Self-supervised Learning of Interpretable Keypoints from Unlabelled Videos

Tomas Jakab
Visual Geometry Group
University of Oxford
tomj@robots.ox.ac.uk

Ankush Gupta
DeepMind, London
ankushgupta@google.com

Hakan Bilen
School of Informatics
University of Edinburgh
hbilen@ed.ac.uk

Andrea Vedaldi
Visual Geometry Group
University of Oxford
vedaldi@robots.ox.ac.uk

Appendix

This supplementary material provides further technical details, illustrations and analysis. We provide detailed quantitative evaluation on Human3.6M dataset (appendix A), extended versions of our qualitative results on factorization of appearance and geometry (appendix B), facial landmarks detection (appendix C), human pose estimation (appendix D), and cat head landmarks detection (appendix E). Finally, we give further implementation details in appendix F.

A. Human3.6M detailed results

We report the performance for each activity of the Human3.6M test set in table 1. We evaluated the performance on every 60th frame of the video sequences.

Method	all	wait	pose	greet	direct	discuss	walk	eat	phone	purchase	sit	sit down	smoke	take photo	walk dog	walk together
fully supervised																
Newell *et al*. [6]	19.52	15.53	13.88	17.14	15.81	19.55	13.74	15.33	18.81	19.88	25.85	39.07	19.40	22.24	21.58	14.96
self-supervised + supervised post-processing																
Jakab *et al*. [2]	19.12	16.63	15.01	16.68	14.73	15.69	17.74	16.53	23.27	17.35	24.66	33.14	20.31	20.96	**17.77**	16.31
self-supervised (no post-processing)																
Ours *3DHP prior*	18.94	15.33	14.37	16.08	15.90	17.24	14.51	17.30	19.66	17.39	22.79	30.84	18.50	24.21	23.77	16.16
Ours *H3.6M prior*	**14.46**	**11.40**	**10.39**	**11.85**	**11.26**	**13.72**	**11.85**	**12.02**	**14.42**	**12.90**	**17.01**	**25.71**	**14.35**	**18.67**	19.42	**11.90**

Table 1. **Human landmark detection (full H3.6M).** Comparison on Human3.6M test set with a supervised baseline Newell *et al*. [6], and a self-supervised method [2]. We report the MSE in pixels [1] for each activity. We highlight the minimum error across all models in bold.

B. Appearance and geometry factorization

Figure 1. **Factorization of appearance and geometry.** We supply different identities for *style* and *target* input images. *Reconstructed* image inherits appearance from the *style* image and geometry from the *target* image. **[top]:** human pose samples from Human3.6M. **[bottom]:** face samples from VoxCeleb2.

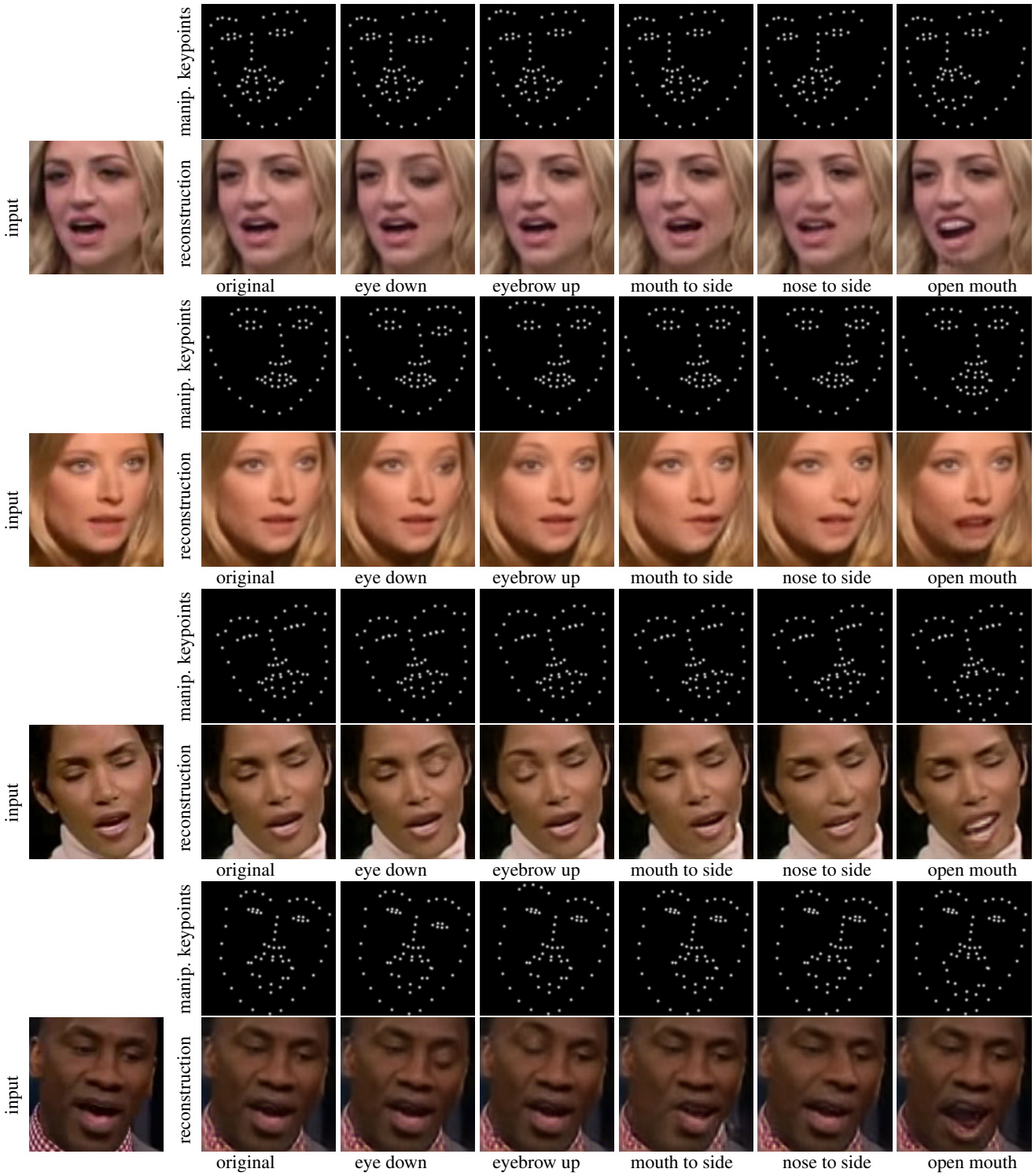

Figure 2. **Image editing using detected landmarks.** We show fine-grained control over the generated image by manipulating the coordinates of detected landmarks (*manip. keypoints*). For example, we pick landmarks corresponding to an eye and move them down [second column], or open the mouth [last column] (note, the generator fills in the teeth absent in the input images). The resulting changes are localized and allow for fine-grained control. Apart from demonstrating successful disentanglement of appearance and geometry, this also suggests that the model assigns correct semantics to the detected landmarks.

C. Facial landmarks detections

Figure 3. **Facial landmark detections on 300-W.** Randomly sampled predictions from 300-W test set. The model was trained with unlabelled images from VoxCeleb2 face videos dataset and unpaired landmarks sampled from MultiPIE dataset, hence shows significant generalization. Green markers denote our detections, blue correspond to the ground truth.

D. Human pose estimation

D.1. Pose detection on Human3.6M

Figure 4. **Pose estimation on Human3.6M.** Randomly sampled results from Human3.6M test set. The model is trained with unpaired images and skeletons from Human3.6M.

D.2. Pose detection on Simplified Human3.6M

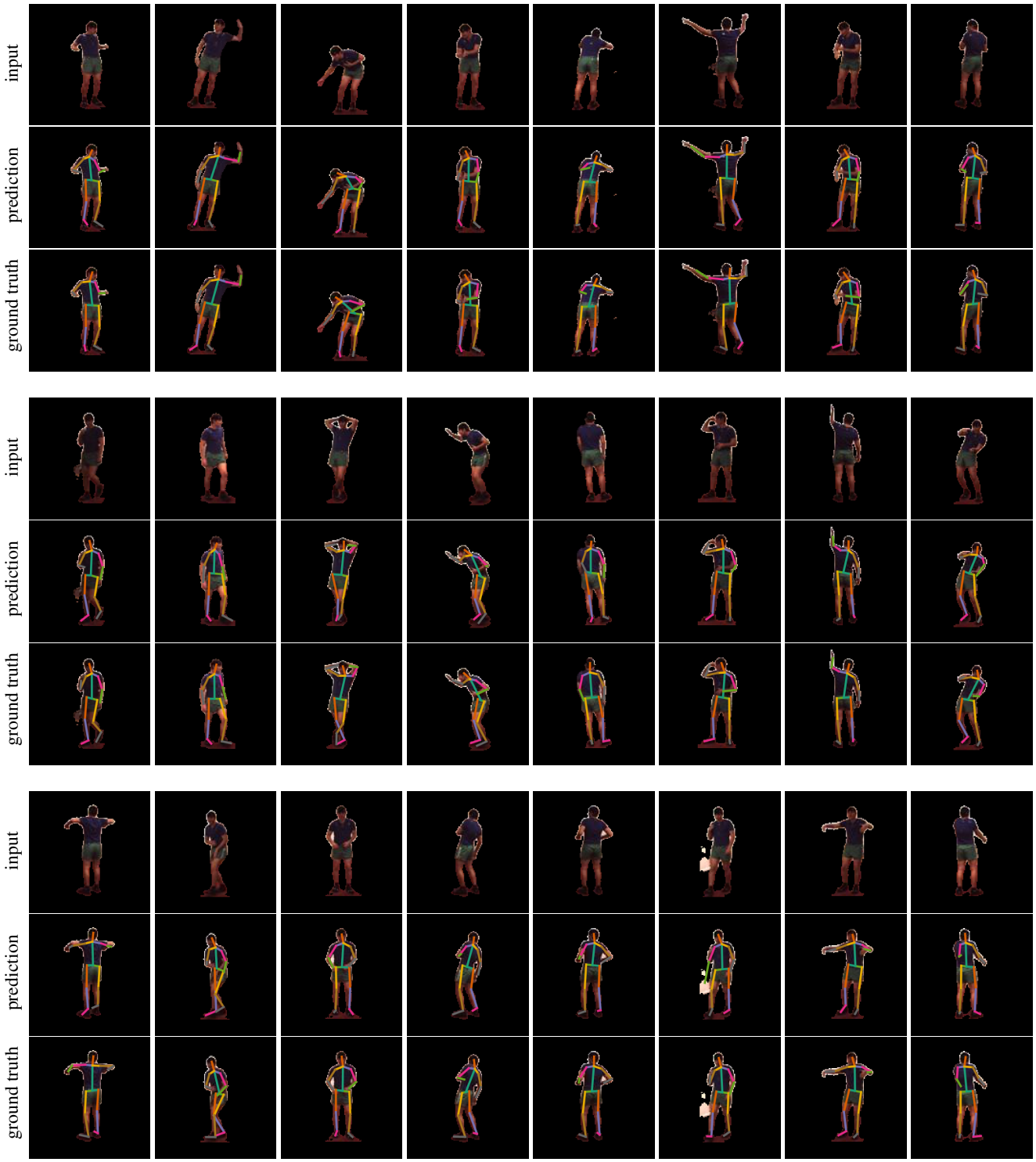

Figure 5. **Pose estimation on the Simplified Human3.6M.** Randomly sampled results from the Simplified Human3.6M test set. The model is trained with unpaired images and skeletons from Simplified Human3.6M.

E. Landmarks detection on Cat Heads

Figure 6. **Landmark detections on Cat Head.** Randomly sampled predictions on Cat Head test set.

F. Implementation details

F.1. Training details

The auto-encoder functions Φ and Ψ and the discriminator D are trained by optimizing the overall objective in eq. (5) of the main paper while setting $\lambda = 10$ (η is pre-trained using unpaired landmarks as detailed below). We use the Adam optimiser [4] with a learning rate of $2 \cdot 10^{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The batch size is set to 16 and the norm of the gradients is clipped to 1.0 for stability.

F.2. Pre-training the function η

The network η mapping the skeleton image $\boldsymbol{y}$ to its corresponding keypoint locations $\boldsymbol{p}$ is pre-trained before optimizing the overall objective (eq. (5) of the main paper). This is done by using the unpaired pose samples $\{\bar{\boldsymbol{p}}_j\}_{j=1}^M$ and by optimizing the loss $\frac{1}{M}\sum_{j=1}^M \mathcal{L}(\eta|\bar{\boldsymbol{p}}_j)$ where

$$\mathcal{L}(\eta|\bar{\boldsymbol{p}}) = \|\eta \circ \beta(\bar{\boldsymbol{p}}) - \bar{\boldsymbol{p}}\|^2 \tag{1}$$

is a simple ℓ^2 regression loss.

During the optimization of the overall objective, the function η is further fine-tuned by minimizing the same loss plus eq. (1) an additional term $\mathcal{L}(\eta|\boldsymbol{y}) = \lambda'\|\beta \circ \eta(\boldsymbol{y}) - \boldsymbol{y}\|^2$, where $\boldsymbol{y}$ is a reconstructed pose (see fig. 2 of the main paper). The latter ensures that network η works for poses that appear in the videos but not necessarily in the pose prior. The two terms are balanced by the coefficient λ'. After fine-tuning η, we noticed that it loses some of its ability to distinguish between frontal and dorsal views of human body (which is fairly ambiguous given only a skeleton image as input). We correct its predictions by using the pre-trained version of η at eq. (1) to determine the orientation of human body.

The function η is designed as a neural network that converts the skeleton image $\boldsymbol{y}$ into K heatmaps. The locations of keypoints are further obtained as in [2] by converting each heatmap into a 2D probability distribution. The expectation of this probability distribution corresponds to the location of the keypoints. The spatial coordinates are normalised to the $[-1, 1]$ range and we set $\gamma = \frac{1}{0.04}$ in eq. (2) of the main paper. The function is learned by minimizing the loss introduced above with $\lambda' = 0.1$.

F.3. Note on a second cycle constraint and discriminator

Standard CycleGAN [8] enforces two cycle constraints $\Psi \circ \Phi(\boldsymbol{x}) \approx \boldsymbol{x}$ and $\Phi \circ \Psi(\boldsymbol{y}) \approx \boldsymbol{y}$. Our model implements a conditional version of the first, while the second can be written as $\Phi(\Psi(\bar{\boldsymbol{y}}, \boldsymbol{x}')) \approx \bar{\boldsymbol{y}}$. CycleGAN also utilizes a discriminator $D_{\mathcal{X}}$ on images $\hat{\Psi}(\boldsymbol{y})$ generated from skeletons to match their distribution to images $\boldsymbol{x}$; the same discriminator applies here, except that images are generated conditionally $\Psi(\bar{\boldsymbol{y}}, \boldsymbol{x}')$ and they are tested against the distribution of images $\boldsymbol{x}$ from the same video, so $\mathcal{D}_{\mathcal{X}}(\Psi(\bar{\boldsymbol{y}}, \boldsymbol{x}'), \boldsymbol{x}')$ is conditional too. Our ablation study shows that the additional cycle constraint and discriminator leads to worse performance, so we do not include them in our final version of the model.

F.4. Architectures

Figures 7 to 11 provide detailed descriptions of network architectures used in experiments.

Type	Kernel	Stride	Output channels	Output size	Norm.	Activation
Input $\boldsymbol{x}$	-	-	3	128	-	-
Conv	7	1	32	128	Batch	ReLU
Conv	3	1	32	128	Batch	ReLU
Conv	3	2	64	64	Batch	ReLU
Conv	3	1	64	64	Batch	ReLU
Conv	3	2	128	32	Batch	ReLU
Conv	3	1	128	32	Batch	ReLU
Conv	3	2	256	16	Batch	ReLU
Conv	3	1	256	16	Batch	ReLU
Conv	1	1	256	16	None	None
Conv	3	1	256	16	Batch	ReLU
Conv	3	1	256	16	Batch	ReLU
Bilinear upsampl.	-	-	128	32	-	-
Conv	3	1	128	32	Batch	ReLU
Conv	3	1	128	32	Batch	ReLU
Bilinear upsampl.	-	-	64	64	-	-
Conv	3	1	64	64	Batch	ReLU
Conv	3	1	64	64	Batch	ReLU
Bilinear upsampl.	-	-	32	128	-	-
Conv	3	1	32	128	Batch	ReLU
Conv	3	1	32	128	Batch	ReLU
Conv	3	1	1	128	None	None

Figure 7. **Image encoder Φ.** The network is based of the encoder and decoder network from [2]. Arrows on the side denote skip connections that are concatenated to the other input.

Type	Kernel	Stride	Output ch.	Output size	Norm	Activ
Input $\boldsymbol{x}'$	-	-	3	128	-	-
Conv	7	1	32	128	Batch	ReLU
Conv	3	1	32	128	Batch	ReLU
Conv	3	2	64	64	Batch	ReLU
Conv	3	1	64	64	Batch	ReLU
Conv	3	2	128	32	Batch	ReLU
Conv	3	1	128	32	Batch	ReLU
Conv	3	2	256	16	Batch	ReLU
Conv	3	1	256	16	Batch	ReLU
Conv	1	1	256	16	None	None

Type	Kernel	Stride	Output ch.	Output size	Norm	Activ
Input $\boldsymbol{y}^*$	-	-	1	128	-	-
Conv	7	1	32	128	Batch	ReLU
Conv	3	1	32	128	Batch	ReLU
Conv	3	2	64	64	Batch	ReLU
Conv	3	1	64	64	Batch	ReLU
Conv	3	2	128	32	Batch	ReLU
Conv	3	1	128	32	Batch	ReLU
Conv	3	2	256	16	Batch	ReLU
Conv	3	1	256	16	Batch	ReLU
Conv	1	1	256	16	None	None

Type	Kernel	Stride	Output ch.	Output size	Norm.	Activ.
Concat	-	-	512	16	-	-
Conv	3	1	256	16	Batch	ReLU
Conv	3	1	256	16	Batch	ReLU
Bi. upsampl.	-	-	128	32	-	-
Conv	3	1	128	32	Batch	ReLU
Conv	3	1	128	32	Batch	ReLU
Bi. upsampl.	-	-	64	64	-	-
Conv	3	1	64	64	Batch	ReLU
Conv	3	1	64	64	Batch	ReLU
Bi. upsampl.	-	-	32	128	-	-
Conv	3	1	32	128	Batch	ReLU
Conv	3	1	32	128	None	None

Figure 8. **Image decoder Ψ.** Image encoder first processes the conditioning image $\boldsymbol{x}'$ and the skeleton $\boldsymbol{y}^*$ in two separate independent branches before it concatenates them into a single stream. The design follows [2].

Type	Kernel size	Stride	Output channels	Output size	Norm.	Activation
Input $\boldsymbol{y}$	-	-	3	128	-	-
Conv	7	1	32	128	Batch	ReLU
Conv	3	1	32	128	Batch	ReLU
Conv	3	2	64	64	Batch	ReLU
Conv	3	1	64	64	Batch	ReLU
Conv	3	2	128	32	Batch	ReLU
Conv	3	1	128	32	Batch	ReLU
Conv	3	2	256	16	Batch	ReLU
Conv	3	1	256	16	Batch	ReLU
Conv	1	1	*n keypoints*	16	None	None

Figure 9. **Skeleton encoder η.** The architecture is based on the encoder from [2]. The last layer has as many output channels as the number of keypoints to predict.

Type	Kernel size	Stride	Output channels	Output size	Norm.	Activation
Input ($\tilde{\boldsymbol{y}}$ or $\boldsymbol{y}$)	-	-	1	128	-	-
Conv	4	2	64	64	Instance	LReLU
Conv	4	2	128	32	Instance	LReLU
Conv	4	2	256	16	Instance	LReLU
Conv	4	1	512	15	Instance	LReLU
Conv	4	1	1	14	None	None

Figure 10. **Skeleton discriminator $D_{\mathcal{Y}}$.** The architecture follows [8]. LReLU stands for Leaky Rectified Linear Unit [5] that is used with 0.2 negative slope. Instance normalization [7] is used before every activation. We use three such discriminators each for a different scale of the input image. We resize the input images by 1, $\frac{1}{2}$, and $\frac{1}{4}$ factors.

Type	Kernel	Stride	Output ch.	Output size	Norm.	Activ.
Input ($\Psi(\tilde{\boldsymbol{y}}, \boldsymbol{x}')$ or $\boldsymbol{x}$)	-	-	3	128	-	-
Conv	3	1	16	128	Instance	LReLU
Conv	3	1	32	128	Instance	LReLU
Conv	3	1	32	128	Instance	LReLU
Avg. pool.	2	2	32	64	-	-
Conv	3	1	64	64	Instance	LReLU
Conv	3	1	64	64	Instance	LReLU
Avg. pool.	2	2	64	32	-	-
Conv	3	1	128	32	Instance	LReLU
Conv	1	1	128	32	Instance	LReLU
Avg. pool.	2	2	128	16	-	-
Conv	3	1	256	16	Instance	LReLU
Conv	1	1	256	16	Instance	LReLU
Avg. pool.	2	2	256	8	-	-

weight sharing

Type	Kernel	Stride	Output ch.	Output size	Norm.	Activ.
Input $\boldsymbol{x}'$	-	-	3	128	-	-
Conv	3	1	16	128	Instance	LReLU
Conv	3	1	32	128	Instance	LReLU
Conv	3	1	32	128	Instance	LReLU
Avg. pool.	2	2	32	64	-	-
Conv	3	1	64	64	Instance	LReLU
Conv	3	1	64	64	Instance	LReLU
Avg. pool.	2	2	64	32	-	-
Conv	3	1	128	32	Instance	LReLU
Conv	1	1	128	32	Instance	LReLU
Avg. pool.	2	2	128	16	-	-
Conv	3	1	256	16	Instance	LReLU
Conv	1	1	256	16	Instance	LReLU
Avg. pool.	2	2	256	8	-	-

Type	Kernel size	Stride	Output channels	Output size	Norm.	Activation
Concat	-	-	512	8	-	-
Conv	1	1	512	8	Instance	LReLU
Conv	3	1	512	8	Instance	LReLU
Conv	3	1	512	8	Instance	LReLU
Avg. pool.	2	2	512	4	-	-
Conv	3	1	512	4	Instance	LReLU
Conv	1	1	512	4	Instance	LReLU
Conv	4	1	512	1	Instance	LReLU
Conv	1	1	1	1	None	None

Figure 11. **Conditional image discriminator $D_{\mathcal{X}}$.** Conditional image discriminator starts with a Siamese architecture until the two streams are concatenated. When the version without conditioning is required, the second branch in the Siamese part is simply omitted. LReLU stands for Leaky Rectified Linear Unit [5]. We set the negative slope to 0.2. Every activation is preceded by instance normalization [7]. The architecture is loosely based on [3].

References

[1] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *TPAMI*, 36(7):1325–1339, jul 2014. 1

[2] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Proc. NIPS*, 2018. 1, 8, 9, 10

[3] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 10

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[5] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013. 10

[6] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proc. ECCV*, 2016. 1

[7] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 10

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. CVPR*, 2018. 8, 10