Synthesizing Human Pose from Captions

1. Introduction

The generative model is an important topic of machine learning. This kind of model can learn from training data (like pictures) their underlying patterns, and then generates ‘synthesized’ new data that share the same probability distribution with the training data. For example, generative models succeed in synthesizing digits and pictures of human faces, bedrooms, birds and flowers, etc. that look realistic to human [1][2][3]. Generative models are also useful in image-to-image transferring [4], image classification [2][5], and many other applications. Recent research even goes as far as achieving automatic synthesis of realistic images from captions [3][6], which is of great interest and usefulness.

On the other hand, 2D and 3D human poses are popular research topics in the field of computer vision. Generally, human poses mean the relative locations of several keypoints on the human body (shoulders, hips, etc.). For example, human pose estimation from images and videos have many applications in areas such as surveillance and sports [7][8].

In this thesis we will combine the above two interesting topics and research on generative models for human poses. So far, there has been little such research work. As human poses are quite different from common images, it is interesting to see whether generative models with architectures similar to those of image generative models are able to synthesize realistic human poses. And our ultimate goal is to create a model that can automatically synthesize poses from input captions such as ‘the man is playing tennis’.

2. Background and Related Work

2.1. Generative Adversarial Networks

A generative model is a model whose outputs have a distribution $\mathrm{P_g}$ that estimates the distribution of real data $\mathrm{P_r}$. For a perfect model, $\mathrm{P_g}$ and $\mathrm{P_r}$ should be identical and therefore nothing could tell an output from the model and a sample from the real data apart. The generative adversarial network (GAN) [1], which is currently one of the most popular and powerful classes of generative models [9][10], is designed based on this idea. A GAN actually consists of two parts: a generator network G, and a discriminator network D. Both G and D can be simple multilayer perceptrons, the more complex

convolutional neural networks (CNNs) or other types of network. G is train to generate 'fake' samples while D is trained to correctly discriminate between fake samples and samples from the real data. The two network are trained simultaneously. This can be seen as a two-player game and the theoretical training outcome is that G can generate very realistic fake samples that D can no longer discriminate from real samples.

Mathematically, $G(z)$ represents an output fake sample of G, where z is a noise input to G. The noise z is sampled from some simple noise distribution P_z, for example, normal distribution, and it is necessarily so that the output samples of G have some variance. $D(x) \in [0,1]$ is the discriminative 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 from the real data and be lower if x is generated. When D is being trained, the objective is to maximize both $\log(D(x))$ with $x \sim P_r$ and $\log(1 - D(G(z)))$ with $z \sim P_z$, that is, to correctly label both samples from the real data and fake samples; when G is being trained, the objective is to minimize $\log(1 - D(G(z)))$ with $z \sim P_z$, that is, to fool D with the generated fake samples. So he overall training objective can be seen as the following minimax game (here E means expected value):

$$\min_G \max_D \{E_{x \sim P_r}[\log(D(x))] + E_{z \sim P_z}[\log(1 - D(G(z)))]\} \quad (1)$$

Normally, gradient-based learning algorithms such as stochastic gradient descent (SGD) are applied to train G and D. But as pointed out in [1], early in the training $\log(1 - D(G(z)))$ saturates, and provides insufficient gradient for G to learn. Therefore, in practice, when G is trained, the objective is maximizing $\log(D(G(z)))$ instead of minimizing $\log(1 - D(G(z)))$, as $\log(D(G(z)))$ will not saturate early in the training. The above description can be summarized by the following algorithm:

Algorithm 1 GAN [1]

---

for number of training iterations **do**

  for k steps **do**

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

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

    Update D using the gradient:

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

  end for

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

  Update G using the gradient:

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

---

end for

As shown, D and G are trained alternately: k steps of updating D and then one step of updating G. Usually $\mathrm{k}=1$ is chosen, which is the least computationally expensive [1].

In [1], the authors proof that theoretically,if D and G have infinite capacity, **Algorithm 1** will eventually converge to the global optimum, where $\mathrm{P_g}$ (the distribution of output samples of G given that the input noise has distribution $\mathrm{P_z}$) and $\mathrm{P_r}$ are identical and the output of D is 0.5. 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 GANs is easy to understand and the architecture is not very complicated. But one challenges aspect of GANs is the training instability [2][10][16]. For example, one common failure of training GANs is called mode collapse, where the output of G collapses into only a few nonsensical samples [1][2]. Also, good training hyperparameters are hard to find [16].

The authors of [2] take a deep look into GANs and try to overcome the instability problem. They propose a specific class of GAN called the Deep Convolutional GAN (DCGAN).

The D and G of DCGAN contain layers of strided convolutions and transposed convolutions [11] (also called fractionally-strided convolutions), respectively. In the D, strided convolutions are used for downsampling. This is different from usual CNNs, which use pooling functions (such as maxpooling) for downsampling instead. In their experiments, with five layers of strided convolutions, the D takes an input image of size $64 \times 64 \times 3$ ($\mathrm{width} \times \mathrm{height} \times \mathrm{channel}$) and produce one number as the output discriminating result. And in the G, transposed convolutions play the role of upsampling and in their experiments G also has five layers of transposed convolutions and transforms an input noise z into an output colored image of size $64 \times 64 \times 3$. The activation function used in G is the rectified linear unit (ReLU) [12] while in D it is the leaky ReLU [13]. What's more, to stabilize training, batch normalization [14] is applied in each layer of convolution/transposed convolution before the nonlinear activation function, except for the first layer in D and the last layer in G.

These authors find the above architecture for DCGAN plus the Adam (adaptive moment estimation) optimizer [15] with tuned hyperparameters to be relatively stable for training through their experiments. And they manage to generate high quality colored images of human faces and bedrooms. Considering their success and the relatively simple architecture of DCGAN, the models in this thesis are all based on DCGAN.

Even with the above specifications of DCGAN, the stability of GANs is still so satisfactory [2]. And much of recent work focuses on this problem [10][16]. [1] proofs that the optimization objective of GANs (**Expression 1**) is actually related to the Jensen-Shannon divergence between the two distributions $\mathrm{P_g}$ and $\mathrm{P_r}$. As pointed out in [16], Jensen-Shannon divergence is not continuous when the two distributions' supports have negligible intersection and therefore

does not provide useful gradients for training. And this is a cause for GANs' instability (detailed analysis in [16]). So the author put forward another way to compare two distributions, the earth mover's or Wasserstein-1 distance $W(P_r, P_g)$, which has better continuity and provide better gradients than Jensen-Shannon divergence, and can be calculated by the following equation [17]:

$$K \cdot W(P_r, P_g) = \sup_{\|f\|_L \le K} E_{x \sim P_r}[f(x)] - E_{x \sim P_g}[f(x)] \tag{2}$$

Here K is some constant and $\|f\|_L \le K$ means f is a K-Lipschitz continuous function[1]. When f is realized by a neural network, clamping all the network's parameters to a fixed interval (the authors use $[-0.01, 0.01]$) can assure the K-Lipschitz requirement for some K. 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. Meanwhile the generator G's task is to reduce this distance by generating fake samples. So the optimization objective of the GAN (**Expression 1**) now becomes:

$$\min_G \max_D \{E_{x \sim P_r}[D(x)] - E_{z \sim P_z}[D(G(z))]\} \tag{3}$$

Here the maximization part corresponds to **Expression 2**, estimating the Wasserstein-1 distance. We should remember that D's parameters must be clamped, as explained above. And the authors of [16] name this type of GAN with the above optimization objective the Wasserstein GAN (WGAN) (**Algorithm 2**).

Algorithm 2 WGAN [16]

---

for number of training iterations **do**

  for k steps **do**

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

  Update G's parameters using the gradient:

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

end for

Algorithm 2 has a hyperparameter c for parameter clamping. And the authors argue that as the Wasserstein-1 distance is continuous, D should be trained to optimal before G get updated [16], so they choose $k = 5$ instead of the commonly chosen $k = 1$ in **Algorithm 1**. They also find out that

---

[1] Simply put, f is K-Lipschitz continuous if there exists a constant K such that for all x, y from f's domain, $|f(x) - f(y)| \le K|x - y|$.

momentum-based optimizers such as Adam will lead to unstable training so they switch to RMSProp (root mean square propagation) [18], which does not use momentum.

[16]'s experiments show that the WGAN has two obvious advantages over the standard GAN (**Algorithm 1**). First, unlike standards GAN, the WGAN loss (the part inside the brackets in **Expression 3**) correlates well with the visual quality of the generated samples. During the training, the loss gets smaller as the generated samples get better. This provides a convenient way to check if the training goes well [16]. Second, the training stability improves. In all the experiments using the WGAN algorithm in [16], the generated images of bedrooms are visually very good, and mode collapse never occurs.

Still, the WGAN algorithm proposed by [16] has some problems. The authors of [10] observe that the parameter clamping applied in WGAN to enforce the Lipschitz constraint can lead to some undesired behaviors of the networks: 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. So they propose 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, they choose to directly constrain the discriminator D's gradient of the output with respect to the input. They implement this by adding a penalty term on the gradient norm of some random samples $\hat{x}\sim P_{\hat{x}}$ to D's loss function. $\hat{x}$'s are sampled uniformly along straight lines between pairs of points sampled from P_r and P_g. They claim that constraining gradient norm at such $\hat{x}$ is a sufficient measure to enforce D's Lipschitz constraint [10]. Therefore the loss function of D is now changed to:

$$E_{z\sim P_z}[D(G(z))] - E_{x\sim P_r}[D(x)] + \lambda E_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right] \qquad (4)$$

Here the third term is the added gradient penalty term and λ is the penalty coefficient. The authors use $\lambda = 10$ in their experiments to get good results. Below is the new WGAN algorithm with gradient penalty (WGAN-GP):

Algorithm 3 WGAN-GP [10]

---

for number of training iterations **do**

  for k steps **do**

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

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

  end for

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

---

Update G using the gradient:

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

end for

For WGAN-GP, there should not be batch normalization in D, since it is not compatible with the loss function [10]. The authors see proved performance of WGAN-GP over WGAN, and the discriminator loss (**Expression 4**) still correlates with the quality, the same as in [16]'s WGAN.

2.2. Conditional GANs

We will use GANs to synthesize human poses from captions. But in this scenario, the GAN is not the simple unconditional GAN described in the last section because we have a condition - the caption text t. Both G and D take t as additional condition input: G generates fake samples from t and the input noise z, while D gets data samples paired with t and outputs the probability that the pairs come from the real data. Here the distribution of the real data x given t is a conditional probability distribution $P_r(x|t)$, and so is the distribution of the data generated by G given t: $P_g(x|t)$. That is the conditional GAN.

[3] is a good example work of conditional GANs. Their task is synthesizing images of birds and flowers from captions, which is very similar to our task.

First of all, they must find proper vector representations (encodings, or embeddings) of the captions so that caption information can be put into the networks. They follow the approach of [19]. In this approach, the training data contain images, the corresponding text descriptions and class labels and a text encoder is trained together with a image encoder. Basically, the encoders are trained so that a text encoding has a higher compatibility score with the image encodings of the than class than other classes and vice-versa [3]. The authors of [3] train a deep convolutional recurrent text encoder $\varphi(t)$, which produces 1024-dimensional encodings for captions t. Such encoding approach, however, cannot be adopted in this thesis, because our data do not contain class label. We will instead use a simpler text encoding solution.

Then they need to find a way to insert these caption encodings into the G and D networks. They do it in this way: in G, they first compress the encoding to 128 dimensions through a fully connected layer and then concatenate the compressed vector to the 100-dimensional noise; in D, they also compress the encoding first, then replicate it 4×4 times, and finally concatenate it to the convolutional feature maps after the fourth convolutional layer. The image size in their GAN is $64 \times 64 \times 3$.

Because this GAN has conditional input, the training process is a little different from that of the unconditional GAN (**Algorithm 1**). The authors argue that as the discriminator encounters two types of errors: unrealistic images and realistic images with mismatching captions, separating them explicitly during training can help the discriminator learn better. So they change the terms to be

maximize for D from the simple unconditional version $\log(D(x)) + \log(1 - D(G(z)))$ to:

$$\log(D(x, h)) + \frac{1}{2}\left[\log\left(1 - D(x, \hat{h})\right) + \log(1 - D(G(z, h), h))\right] \quad (5)$$

Here in the conditional GAN, both D and G have an additional input: the caption encoding h or $\hat{h}$. x and h are a pair of matching image and caption encoding from the training data, while x and $\hat{h}$ are mismatching. The second term in **Expression 5** is used to separate the above-mentioned errors.

Another modification is adding interpolated captions encodings in the training of the generator. They change the terms to be maximize for G to:

$$\log(D(G(z, h), h)) + \log\left(D(G(z, \tilde{h}), \tilde{h})\right) \quad (6)$$

where

$$\tilde{h} = \beta h_1 + (1 - \beta) h_2 \quad (7)$$

Here h, h_1 and h_2 are caption encodings from the training data; 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 G learn better [3]. In their experiments, the authors fix $\beta = 0.5$.

To sum up, the conditional GAN algorithm in [3] goes as follows:

Algorithm 4 Conditional GAN [3]

---

for number of training iterations **do**

  for k steps **do**

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

    Take a minibatch of m real image-caption pair samples $\{(x_1, t_1), \cdots, (x_m, t_m)\}$

    Take a minibatch of m mismatching captions $\{\hat{t}_1, \cdots, \hat{t}_m\}$

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

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

    Update D using the gradient:

$$-\nabla \frac{1}{m} \sum_{i=1}^{m} \left\{\log(D(x_i, h_i)) + \frac{1}{2}\left[\log\left(1 - D(x_i, \hat{h}_i)\right) + \log(1 - D(G(z_i, h_i), h_i))\right]\right\}$$

  end for

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

  Take three minibatches of m captions $\{t_1, \cdots, t_{3m}\}$

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

  Interpolate two minibatches of m captions

$\{\tilde{h}_i | \tilde{h}_i = \beta h_{i+m} + (1 - \beta) h_{i+2m}, i = 1, \cdots, m\}$

  Update G using the gradient:

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

end for

Trained on different data sets, the model of [3] successfully synthesizes visually plausible images of birds and flowers given text captions. We will base our model on the architecture and training method presented in [3], although they

do not use the WGAN algorithm. We would like to see if our model can achieve similar level of success as theirs in another task - synthesizing human pose from captions.

[6] is a piece of work that adapts the model in [3] for the WGAN algorithm so that the training becomes more robust.

First, the author adds a parameter α to **Expression 5** to controls the level of text-image matching and change the expression to the WGAN format:

$$(1+\alpha)D(x,h) - D(G(z,h),h) - \alpha D(x,\hat{h}) \tag{8}$$

That's what D is to maximize.

Second, in WGAN D must by Lipschitz continuous. In the conditional scenario, D has two inputs: the data sample (image) x and the condition (caption) h. So the Lipschitz constraint should have two terms, one contain $\nabla_{\hat{x}} D(\hat{x},h)$ the gradient with respect to x and another one $\nabla_h D(\hat{x},h)$. $\hat{x}$ as previously is sampled between a real data sample x and a generated data sample conditioned on x's matching h.

Third, in the WGAN-GP algorithm, the Lipschitz constraint is enforced by adding a penalty term with coefficient λ in D's loss function (**Expression 4**). However, in the conditional GAN, as D's loss can be much larger, λ should also be set larger, WGAN-GP algorithm is not very robust to changes in λ [6][20]. [20] proposes a new strategy called the Lipschitz penalty (LP), which is robust to λ. It just changes the penalty to one-sided, meaning encouraging the norm of the gradient to stay below 1. In contrast, in WGAN-GP, the penalty is two-sided (**Expression 4**), meaning encouraging the norm of the gradient to go towards 1 [10].

And finally, the minimization objective of D is changed to [6]:

$$D(G(z,h),h) + \alpha D(x,\hat{h}) - (1+\alpha)D(x,h) + \lambda\left\{\left[\max\left(0, \|\nabla_{\hat{x}} D(\hat{x},h)\|_2 - 1\right)\right]^2 + \left[\max\left(0, \|\nabla_h D(\hat{x},h)\|_2 - 1\right)\right]^2\right\} \tag{9}$$

With the new penalty term (inside **Expression 9**'s brackets), the WGAN is now called WGAN-LP. We will apply this algorithm later in the thesis.

2.3. Text Encodings

Although this is not the focus of this thesis, we need to encode the captions into vectors in order to synthesize poses from them. The basis is word representations derived from large language corpora [21]. We will use a simple and fast word vector model - "fastText"[2] [21]. The basic idea is to represent a word as a bag of character n-grams (n=3~6) and 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 between a word's vector and its context words' vectors are high while the scalar products

[2] https://fasttext.cc/

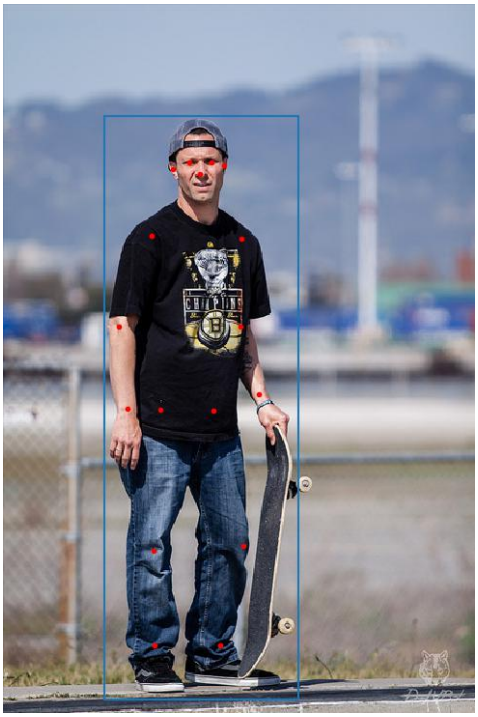
between the word and random words are low (details in [21]). The fastText model can do well in tasks such as word similarity and word analogies. Besides being simple and fast to train, this model has the advantage that it considers not only language structures but also internal structures of words because of using n-grams, and it can also properly represent words that have not appeared in the training corpora.
The above is about encodings of individual word but what we need is encodings of captions (sentences). In the fastText library, a sentence is represented by the average of normalized vectors of all its words. In this thesis, we will use a pre-trained model downloaded from fastText's website. This model is trained on the English Wikipedia. Unlike in [3], this model is not trained on our own data set, not related to what we are going to generate (human poses), so whether this neutral model works for our project remains to be seen.

3. Model

3.1. Data Set

The data set we are using is Microsoft COCO[3] (common objects in context) [22]. 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 shown in **Figure 1**) 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 1**, the red dots) are also given. There are in total 17 keypoints available: they are 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 1**, all 17 keypoints are visible but in most cases only some of them are visible. There are some poses in persons in the data set that have too few keypoint visible, and thus they do not provide much pose information. 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 1** are shown beside the image. We will use the keypoint information and captions from the COCO data set in our models.

[3] http://cocodataset.org/



A man is standing with a skateboard in his hand.
A man standing holding a skateboard vertical in one hand.
A man standing on the street holding a skateboard
A man with a hat on backwards stands with a skateboard.
A man is standing outside with his skateboard.

Figure 1 An example from the COCO data set. There is a person in this image. The blue rectangle is his bounding box and the red dots indicate where his keypoints are. On the right are the five captions for this image.

3.2. Data Preprocessing

The keypoint information given by the COCO data set cannot be used directly. We must first transform it into “images” so that they can be processed by our models based on DCGAN. We will process it this way:

Each keypoint corresponds to one channel of the image, so there are 17 channels. In each channel, if the corresponding keypoint is not visible in the image, all the values will be zero; if the keypoint is visible, the values will have a bell shaped profile, with the center in the place of the keypoint and the Figure 2maximum value one. More specifically, the value of the point (x, y) in this channel will be $e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}$, where (x_0, y_0) are the coordinates of the keypoint and σ controls the width of the bell shape. We call such kind of image representing pose keypoints a “heatmap”. The underlying principle of the heatmap is that, the higher the value of a point in the heatrmap is, the more likely the pose keypoint is to lie on this point. The height and width of the images processed in our models are all 64, so our heatmap size is $64 \times 64 \times 17$. However the images from the COCO data set have varying sizes. How we deal with this is that, with the help of the bounding box, we perform an affine transformation on the keypoints of a person, such that they fit in the center of our $64 \times 64 \times 17$ heatmap. And we set the width $\sigma = 2$. **Figure 2** is an illustration of an example heatmap. 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.

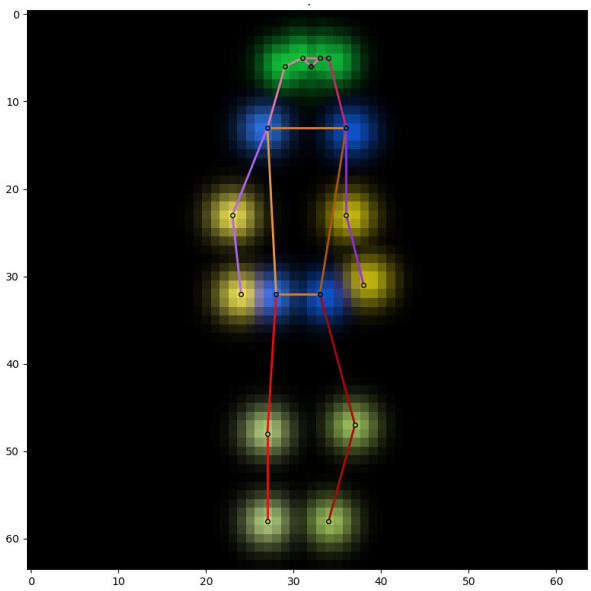


Figure 2 The heatmap of the pose of the person in **Figure 1**. We use different colors to show different channels and the brighter the higher values (value range $[0,1]$). Adjacent keypoints can be connected together to form a “skeleton” of this pose.

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.

3.3. Model Architectures

3.3.1. Unconditional Model

We first construct an unconditional model that merely synthesizes human poses (in the form of heatmaps) without considering any captions. It is just a plain unconditional DCGAN, with the size of images $64 \times 64 \times 17$. Our philosophy is starting with simple things then 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.

We add a few extra convolution layers to our GAN (compared to the DCGAN in [2]) so as to make it easier to incorporate caption encodings later on.

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 et al. 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. The IEEE Conference on Computer Vision and Pattern Recognition. 2015.
[12] Nair, Vinod et al. 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, Sergeyet al. Batch normalization: accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning. 2015.
[15] Kingma, Diederik P. et al. 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, Cédric. Optimal transport: old and new. Springer Science & Business Media. 2008.
[18] Tieleman, Tijmen et al. 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]