbor in S_g. If the model is good, the two average distances $\overline{d^r}$ and $\overline{d^g}$ over all x_i should be very close. For our model, $\overline{d^r} \approx 100.9, \overline{d^g} \approx 105.2$, $\overline{d^g}$ is only about 4% larger than $\overline{d^r}$. We can also plot the histograms of d_i^r and d_i^g (**Figure 37**). The two histograms also look alike. Therefore, in term of this measure, our model performs very well.

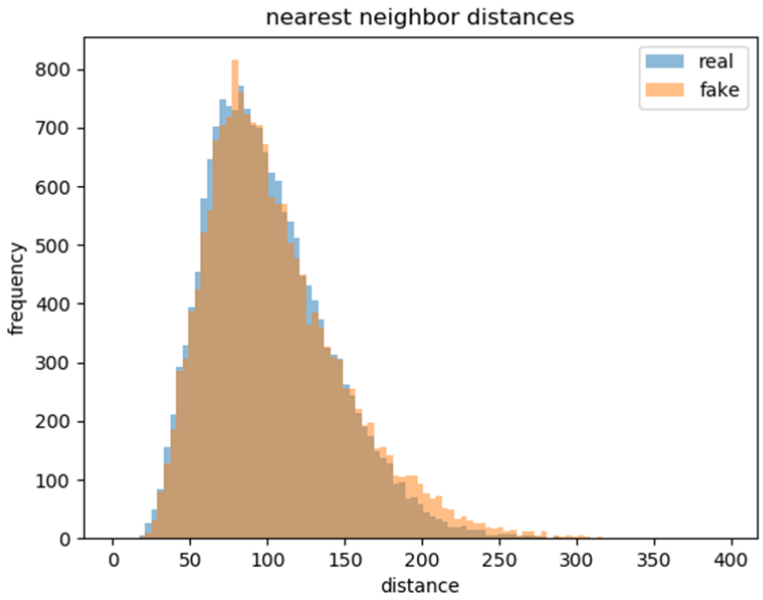


Figure 37 Histograms of nearest neighbor distances of poses in the validation set to the real sample set (blue) and the fake sample set (orange).

The above two measures indicate that our model is reasonably good at least in the unconditional sense. Nonetheless, neither of them counts captions. Good quantitative conditional GAN measures might evaluate our conditional model better.

So we propose the third measure – our own conditional measure to evaluate our model, inspired by the above two unconditional measures. Our measure is not perfect but might still tell us something about the performance of our model. We again use the validation set. From each caption h_i in the validation set, the model synthesizes 10 fake poses $\{x_i^1, \cdots, x_i^{10}\}$ using different noises. The caption's ground truth pose in the validation set is x_i^r. And now we calculate three distances for $\{x_i^1, \cdots, x_i^{10}\}$ and compute the average values among the 10 fake poses: the first d_i^{nn} is the average distance to their nearest neighbors among all ground truth poses $\{x_1^r, \cdots, x_n^r\}$; the second d_i^{gt} is the average distance to their ground truth x_i^r; and the third d_i^m is the average mean distance to all ground truth poses $\{x_1^r, \cdots, x_n^r\}$. Ideally, d_i^{nn} and d_i^{gt} should be equal, meaning that the nearest neighbor is always exactly the ground truth and the model generates fake poses very similar

to the ground truth pose of the same caption h_i, and d_i^{gt} should also be significantly smaller than d_i^m, meaning that the poses synthesized from the caption h_i are rather different from poses of the other captions. The closer d_i^{gt} is to d_i^{nn} and the smaller d_i^{gt} is compared to d_i^m, the better the model can model the input condition (the caption) instead of just generating random irrelevant poses.

The results for our model: the probability that a generated pose's nearest neighbor is indeed the ground truth pose of the same caption is only about 1%, but the averages over all generated poses $\overline{d^{nn}} \approx 105.5, \overline{d^{gt}} \approx 251.1, \overline{d^m} \approx 289.7$. Though much larger than $\overline{d^{nn}}$, $\overline{d^{gt}}$ is smaller than $\overline{d^m}$, meaning that the condition is guiding the synthesis at least to the correct direction. The histograms (**Figure 38**) show this more clearly. We can see that the nearest neighbor distances are much small than the mean distances and the ground truth distances shift from the mean towards the nearest neighbor. These results indicate that our model can handle the condition properly.

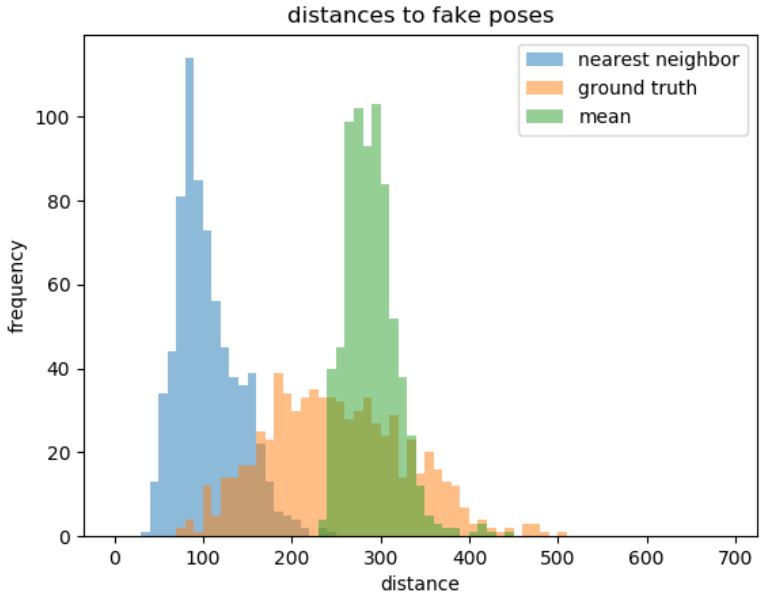


Figure 38 Histograms of distances between fake poses and their nearest neighbors in the validation set (blue), distances between fake poses and their ground truth poses (orange), and fake poses' mean distances to poses in the validation set (green).

Besides the above conditional measure, we can also calculate distances (L^2 distances) in the caption encoding space: to compare the average distance $d_{h,i}^{nn}$ between the fake poses' ($\{x_i^1, \cdots, x_i^{10}\}$) caption encoding h_i to their nearest real poses' caption encodings ($\{h_i^{nn,1}, \cdots, h_i^{nn,10}\}$), and h_i's mean distance $d_{h,i}^m$ to all the caption encodings. For our model, on average, those two distances $\overline{d_h^{nn}} \approx 10.5$ and $\overline{d_h^m} \approx 10.9$, are very close, but when looking at their histograms (**Figure 39**), we can see that the nearest neighbor distances have some small values, meaning that sometimes the nearest neighbor has similar captions with the fake pose. But it can also happen that two poses are very alike, but their captions are quite different.

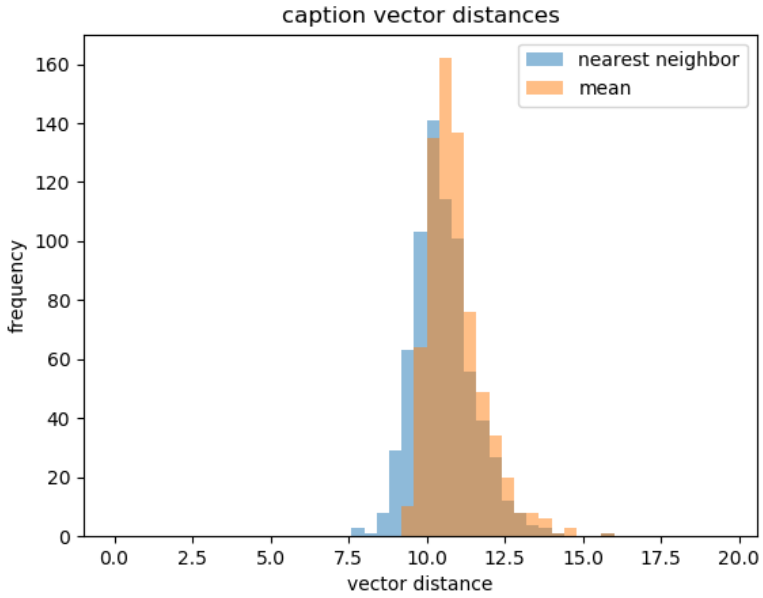


Figure 39 Histograms of distance fake poses' caption encodings to their nearest real poses' caption encodings (blue) and the fake poses' caption encodings' mean distance to all the caption encodings (orange).

We also use these measures on the randomly initialized model, the model trained with WGAN-GP, as well as the multi-person model (for this, the data sets used for evaluation are the same as the single-person models). **Table 5** compares the results of them. We can see that when the model gets specially initialized (using unconditional model's training result), the performance is only slightly better. As for the model trained with the WGAN-GP algorithm, the performance is almost equally good, compared with the model trained with the WGAN-LP algorithm. We may come to the conclusion that there is no significant difference between WGAN-GP and WGAN-LP in model performance for our task, both qualitatively and quantitatively. On the other hand, the multi-person model's performance is much worse, meaning that our model needs much improvement to deal with captions which may describe activities involving more than one person and to properly synthesize multi-person poses.

Measure	Value	Specially initialized	Trained with WGAN-GP	Randomly initialized	Multi-person
classifier two-sample tests	LOO accuracy	66.7%	65.3%	68.6%	92.7%
image retrieval performance	$\overline{d^r}$	100.9			
	$\overline{d^g}$	105.2	106.4	112.0	157.7
our own conditional measure	$\overline{d^{nn}}$	107.4	107.8	112.0	158.7
	$\overline{d^{gt}}$	252.4	254.0	254.4	278.4
	$\overline{d^m}$	292.3	291.3	292.5	296.4

Table 5 Quantitative measures on 1. the (conditional) model initialized from the unconditional model (specially initialized) trained with WGAN-LP, 2. the model specially initialized trained with WGAN-GP, 3. the model randomly initialized trained with WGAN-LP, and 4. The model randomly initialized trained on multi-person poses trained with WGAN-LP.

7.2.7.3. Subjective Evaluation

Finding good measures for generative models is a big challenge, and none of the measures available now can evaluation the models in a very thorough way [9]. So sometimes people turn to subjective evaluation such as user tests. We will let human beings evaluate our model, too.

We do this through an online questionnaire. On the questionnaire there are 20 questions, and in each question, the subject is presented with one caption and two human poses. The caption is from the validation set. One pose is the real pose of this caption, and the other pose is synthesized from this caption by our model. The 20 captions are randomly selected from all the captions describing human activities in the validation set and are not cherry-picked. The subject is asked to choose which of the two poses matches the caption better or if they match the caption equally well. Apparently, the higher the percentage of choosing the fake poses or "equally well" is, the better the model performs. **Table 6** shows the captions and fake/real poses on our questionnaire as well as the response percentage. We have in total 80 subjects to respond to this questionnaire. For reference, **Figure 40** is a standard pose, which is also shown to the subjects at the beginning of the questionnaire.

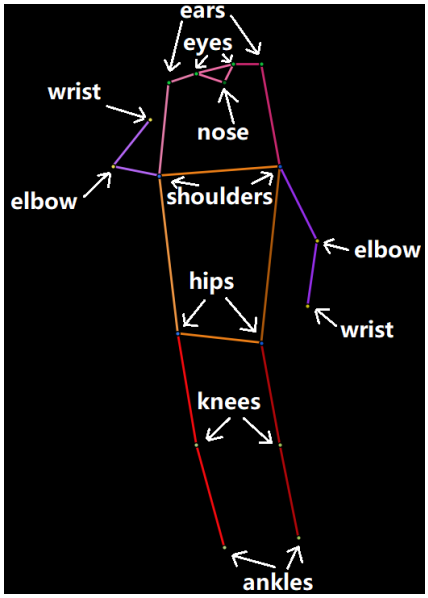
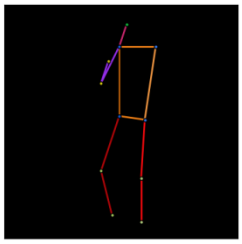
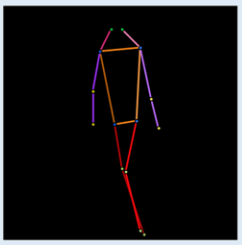
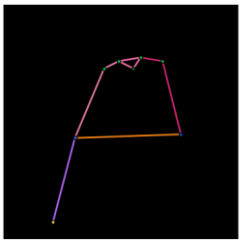
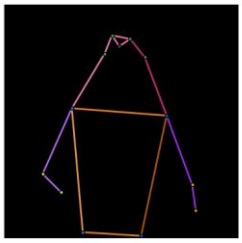
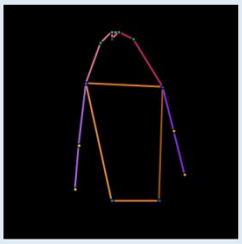
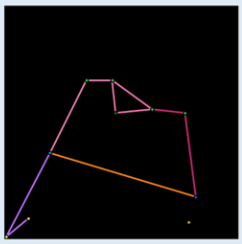
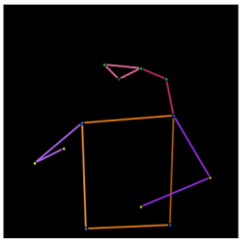
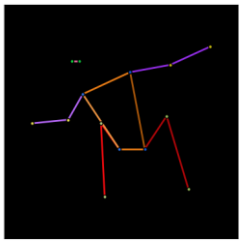
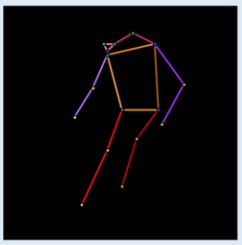
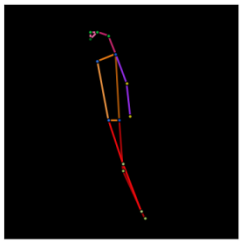
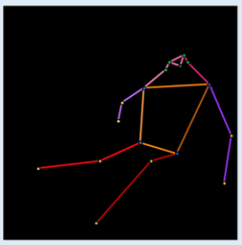
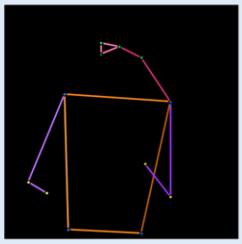
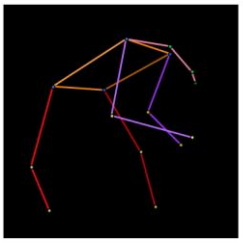
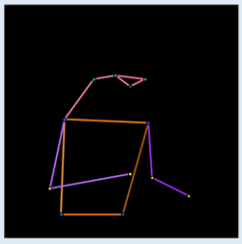
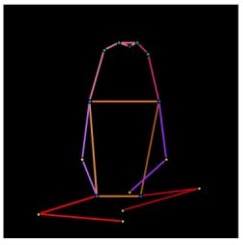
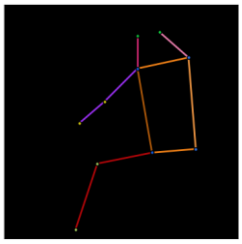
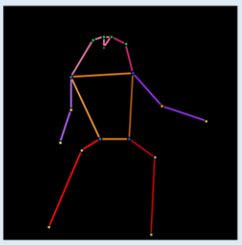
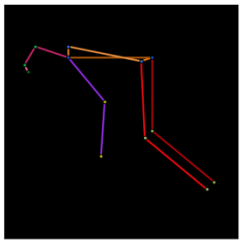
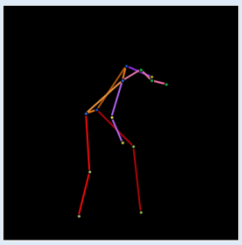
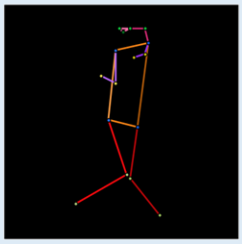
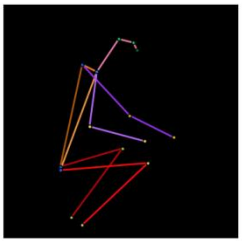
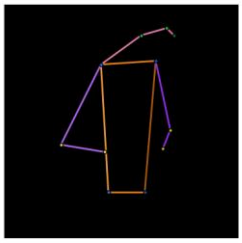
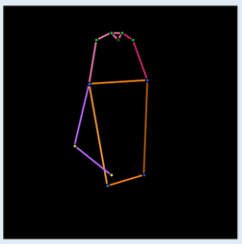
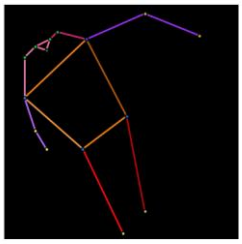
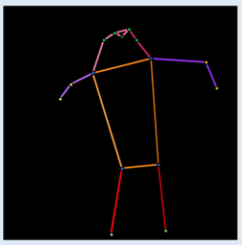
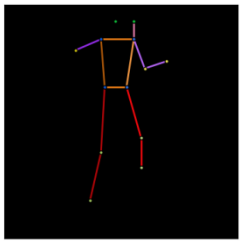
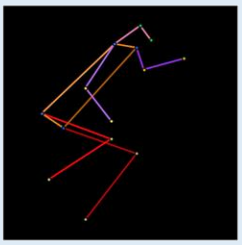
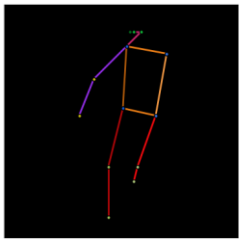
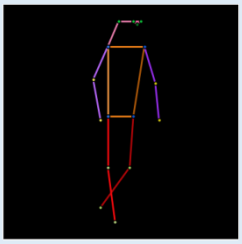
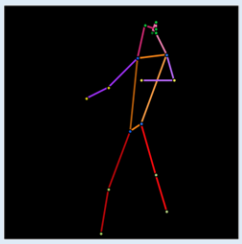
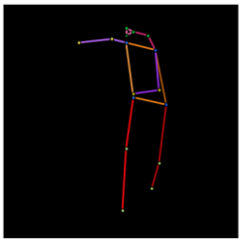


Figure 40 Standard pose for reference on the questionnaire.

1	A man walking toward a line of surfboards lined up along a shop wall.		
			equally well
	23.75%	66.25%	10.00%
2	A woman looking at a cat licking pizza grease		
			equally well
	70.00%	15.00%	15.00%
3	A man is standing talking in front of a picture.		
			equally well
	50.00%	22.50%	27.50%
4	The baby is holding and looking at his tooth brush.		
			equally well
	17.50%	75.00%	7.50%

5	White suited snowboarder displaying aerial skills at ski slope.		
			equally well
	43.75%	40.00%	16.25%
6	A girl glides down a snowy hill on a snowboard.		
			equally well
	31.25%	56.25%	12.50%
7	a man holding a cat in front of a stove		
			equally well
	53.75%	37.50%	8.75%
8	A young child that is sitting by a red cake.		
			equally well
	27.50%	57.50%	15.00%

9	A man riding a motorcycle in overcast weather		
			equally well
	47.50%	25.00%	27.50%
10	A boy skateboarding on a skateboard ramp		
			equally well
	47.50%	36.25%	16.25%
11	A baseball catcher stands ready to catch a ball.		
			equally well
	28.75%	61.25%	10.00%
12	A man is leading a cow with a ribbon on its neck.		
			equally well
	28.75%	47.50%	23.75%

13	A young man is catching or throwing a frisbee,		
			equally well
	48.75%	22.50%	28.75%
14	A person on skis sliding up down a path.		
			equally well
	33.75%	51.25%	15.00%
15	A man walking across a dirt field next to a street with traffic.		
			equally well
	50.00%	36.25%	13.75%
16	a woman running across a tennis court so she can hit the ball		
			equally well
	17.50%	70.00%	12.50%

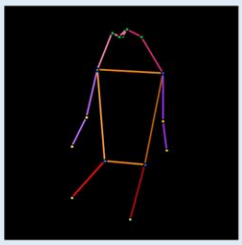
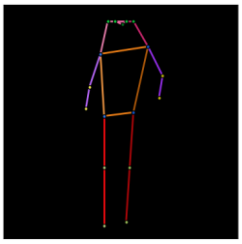
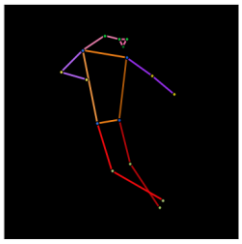
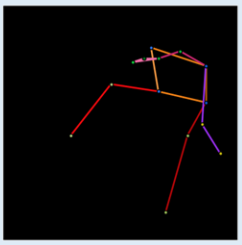
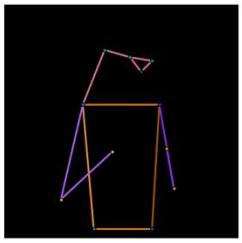
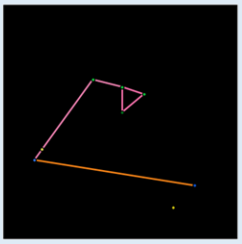
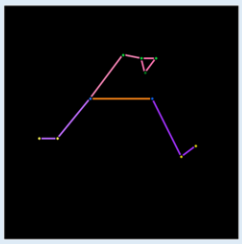
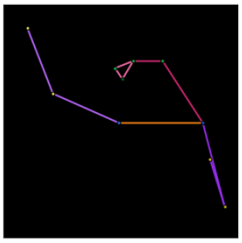
17	A formally dressed man holding his bike.		
			equally well
	3.75%	86.25%	10.00%
18	A skateboarder doing tricks on the rim of a pit		
			equally well
	31.25%	48.75%	20.00%
19	A woman looks down at her phone in her hands.		
			equally well
	85.00%	10.00%	5.00%
20	A woman holding up a large carrot in a backyard.		
			equally well
	25.00%	52.50%	22.50%
overall	**fake**	**real**	**equal**
	35.31%	48.81%	15.88%

Table 6 The captions, the fake and real poses on the questionnaire, and the response percentage. Fake poses are in blue background.

The responses are quite encouraging. Although “equally well” only takes up 15.88%, the ratio between choosing fake and real is around 5:7, which is quite close. And if we see it from another point of view, for more than half of the times (35.31%+15.88%>50%), the subjects cannot correctly pick out the real poses. This is just what we would like to achieve. But the quality of fake poses still has some variance. For example, some of the generated poses look very ‘real’ (like 1, 2 and 6), while some look very ‘fake’ (like 16, 17 and 19). Despite this, we are satisfied with the subjective evaluation result.

8. Conclusion

In this thesis we demonstrate the possibility of synthesizing human poses from captions through designing and implementing a conditional GAN model with a not very complicated architecture. The text encoder used in the model is also simple and pre-trained for general purpose. But the synthesis results are quite promising. This type of architecture (DCGAN) has been widely used to synthesize natural images, and our experiments prove that it is also capable of fulfilling tasks like ours. Achieving semantic consistency between text and image features has been successful before by others and we have shown through our work that it is also possible to achieve semantic consistency between text and pose features.

Our experiments also demonstrate that for the same model, different training algorithms can affect the quality of the final model. We observe evident improvement of the unconditional model's performance from the common GAN algorithm, the WGAN with parameter clamping, to the WGAN-GP. An interesting finding is that WGAN-GP and WGAN-LP seem to lead to the same level of model performance, which is a little different from the conclusion by Petzka et al. [20]. And finally, we conduct a survey to evaluate our model, which proves to be interesting and useful.

There is still much to do to optimize our model. For example, we could replace the fastText encoder with a better text encoder which operates on both sentence level and word level, or with some advanced text model trained jointly with visual features. We could also find or make bigger and more suitable pose-caption data sets for training. Also, as mentioned previously, we can upgrade the model to make it more capable of synthesizing multi-person poses. Apart from the model itself, finding good quantitative measures for the evaluation of the conditional model is another interesting research direction.

The synthesized poses could be helpful in human image synthesis, as has been discussed in the introduction part. The problem of human image synthesis from text is very complicated, but we can divide and conquer it. It can be divided into several quite independent parts: the human pose, the colors of the clothes, the facial expression, the environment, etc. Our model can solve the first part, thus contributing to the solution of the whole problem.

Besides, our results prove the usefulness of our architecture and algorithm, and they could in the future be used to synthesize other things, like 3D human poses, object poses, scenes, and even animations, etc. from text.

References

[1] Goodfellow, Ian et al. ***Generative adversarial nets.*** Advances in Neural Information Processing Systems 27. 2014.

[2] Radford, Alec et al. ***Unsupervised representation learning with deep convolutional generative adversarial networks.*** International Conference on Learning Representations. 2016.

[3] Reed, Scott et al. ***Generative adversarial text to image synthesis.*** International Conference on Machine Learning. 2016.

[4] Liu, Yifan et al. ***Auto-painter: cartoon image generation from sketch by using conditional Wasserstein generative adversarial networks.*** Neurocomputing 311. 2018.

[5] Zhu, Lin et al. ***Generative adversarial networks for hyperspectral image classification.*** IEEE Transactions on Geoscience and Remote Sensing 56(9). 2018.

[6] Bodnar, Cristian. ***Text to image synthesis using generative adversarial networks.*** University of Manchester thesis. 2018.

[7] Tanke, Julian. ***3D pose tracking from multiple views.*** University of Bonn thesis. 2018.

[8] Tanke, Julian and Gall, Juergen. ***Iterative greedy matching for 3D human pose tracking from multiple views.*** German Conference on Pattern Recognition. 2019.

[9] Borji, Ali. ***Pros and cons of GAN evaluation measures.*** Computer Vision and Image Understanding 179. 2019.

[10] Gulrajani, Ishaan et al. ***Improved training of Wasserstein GANs.*** Advances in Neural Information Processing Systems 30. 2017.

[11] Long, Jonathan et al. ***Fully convolutional networks for semantic segmentation.*** IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[12] Nair, Vinod and Hinton, Geoffrey E. ***Rectified linear units improve restricted Boltzmann machines.*** International Conference on Machine Learning. 2010.

[13] Maas, Andrew L. et al. ***Rectifier nonlinearities improve neural network acoustic models.*** International Conference on Machine Learning. 2013.

[14] Ioffe, Sergey and Szegedy, Christian. ***Batch normalization: accelerating deep network training by reducing internal covariate shift.*** International Conference on Machine Learning. 2015.

[15] Kingma, Diederik P and Ba, Jimmy. ***Adam: a method for stochastic optimization.*** International Conference on Learning Representations. 2015.

[16] Arjovsky, Martin et al. ***Wasserstein generative adversarial networks.*** International Conference on Machine Learning. 2017.

[17] Villani, Cedric. ***Optimal transport: old and new.*** Springer Science & Business Media. 2008.

[18] Tieleman, Tijmen and Hinton, Geoffrey. ***Lecture 6.5-RMSProp: divide the gradient by a running average of its recent magnitude.*** COURSERA: Neural Networks for Machine Learning. 2012.

[19] Reed, Scott et al. ***Learning deep representations of fine-grained visual descriptions.*** IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[20] Petzka, Henning et al. ***On the regularization of Wasserstein GANs.*** International Conference on Learning Representations. 2018.

[21] Bojanowski, Piotr et al. ***Enriching word vectors with subword information.*** Transactions of

the Association for Computational Linguistics 5. 2017.
[22] Lin, Tsung-Yi et al. ***Microsoft COCO: common objects in context.*** European Conference on Computer Vision. 2014.
[23] Martinez, Julieta et al. ***On human motion prediction using recurrent neural networks.*** IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[24] Zhang, Han et al. ***StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks.*** IEEE International Conference on Computer Vision. 2017.
[25] Ma, Liqian et al. ***Pose guided person image generation.*** Advances in Neural Information Processing Systems. 2017.
[26] Dong, Haoye et al. ***Soft-gated warping-GAN for pose-guided person image synthesis.*** Advances in neural information processing systems. 2018.
[27] Balakrishnan, Guha et al. ***Synthesizing images of humans in unseen poses.*** IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[28] Toshev, Alexander and Szegedy, Christian. ***DeepPose: human pose estimation via deep neural networks.*** IEEE Conference on Computer Vision and Pattern Recognition. 2014.
[29] Wei, Shih-En et al. ***Convolutional pose machines.*** IEEE Conference on Computer Vision and Pattern Recognition. 2016.
[30] He, Kaiming et al. ***Mask R-CNN.*** IEEE International Conference on Computer Vision. 2017.
[31] Cao, Zhe et al. ***Realtime multi-person 2D pose estimation using part affinity fields.*** IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[32] He, Kaiming, et al. ***Delving deep into rectifiers: surpassing human-level performance on ImageNet classification.*** IEEE International Conference on Computer Vision. 2015.
[33] Iqbal, Umar and Gall, Juergen. ***Multi-person pose estimation with local joint-to-person associations.*** European Conference on Computer Vision. 2016.
[34] Pishchulin, Leonid et al. ***DeepCut: joint subset partition and labeling for multi person pose estimation.*** IEEE Conference on Computer Vision and Pattern Recognition. 2016.