Pose Estimation of Anime/Manga Characters: A Case for Synthetic Data

Pramook Khungurn
Department of Computer Science
Cornell University
Ithaca, NY, USA
pramook@gmail.com

Derek Chou
Department of Computer Science
Cornell University
Ithaca, NY, USA
derek.chou4@gmail.com

ABSTRACT

2D articulated pose estimation is the task of locating the body joint positions of a human figure in an image. A pose estimator that works on anime/manga images could be an important component of an automatic system to create 3D animation from existing manga or anime, which could significantly the lower cost of media production. To create an accurate pose estimator, however, a sizable and high-quality dataset is needed, and such a dataset can be expensive to create.

To alleviate data scarcity, we propose using a database of 3D character models and poses to generate synthetic training data for 2D pose estimators of anime/manga characters based on convolutional neural networks (CNN). We demonstrate that a high-performing estimator can be obtained by pretraining a network on a large synthetic dataset and then fine-tuning it on a small dataset of drawings. We also show that the approach yields a pose estimator competitive with many previous works when applied to a photograph-based dataset, establishing our synthetic data's usefulness beyond the intended domain.

CCS Concepts

•**Computing methodologies** → **Scene understanding;**

Keywords

pose estimation; synthetic data

1. INTRODUCTION

Manga and anime are art forms with avid user communities. Anime are often adapted from manga source materials, and thus creating moving pictures that resemble the original still drawings is an important part of the creative process. In recent years, however, 3D computer graphics has become commonplace in anime production. While posing characters to drawings is much easier than redrawing the frames, it is still a time-consuming and costly process.

With recent advances in computer vision, we envision an automatic system that estimates 3D pose of characters from drawings. Such a system will greatly reduce the burden of animators and lower the cost of anime production. The system will also enable retrieving anime, manga, and drawings of characters based on their poses.

The first problem that the above system must be able to solve is *2D articulated pose estimation*: given an image of a human figure, locate the image-space positions of the body joints. When restricted to humans in photographs, the problem has been well studied by computer vision researchers. Recent research [35, 9] showed that solutions based on *convolutional neural networks* (CNNs) outperformed previous approaches. CNN-based solutions are generic and thus should also work on anime and manga images when trained on appropriate data. However, because CNNs have millions of parameters, they are prone to overfitting. A large dataset is required to train one that generalizes well.

Nevertheless, no previous datasets of drawings of human figures, let alone those done in anime/manga style, exist. Moreover, existing datasets for 2D pose estimation, even those based on photographs, are small and hard to create. While millions of training examples are available for the task of image classification [7], the largest dataset for 2D human pose estimation to date consists of only 40,000 images [1]. An alternative to building large datasets manually is to synthesize them with 3D computer graphics, which allows a large number of training examples to be generated cheaply with fine grained control on data variation.

In this paper, we explore using renderings of 3D character models as training data for CNN pose estimators of characters drawn in anime/manga style. We advocate a training regime where a simple CNN pose estimator is pre-trained with a large synthetic dataset and then fine-tuned with a small dataset based on drawings. While pre-training on an auxiliary dataset and then fine-tuning on a small dataset is not a new technique [11], we are the first to apply it to the 2D pose estimation task. We show that our approach produces a network that performs better than training on either dataset alone.

We also tested the approach on a photograph-based dataset and found that the resulting network was competitive with all but the most recent systems featuring more complicated CNN setups.

Our contribution is as follows:

- A dataset of *rendered 3D anime/manga characters models* annotated with 2D ground-truth joint positions. The set contains about 1,003,900 images in total, with 2000 images held out as the test set. These images are generated from 2,162 character models posed to motion frames from 6,570 motion clips.

- A dataset of *drawings in anime/manga style* annotated with human-labeled 2D joint positions. The set contains 2,000 images with 500 held out as the test set. The dataset was

MANPU '16, December 04 2016, Cancun, Mexico

ISBN 978-1-4503-4784-6...$15.00

DOI: http://dx.doi.org/10.1145/3011549.3011552

created to serve as a benchmark for the proposed training regime. To our knowledge, this is the first dataset of its kind.

- An evaluation of the effectiveness of synthetic data when used as training data for CNN pose estimators. Our conclusion is that, instead of creating a large dataset of drawings, it is sufficient to train a CNN on a large synthetic dataset and then fine-tune it on a small drawings dataset. We also investigate the effect of dataset size on the performance of CNN pose estimators.

2. PREVIOUS WORK

2D human pose estimation. The aim of this task to locate 2D positions of human body joints in an image. Prior to the recent emergence of CNNs in computer vision research, the most widely used approach was the pictorial structure model (PSM), popularized by the work of Felzenszwalb and Huttenlocher [10]. Many subsequent works improve upon PSMs by incorporating better models of its various components.

Toshev and Szegedy were one of the first to use CNNs for 2D pose estimation [35]. They employ a cascade of three CNN regressor in which the next CNN refines the joint positions produced by the previous one. Alternative architectures include one that takes both the whole image and a smaller patch of that image proposed by an object detector [9], one predicts belief maps instead of regress joint positions [26], one that incorporate recurrent networks [2], and one that improves CNN predictions iteratively [3]. Many other works combine CNNs with models of relative positions between parts [15, 34, 4, 5]

While we attempt to solve the same problem as the above works, we stress that this paper is not about a new CNN architecture for the problem. Instead, we study the effectiveness of synthetic data in improving performance of CNN-based pose estimators when real training data is scarce. We believe that our technique is *orthogonal* to the CNN architecture used in a sense that it should improve the performance of *any* such architecture.

Synthetic data for computer vision tasks. Researchers have used computer graphics to generate training data for various computer vision tasks including articulated pose estimation [29, 14, 25]. While other works use computer graphics to create substitutes for photographs, our work renders non-photorealistic images and use them as substitutes for drawings.

3. DATASETS

We now discuss the creation of two new datasets. Each training example in the datasets consists of two components: an image containing a humanoid character, and the pose of that character. The pose is the 2D positions of 21 joints as depicted in Figure 1(a). However, during training, we augmented the pose with middle points of the leg and arm bones as depicted in Figure 1(b), resulting in a total of 29 joints. Adding the middle points is a standard practice in training PSM-based models [4]. We found empirically that it results in slight increased accuracy of locating the original limb joints.

3.1 Synthetic Dataset

The synthetic dataset is generated by rendering 3D character models and composing the rendering onto a background image. We first collect the following three types of assets:

Character models. We take advantage of the 3D animation software MikuMikuDance (MMD) [13], which has a large community of users who publish their work on the Intenet We downloaded 2,162 character models created in anime or manga style.

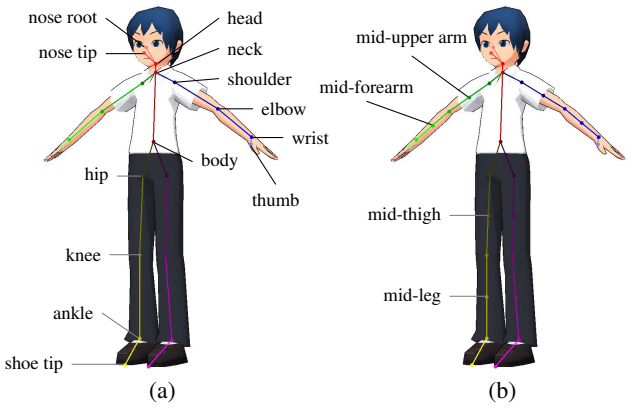


Figure 1: (a) The skeleton of all humanoid characters used in this paper. (b) The middle points added during training.

	Training	Validation	Test	Total
Characters	1,730	216	175*	2,121
Motions	5,256	657	657	6,570
Backgrounds	686	86	86	858
Examples	999,911	2,000	2,000	1,003,911

Table 1: Numbers of assets allocated and examples generated for the training, validation, and test sets. *We manually removed models of the same characters wearing similar outfits from the test asset pool to make it not too similar to the training asset pool.

Character poses. We downloaded 6,570 key-framed motions consisting of mostly dances, walking sequences, and martial arts moves. From each motion, we sample it at 24 frames per second to obtain poses and then eliminate duplicated poses within the same motion, resulting in 4,335,739 poses in total.

Background images. We collected 858 images from three sources. First, we used Google to search for images with key words "background", "anime background," "manga background," and "screentone." Second, we downloaded a number of free background images from Nico Nico Commons [21], a web site providing free art assets for Internet video creators. Third, we selected 241 images from the INRIAPerson dataset [6].

We then randomly split the assets into three sets so that we can use them to generate examples for training, validation, and testing. The ratio of source materials between the three sets is $8:1:1$. The numbers of assets allocated for each dataset are given in Table 1.

A training example is generated by rendering a random character taking a random pose under a random camera and directional light setting. We then alpha-composite the rendering onto a random background image to form the image part of the training example.

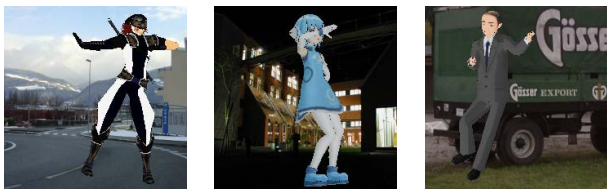


Figure 2: A sample of images generated by our example synthesis system.

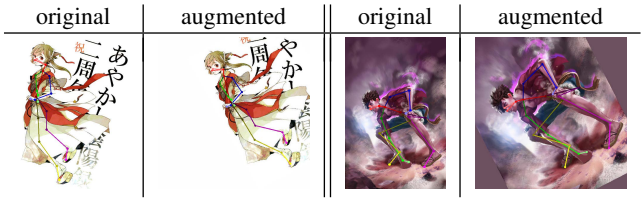


Figure 3: Two examples in the Drawings dataset, overliad with skeletons, before and after augmentation. The left original image is © Penko [23], and the right is © Song [31].

The associated 2D positions of the 29 joints can be extracted from the pose and the camera setting.

We use the above described above to generate $1,003,911$ examples, which are split into the training, validation, and training sets according to Table 1. The process was fully automatic and took about 10 days on a single desktop machine. It is, however, embarassingly parallelizable, and we ran it on three machines simulteneously. A sample of generated images are given in Figure 2.

3.2 Drawings Dataset

To evaluate the effectiveness of our approach on its target domain, we created a dataset of anime/manga drawings. We searched for images having tags "full body," "solo" and "rating:safe" from the image sharing site Danbooru [36] and initially downloaded $4,769$ images. Next, we set up a web application for marking the joint positions and recruited volunteers to help annotate the images. We asked them to guess positions for every joint that was not visible. After cleaning the data, we obtained $2,000$ training examples, each annotated with the positions of all the 21 joints. The data gathering process took roughly two weeks. A sample of the procured examples can be found in Figure 3.

We split the data so that the training set has $1,400$ examples, the validation 100 examples, and the test set 500 examples.

3.3 Data Augmentation

To mitigate overfitting, we augment the Drawings set by applying random rotations, scalings, and translations to the examples, and then filling the uncovered pixels with the average color of the border pixels of the original images. The output of the process is images having the same size as those the CNN takes as input, which we refer to as the "output size." Augmented examples are shown together with their originals in Figure 3.

We augmented the Drawings training set to $1,000,000$ images and the validation set to $1,000$ images. An original example is augmented to roughly the same number of examples. For the test set, we only uniformly resize the original image to the output size and fill uncovered pixels with border pixels' average.

4. TRAINING PROCEDURE

Our simple training procedure consists of two steps. We (1) pre-train a CNN with the synthetic dataset and then (2) fine-tune the pre-trained CNN with the Drawings dataset.

All the networks in this paper are based on the GoogLeNet architecture [33].[1] We modify the architecture as follows. First, we remove the two auxiliary classifiers connected to the outputs of the two lower Inception modules. Second, we replace the topmost inner product layer so that it outputs the 2D coordinates of the joints.

[1]We have also tried AlexNet [18] but found that they performed worse than GoogLeNet.

Third, for training, we use the *weighted Euclidean loss*:

$$\text{loss} = \frac{1}{2}\sum_{i=1}^{m} w_i \|\mathbf{x}_i - \mathbf{y}_i\|_2^2$$

where m is the number of joints ($m = 29$ in our case), w_i is the weight of the ith joint, $\mathbf{x}_i$ is the 2D position of the ith joint produced by the CNN, and $\mathbf{y}_i$ is the ground-truth position of the ith joint. Our setting is similar to that of the DeepPose architecture [35] except that we weight joints differently, and that we only use one pose estimator instead of three.

To increase accuracy on difficult joints, we weight joints that previous works have difficulty locating well 10 times higher than other joints. These include joints in the forearms and lower legs. These include the knees, mid-legs, ankles, shoe-tips, elbows, mid-forearms, wrists, and thumbs. This weighting scheme, however, slightly degrades performance on unweighted joints such as the head.

We implemented all CNNs with Caffe [16] and used the GoogLeNet model that is available from Caffe's Github repository [12]. Unless specified otherwise, we start all training from a GoogLeNet pre-trained for image classification.[2] We use the standard stochastic gradient descent algorithm and largely follow its slower training procedure in which learning reduces by a factor of 0.96 after $100,000$ iteration (1 epoch).

We used the starting learning rate of 10^{-3} when training on the synthetic dataset because we found that validation loss decays too slowly under smaller learning rates, and a higher learning rate such as 3×10^{-3} leads to numerical instability. By contrast, the starting learning rate is 10^{-4} when training on other smaller datasets because we often found through validation that training overfits if we use 10^{-3} and converges much more slowly if we use 10^{-5}.

For the number of iterations, we found that generally validation loss plateaued after $200,000$ iterations. As a result, we chose to let training completes after $400,000$ iterations. The batch size was 10 (i.e., 4 epochs on a data set with $1,000,000$ examples).

5. EVALUATION

In this section, we discuss experiments to assess the effectiveness of our synthetic data and training procedure. We first evaluate them on the Drawings dataset, their main target. We also evaluate them on the Leeds Sports Poses dataset, which is based on photographs, to show that they are useful beyond the domain of anime/manga images.

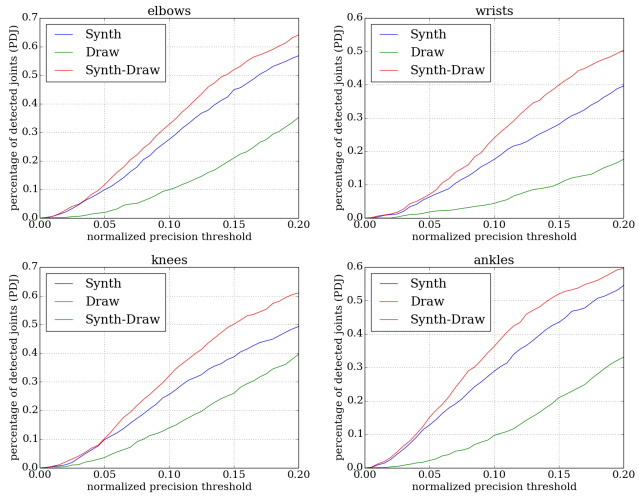
We evaluate performance of pose estimators using two metrics. The first is the *Percentage of Detected Joint* (PDJ) metric proposed by Sapp and Taskar [28]. We generally show PDJ graphs of joints that are known to be difficult to locate (i.e., elbows, wrists, knees, and ankles) as the fraction varies from 0 to 0.2, and we will also give a table of PDJ values at the fraction of 0.2 of other joints.

The second metric is the widely used *Percentage of Correct Parts* (PCP) metric [8]. We use the *strict* version of PCP where a bone is considered correctly located if two of its end joints are within a certain fraction of the bone's length to their ground truth positions. Like previous works, we evaluate PCP values with the fraction of 0.5. We only use this metric when we would like to compare with previous works.

5.1 Performance on Drawings Dataset

We trained the following three networks:

[2]Our training procedure thus never trains a network from scratch. Rather, it fine-tunes a network two times: the first time from an ImageNet classifier and the second time from the result of the first.



Methods	Body	Head	N.root	Elbows	Wrists	Knees	Ankles
SYNTH	65.0	85.4	78.4	56.8	39.6	49.3	54.5
DRAW	60.0	68.6	60.4	35.2	17.6	39.6	33.1
SYNTH-DRAW	**78.6**	**89.0**	**79.4**	**64.1**	**50.3**	**61.0**	**59.6**

Figure 4: PDJ results on the Drawings test set of the network trained on synthetic data (SYNTH), the network trained on Drawings data (DRAW), and SYNTH fine-tuned with Drawings data (SYNTH-DRAW).

- SYNTH: GoogLeNet trained on the synthetic dataset in Section 3.1.
- DRAW: GoogLeNet trained on the augmented Drawings dataset in Section 3.3.
- SYNTH-DRAW: the network SYNTH fine-tuned with the augmented Drawings dataset.

The SYNTH-DRAW network corresponds to our full training procedures, and the first two networks are used as baselines.

We evaluated the three networks on the test set of the Drawings dataset. The PDJ results are given in Figure 4, and samples of estimated poses can be found in Figure 5. Clearly, the network trained with our full procedure significantly outperforms the other two networks. Moreover, the network trained on synthetic data outperforms that trained on the Drawings data, indicating that the two types of data are sufficiently similar.

5.2 Performance on A Photograph Dataset

We also tested whether our approach is useful for pose estimation of humans in photographs. To do so, we used the extended *Leeds Sports Pose* (LSP) dataset [17] to train two more networks:

- LSP: GoogLeNet trained on the LSP dataset.
- SYNTH-LSP: the network SYNTH fine-tuned with the LSP dataset.

We augmented the LSP dataset using the process in Section 3.3 to expand the training set of 11,000 examples to one with 1,000,000 examples. The test set was also processed in the same way as indicated in Section 3.3. When training, we set to zero the weights of any joints that were indicated by the dataset as invisible.

We evaluated the SYNTH network along with the two new networks on the test set of the LSP dataset. The resulting PDJ curves are given in Figure 6, and PCP scores in Table 2.

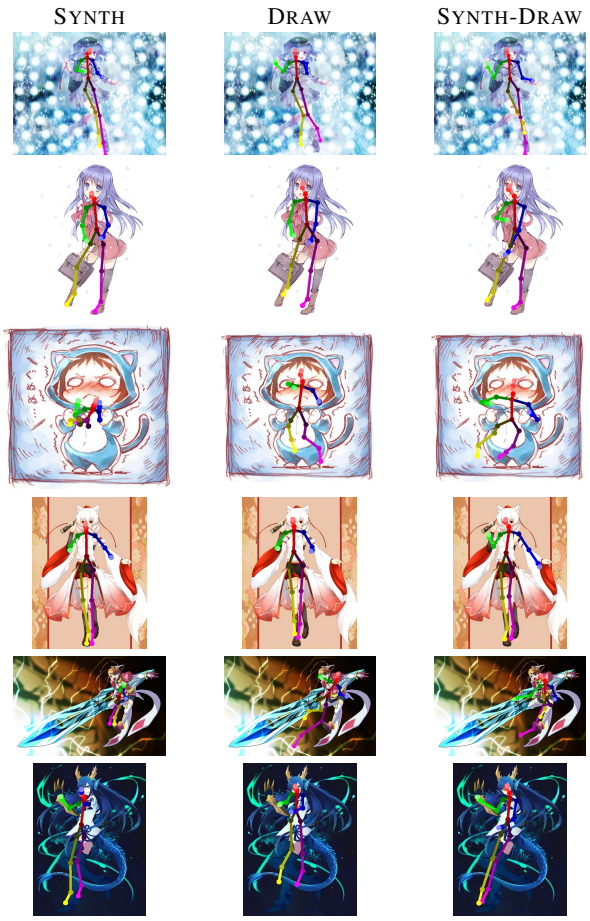


Figure 5: Visualization of poses estimated from random images in the drawings set by the SYNTH, DRAW, and SYNTH-DRAW networks in Section 5.1. From top to bottom, the images are © neme [20], © Suzushiro [32], © Saki no Shingetsu [27], © Shirakaba [30], © MURA [19], and © Niyasuke [22].

From the results, we can draw two conclusions. First is that the synthetic dataset is more similar to the Drawings dataset than the LSP set because the network SYNTH performs better on the Drawings set than on the LSP set. This result further validates the data generation process in Section 3.1.

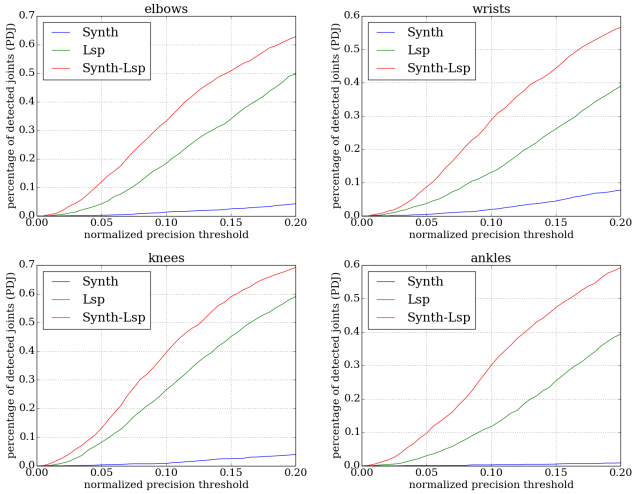
Second is that fine-tuning from a network that was pre-trained on the synthetic data improves performance even on a dataset based on photographs. This result shows that training on the synthetic data imparts to the network some knowledge about the 2D pose estimation problem not specific to the particular appearance of the synthetic data.

Moreover, our simple approach is competitive on locating difficult joints with previous approaches such as DeepPose [35] and those based on PSMs and non-CNN classifiers (the best being [24]). We believe that synthetic data can further improve performance of more elaborate systems such as [9] and [4].

5.3 Effect of Dataset Size

We saw that synthetic data leads to better performance. How large, then, should the synthetic data be? Can we get similar performance with a much smaller dataset? We performed an experiment to answer these questions.

We used our system in Section 3.1 to generate approximately



Methods	Hips	Neck	H.top	Elbows	Wrists	Knees	Ankles
SYNTH	12.8	0.6	—	4.3	7.8	4.0	0.9
LSP	71.6	82.2	60.5	49.6	38.9	59.1	39.4
SYNTH-LSP	**73.2**	**85.0**	**67.8**	**62.7**	**56.5**	**69.2**	**59.2**

Figure 6: PDJ results on the LSP test set of the network trained on synthetic data (SYNTH), the network trained on the LSP set (LSP), and SYNTH fine-tuned with the LSP set (SYNTH-LSP).

Methods	Torso	Head	U.arms	L.arms	U.legs	L.legs
SYNTH	—	—	0.4	0.5	5.7	3.0
LSP	97.7	56.3	52.1	26.8	78.0	65.9
SYNTH-LSP	97.2	61.6	63.8	48.1	82.2	75.8
Fan et al. [9]	**98**	85	**80**	63	**90**	**88**
Chu et al. [5]	95.4	**89.6**	77.0	**65.2**	87.6	83.2
Pishchulin et al. [24]	88.7	85.6	61.5	44.9	78.8	73.4
DeepPose [35]	—	—	56.0	38.0	77.0	71.0

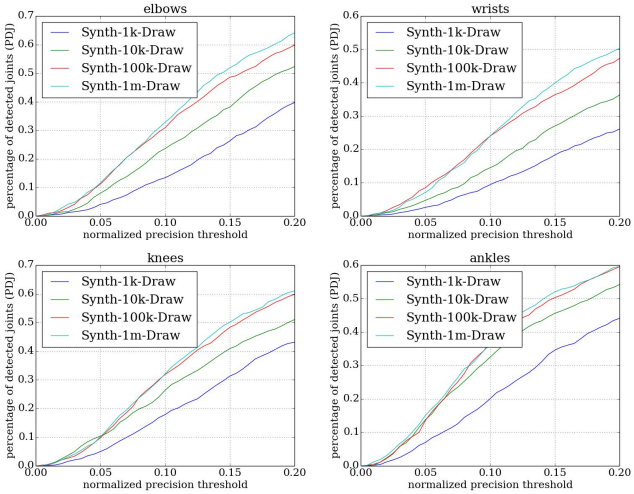
Table 2: PCP results on the LSP test set of the three networks in Figure 6 along with those of recent techniques. Note that we cannot compare with many recent works such as Rafi et al.[26] and Belagiannis and Zisserman [2] because they were evaluated with a different metric.

1,000, 10,000, and 100,000 synthetic examples and then augmented each set to 1,000,000 examples using the process in Section 3.3. Let us refer to the three datasets as the 1K, 10K, and 100K dataset, respectively. We then trained the following four networks:

- SYNTH-1K: GoogLeNet trained on the 1K set.
- SYNTH-10K: GoogLeNet trained on the 10K set.
- SYNTH-100K: GoogLeNet trained on the 100K set.
- SYNTH-1M: GoogLeNet trained on the synthetic data in Section 3.1 ($\approx 1,000,000$ examples). This network is the SYNTH network in Section 5.1.

We trained the first two networks with the starting learning rate of 10^{-4} as it is the learning rate we used when training on smaller datasets of comparable sizes (i.e., Drawings to 1K, and LSP to 10K). On the other hand, we trained the first two with the starting learning rate of 10^{-3}, which is normally used to train SYNTH-1M.

We then fine-tuned the networks on the Drawings set to obtained the SYNTH-1K-DRAW, SYNTH-10K-DRAW, and so on. We evaluated all networks on the Drawings test set. The results are given in Figure 7.



Methods	Body	Head	N.root	Elbows	Wrists	Knees	Ankles
SYNTH-1K-DRAW	64.8	77.0	69.4	39.7	26.1	43.1	44.1
SYNTH-10K-DRAW	68.4	86.0	76.8	52.3	36.3	51.0	54.2
SYNTH-100K-DRAW	74.8	88.2	79.2	59.8	47.3	59.8	59.5
SYNTH-1M-DRAW	**78.6**	**89.0**	**79.4**	**64.1**	**50.3**	**61.0**	**59.6**

Figure 7: PDJ results on the Drawings test set of the networks trained on training sets obtained from increasing number of synthetic examples as defined in Section 5.3.

It can be seen that more synthetic training examples yield better. However, SYNTH-100K-DRAW performance is very close to that of SYNTH-1M-DRAW. As such, a synthetic dataset with only a few hundred thousands examples may have been indistinguishable to one with a million.

6. CONCLUSION

We have proposed using synthetic data created from rendering 3D character models as training data for CNN pose estimators. We recommended pre-training on synthetic data and then fine-tuning on a smaller, real dataset. While our main target domain is anime and manga images, we have demonstrated that this procedure also worked well for photographs of humans too. However, while larger synthetic data leads to increased performance, increasing the size beyond a few hundred thousands examples is not likely to yield significant performance gain.

This work is but the first step in creating an automatic 3D pose estimator we envision. Our approach, coupled with off-the-shelf CNN architecture, is competitive with a number of previous works, but still is not reliable enough for such a system. Because the approach should be orthogonal to the underlying CNN implementation, one potential future work is to see whether our synthetic data can be used to boost performance of the best CNN setup such as that of Fan et al. [9].

Acknowledgments. We thank Kavita Bala and Steve Marschner for their advice and supervision. We are grateful for the volunteers' help in annotating the dataset. We are also indebted to the illustrators, 3D modelers, and animators for making their works available on the Internet. Kavita Bala would like to thank Adobe for their donation.

7. REFERENCES

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art

analysis. In *CVPR*, 2014.

[2] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. https://arxiv.org/pdf/1605.02914v1.pdf, 2016. Accessed: 2016-09-22.

[3] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. 2015.

[4] X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[5] X. Chu, W. Ouyang, H. Li, and X. Wang. Structured feature learning for pose estimation. In *CVPR*, 2016.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[8] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari. 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. *International Journal of Computer Vision*, 99(2):190–214, 2012.

[9] X. Fan, K. Zheng, Y. Lin, and S. Wang. Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation. *arXiv preprint arXiv:1504.07159*, 2015.

[10] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[12] S. Guadarrama. bvlc_googlenet. https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet, 2015. Accessed: 2015-05-27.

[13] Y. Higuchi. VPVP. http://www.geocities.jp/higuchuu4/index_e.htm, 2015. Accessed: 2015-05-20.

[14] A. Jain, T. Thormählen, H.-P. Seidel, and C. Theobalt. Moviereshape: Tracking and reshaping of humans in videos. *ACM Trans. Graph.*, 29(6):148:1–148:10, Dec. 2010.

[15] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning human pose estimation features with convolutional networks. In *International Conference on Learning Representations (ICLR)*, April 2014.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[17] S. Johnson and M. Everingham. Learning effective human pose estimation from inaccurate annotation. In *CVPR*, 2011.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[19] MURA. Mai-hime kanren matome. http://www.pixiv.net/member_illust.php?mode=manga_big&illust_id=29697555&page=2, 2012. Accessed: 2015-09-15.

[20] neme. Mudai document. http://sugarpot.digi2.jp/touhou/nitorika-do.html, 2011. Accessed: 2015-09-15.

[21] niwango, inc. Nico Nico Commons. http://commons.nicovideo.jp, 2015. Accessed: 2015-05-20.

[22] Niyasuke. Karin. http://www.pixiv.net/member_illust.php?mode=medium&illust_id=45738647, 2012. Accessed: 2015-09-15.

[23] Penko. Iwai! http://www.pixiv.net/member_illust.php?mode=medium&illust_id=43897212, 2014. Accessed: 2015-09-15.

[24] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Strong appearance and expressive spatial models for human pose estimation. In *ICCV*, 2013.

[25] L. Pishchulin, A. Jain, C. Wojek, M. Andriluka, T. Thormahlen, and B. Schiele. Learning people detection models from few training samples. In *CVPR*, 2011.

[26] U. Rafi, I. Kostrikov, J. Gall, and B. Leibe. An efficient convolutional network fof human pose estimation. http://www.iai.uni-bonn.de/~gall/download/jgall_posecnn_bmvc16, 2016. Accessed: 2016-09-22.

[27] Saki no Shingetsu. Pajama mashiro. http://www.pixiv.net/member_illust.php?mode=medium&illust_id=42443538, 2014. Accessed: 2015-09-15.

[28] B. Sapp and B. Taskar. Modec: Multimodal decomposable models for human pose estimation. In *CVPR*, 2013.

[29] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, 2003.

[30] P. Shirakaba. Oyama no telegnosis. http://www.pixiv.net/member_illust.php?mode=medium&illust_id=43832960, 2014. Accessed: 2015-09-15.

[31] Y. Song. Luffy. http://young-street.deviantart.com/art/Luffy-462966102, 2014. Accessed: 2015-09-15.

[32] K. Suzushiro. Saikin no rakugaki log. http://www.pixiv.net/member_illust.php?mode=manga_big&illust_id=43505100&page=20, 2014. Accessed: 2015-09-15.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[34] J. J. Tompson, A. Jain, Y. Lecun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[35] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659, 2013.

[36] A. Yi. Danbooru. http://danbooru.donmai.us/posts?tags=rating:safe, 2015. Accessed: 2015-05-26.