Appendices

Unsupervised Discovery of Object Landmarks as Structural Representations

Yuting Zhang[1], Yijie Guo[1], Yixin Jin[1], Yijun Luo[1], Zhiyuan He[1], Honglak Lee[2,1]

[1]University of Michigan, Ann Arbor [2]Google Brain

{yutingzh, guoyijie, jinyixin, lyjtour, zhiyuan, honglak}@umich.edu honglak@google.com

Code and results, including images and videos, are available at the project web page:
http://ytzhang.net/projects/lmdis-rep

Supplementary videos referred to in the appendices can be found in the zip file of the supplementary materials or obtained at http://ytzhang.net/files/lmdis-rep/supp-videos.tar.gz

Contents

A. More details and results on face manipulation using unsupervised landmarks

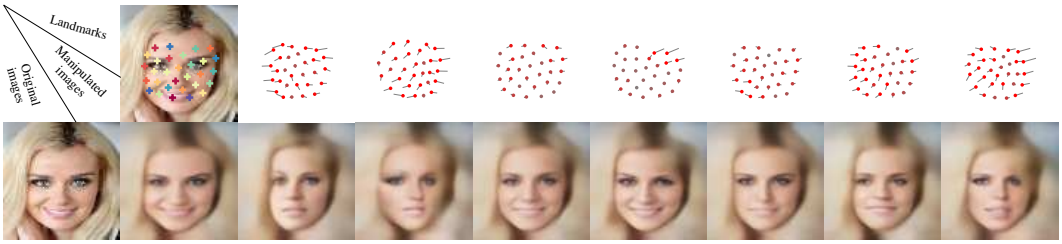
The discovered landmarks constitute the explicitly structural part of the image representation learned by our model. They provide an interface for humans to manipulate the image representation intuitively. Our decoding module can generate realistic facial images using the landmark descriptors extracted from a given image and different sets of landmarks.

In addition to the results shown in the main paper, we provide more qualitative results for unsupervised landmark-based face manipulation in this section. We train models of 10, 20, and 30 landmarks. To evaluate our method on many target landmarks, we take the landmarks discovered from other images and the interpolation/extrapolation between the landmarks discovered on two images as the targets.

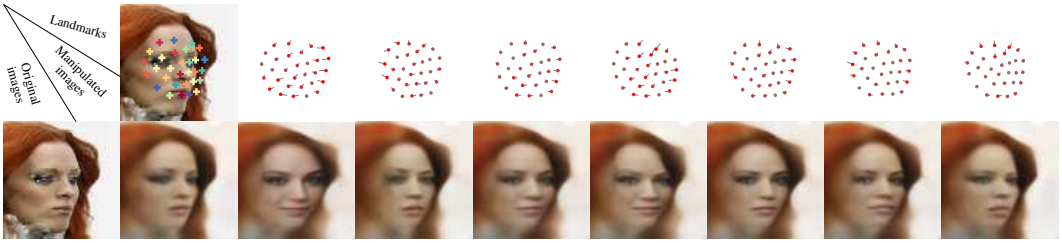
In this section, we show results for our 30-landmark model (results for our 10,20-landmark model are available as supplementary videos). Figure 12 and 13 show results for manipulating all 30 landmarks. Figure 14 and 15 show results for manipulating the 3 landmarks at the mouth. Figure 16 and 17 show results for manipulating the 5 landmarks at the mouth and jaw. Videos are available in the following folders for gradually morphing the landmarks from their original coordinates to the target by linear interpolation.

- 30-landmark models:
  `videos/face.30landmark-model.manipulate-{all|mouth|mouthext}-landmarks`
- 10,20-landmark models:
  `videos/face.{10|20}landmark-model.manipulate-{all|mouth}-landmarks`

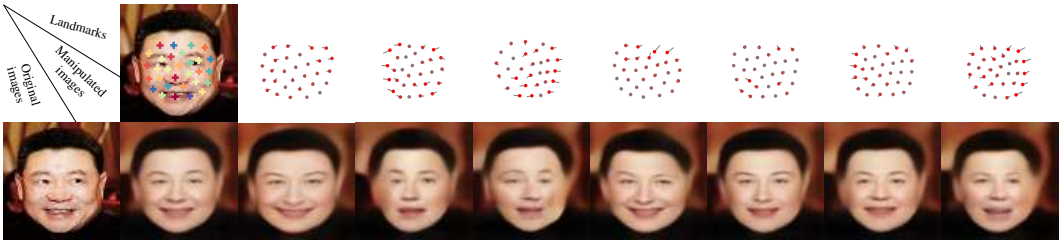
See next page for the figure.



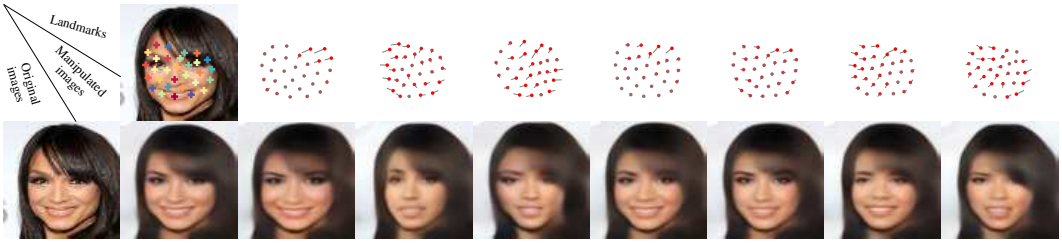
(a) Video: videos/face.30landmark-model.manipulate-all-landmarks/01.mp4



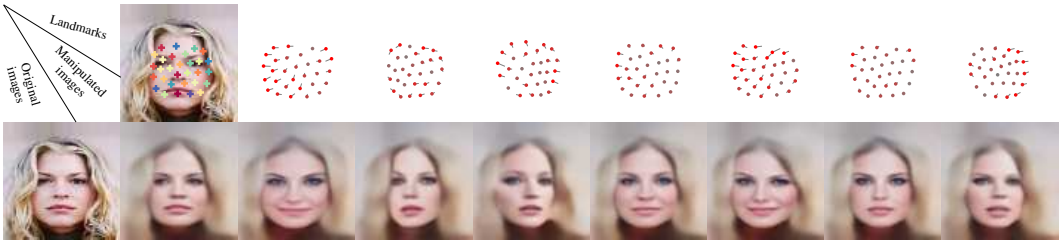
(b) Video: videos/face.30landmark-model.manipulate-all-landmarks/02.mp4



(c) Video: videos/face.30landmark-model.manipulate-all-landmarks/03.mp4

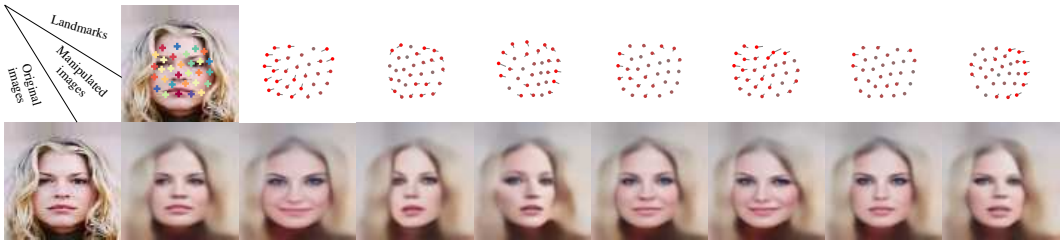


(d) Video: videos/face.30landmark-model.manipulate-all-landmarks/04.mp4

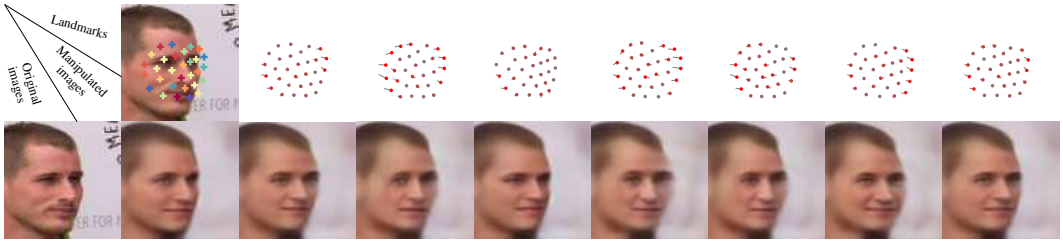


(e) Video: videos/face.30landmark-model.manipulate-all-landmarks/06.mp4

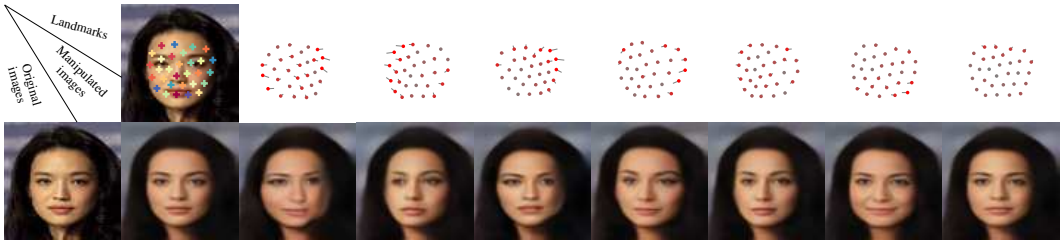
Figure 12: Face manipulation by modifying all 30 discovered landmarks on the MAFL testing set. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at videos/face.30landmark-model.manipulate-all-landmarks for the morphing process.



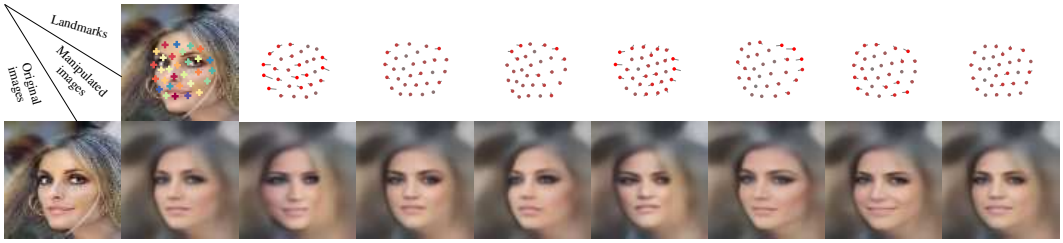
(a) Video: videos/face.30landmark-model.manipulate-all-landmarks/06.mp4



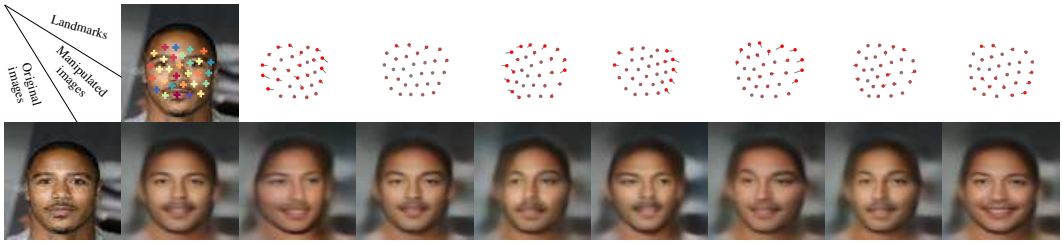
(b) Video: videos/face.30landmark-model.manipulate-all-landmarks/07.mp4



(c) Video: videos/face.30landmark-model.manipulate-all-landmarks/08.mp4

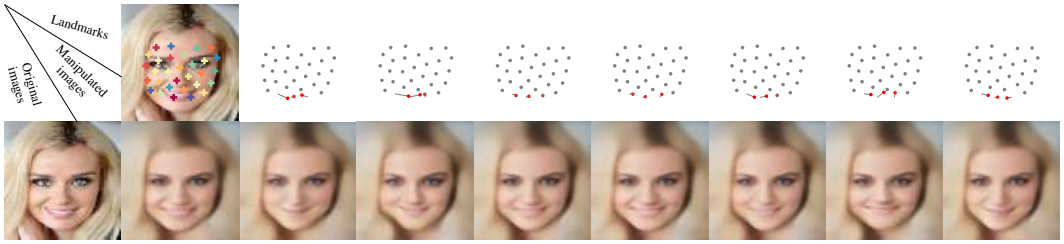


(d) Video: videos/face.30landmark-model.manipulate-all-landmarks/09.mp4

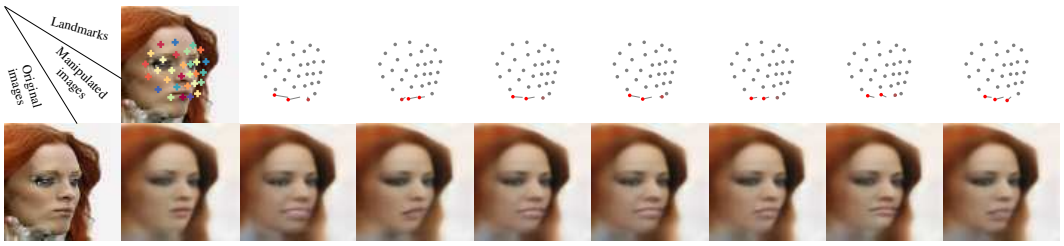


(e) Video: videos/face.30landmark-model.manipulate-all-landmarks/10.mp4

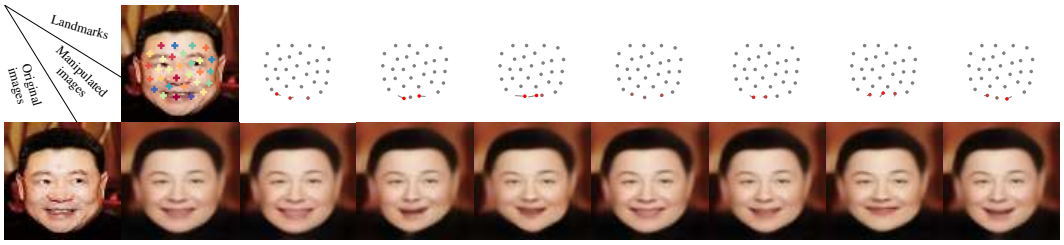
Figure 13: Continued from Figure 12. Face manipulation by modifying all 30 discovered landmarks on the MAFL testing set. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at videos/face.30landmark-model.manipulate-all-landmarks for the morphing process.



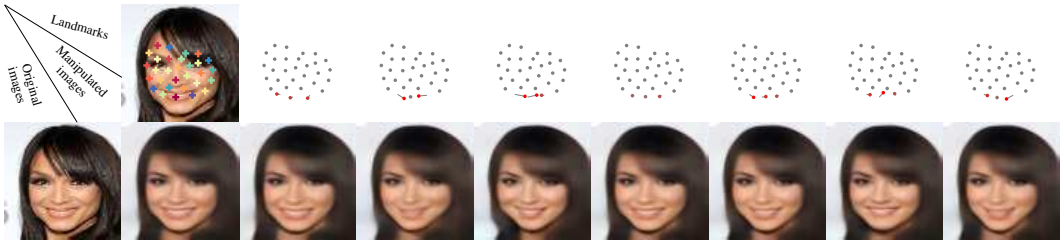
(a) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/01.mp4



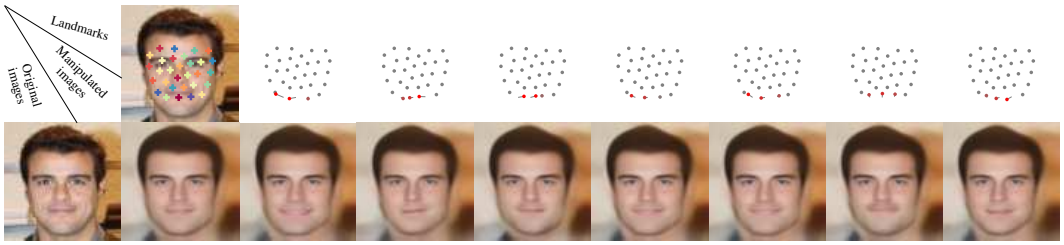
(b) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/02.mp4



(c) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/03.mp4

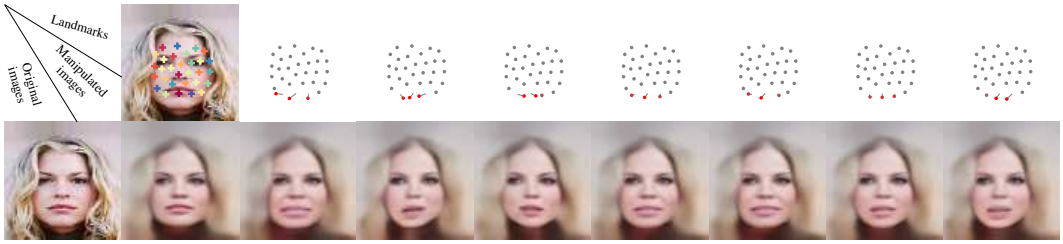


(d) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/04.mp4

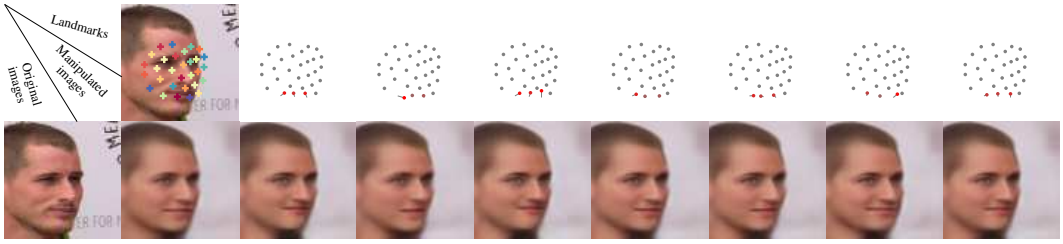


(e) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/05.mp4

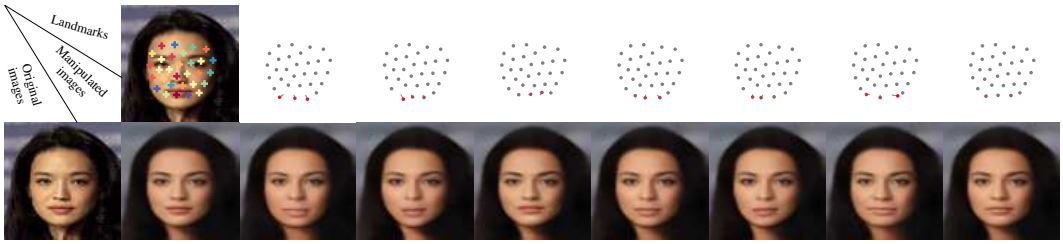
Figure 14: Face manipulation by modifying 3 discovered mouth landmarks on the MAFL testing set. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at videos/face.30landmark-model.manipulate-mouth-landmarks for the morphing process.



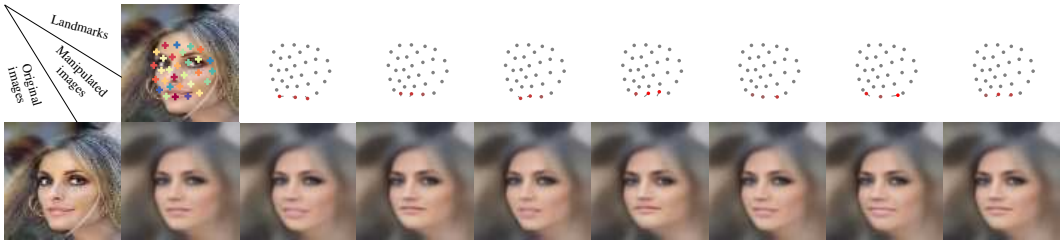
(a) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/06.mp4



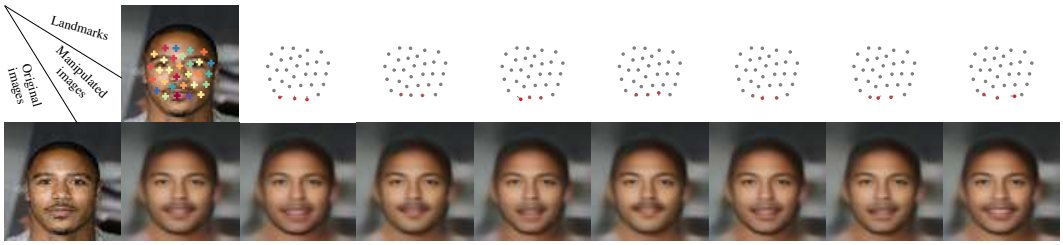
(b) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/07.mp4



(c) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/08.mp4

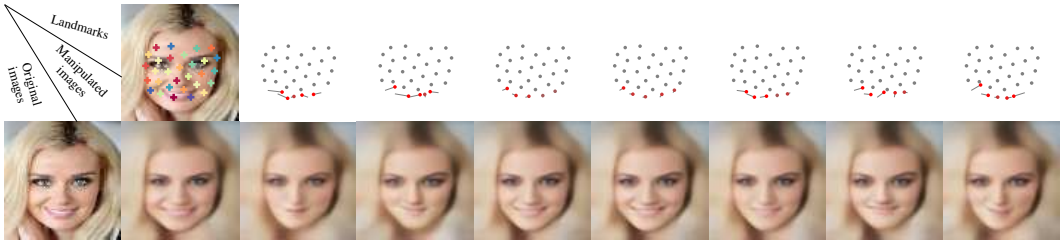


(d) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/09.mp4

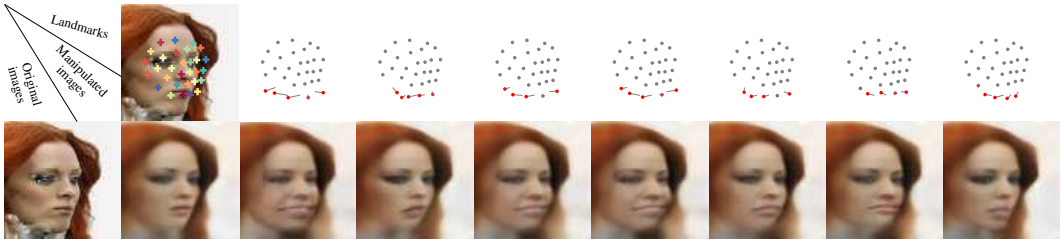


(e) Video: videos/face.30landmark-model.manipulate-mouth-landmarks/10.mp4

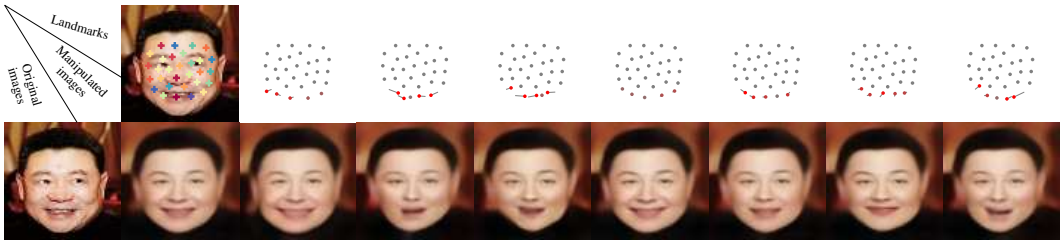
Figure 15: Continued from Figure 14. Face manipulation by modifying mouth 3 discovered mouth landmarks on the MAFL testing set. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at videos/face-manipulation-mouth-landmarks for the morphing process.



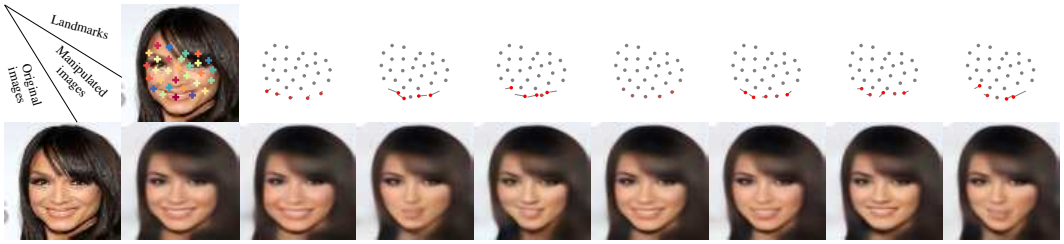
(a) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/01.mp4



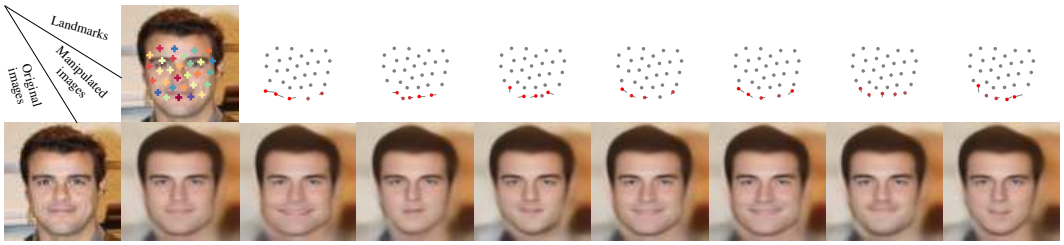
(b) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/02.mp4



(c) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/03.mp4

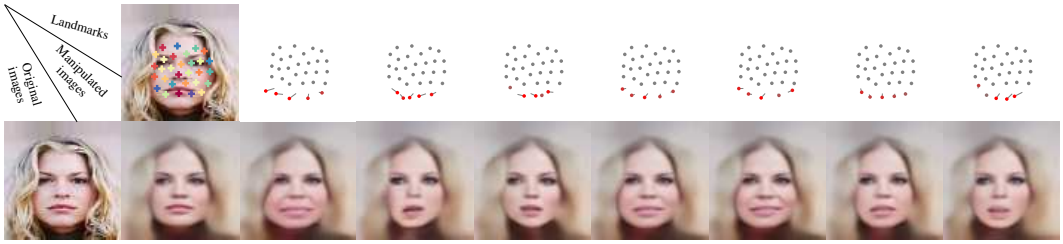


(d) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/04.mp4

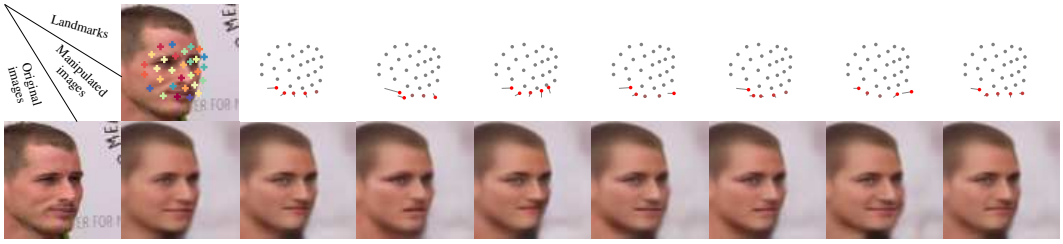


(e) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/05.mp4

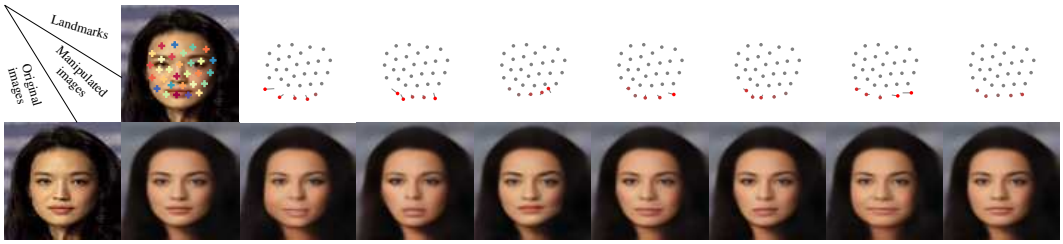
Figure 16: Face manipulation by modifying 6 discovered mouth and jaw landmarks on the MAFL testing set. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at videos/face.30landmark-model.manipulate-mouthext-landmarks for the morphing process.



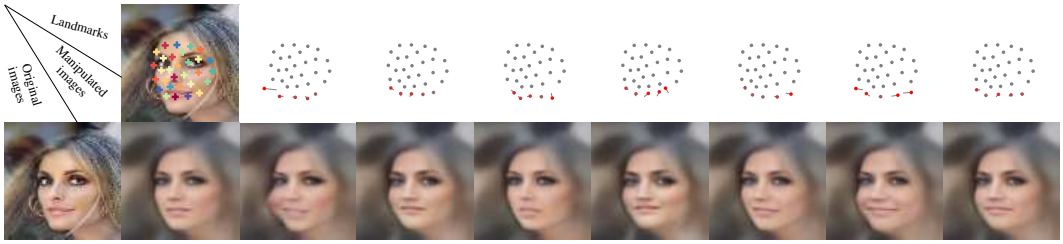
(a) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/06.mp4



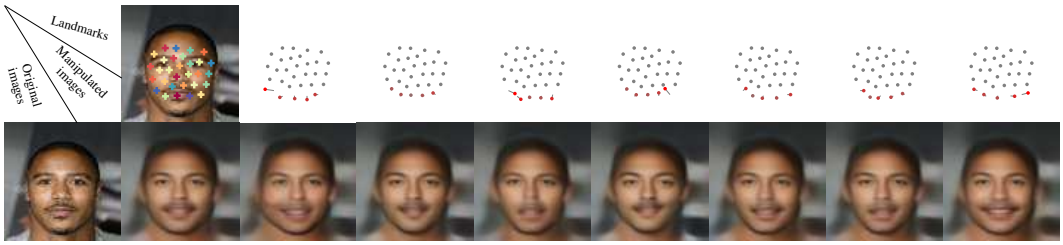
(b) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/07.mp4



(c) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/08.mp4



(d) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/09.mp4



(e) Video: videos/face.30landmark-model.manipulate-mouthext-landmarks/10.mp4

Figure 17: Continued from Figure 14. Face manipulation by modifying mouth 6 discovered mouth and jaw landmarks on the MAFL testing set. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at videos/face-manipulation-mouthext-landmarks for the morphing process.

B. Our model without landmark descriptors on MNIST

We train our landmark discovery model without the landmark descriptor pathway on MNIST in two settings: one model for each digit (Appendix B.1) and one model for all ten digits (Appendix B.2). Using the model for all digits, we can perform geometrically meaningful morphing between different digits (Appendix B.3), e.g., morphing 2 to 9. Videos for the morphing process are available in the folder `videos/mnist-morphing` .

B.1. Models for individual digits

In this section, our landmark discovery model is trained for each digit independently. As shown in Figure 18, the discovered landmarks are consistent within each digit despite the shape variations.

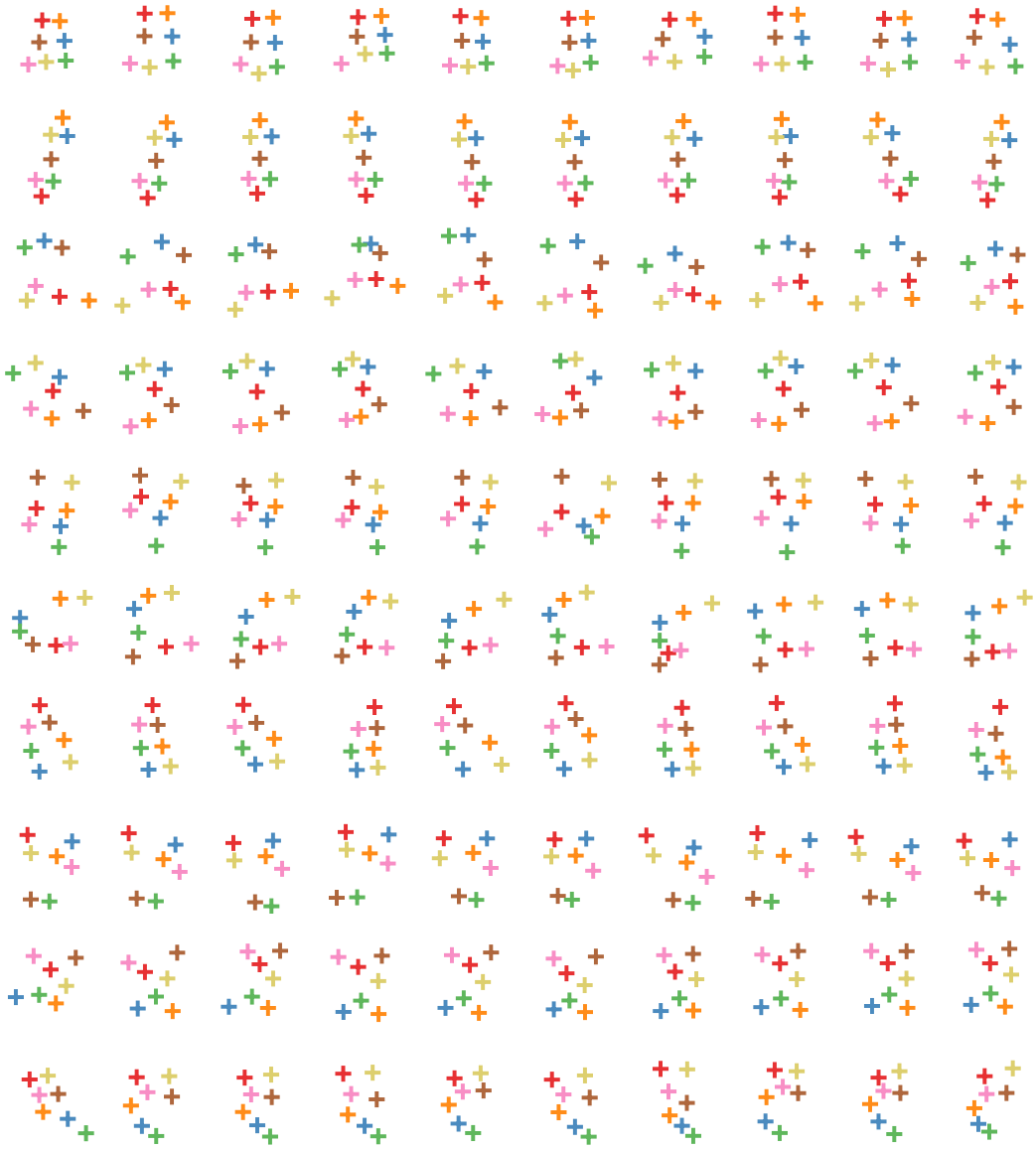


Figure 18: Discovering 7 landmarks on MNIST. Our model is trained independently for each digit.

B.2. Models for all digits

In this section, we train a single model for all ten digits together. In Figure 19, the corresponding landmarks are shown in the same color. The landmarks discovered on the same digits are consistent across different image. **More interesting, the corresponding landmarks across different digits are also semantically consistent.** For examples, the orange cross is always in the middle of a digits, the blue cross at the bottom, and the green one at the most left-top part of every digit.

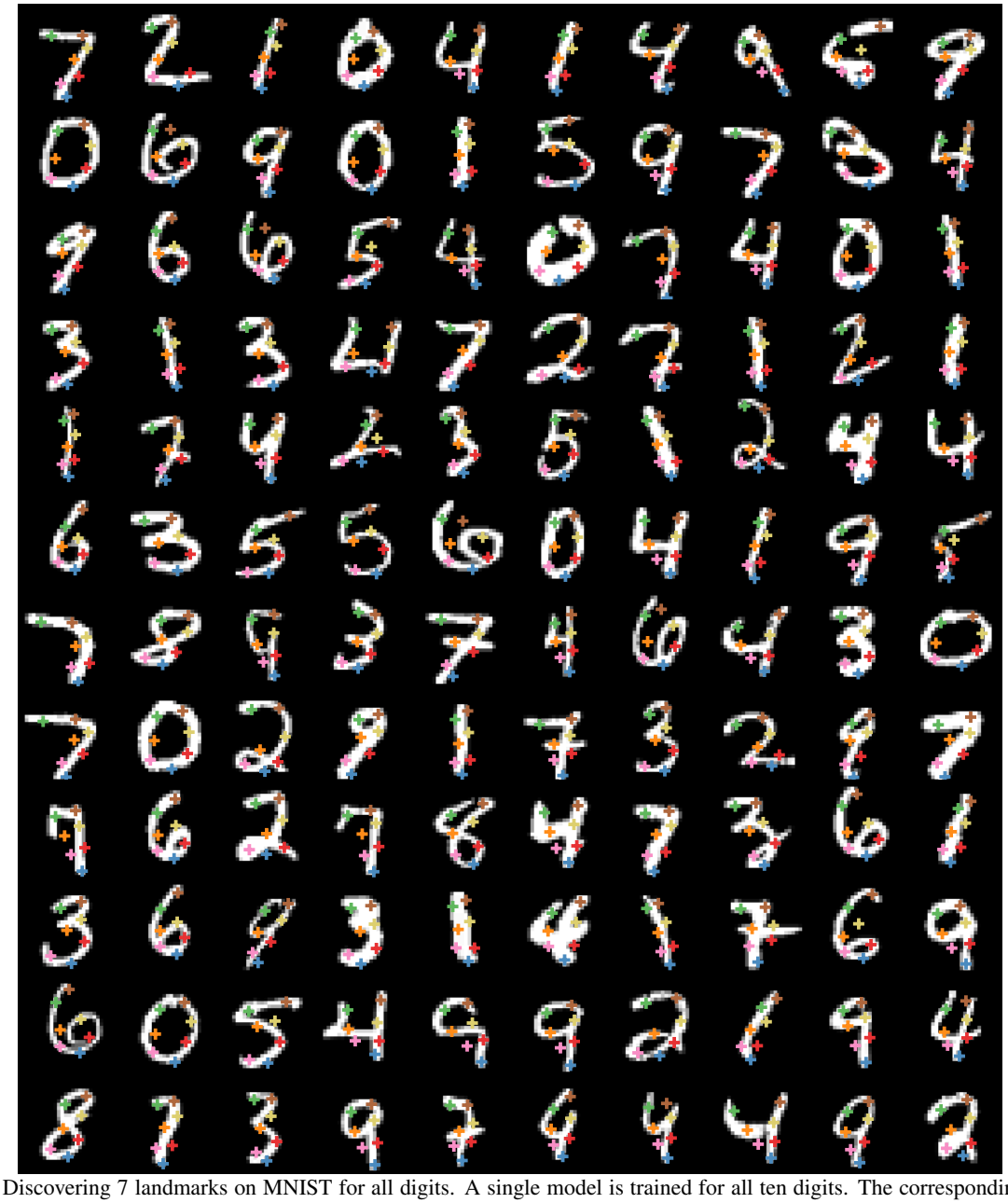


Figure 19: Discovering 7 landmarks on MNIST for all digits. A single model is trained for all ten digits. The corresponding landmarks for different images are in the same color.

B.3. Digit morphing using discovered landmarks

Our model trained on the mix of all ten digits can discover corresponding patterns among different digit categories. Using this model, we can perform geometrically meaningful cross-category morphing. Figure 20 and 21 illustrate the morphing process. Note that the landmark coordinates constitute the full image representation for MNIST digits. Videos for the morphing process are available in the folder `videos/mnist-morphing` .

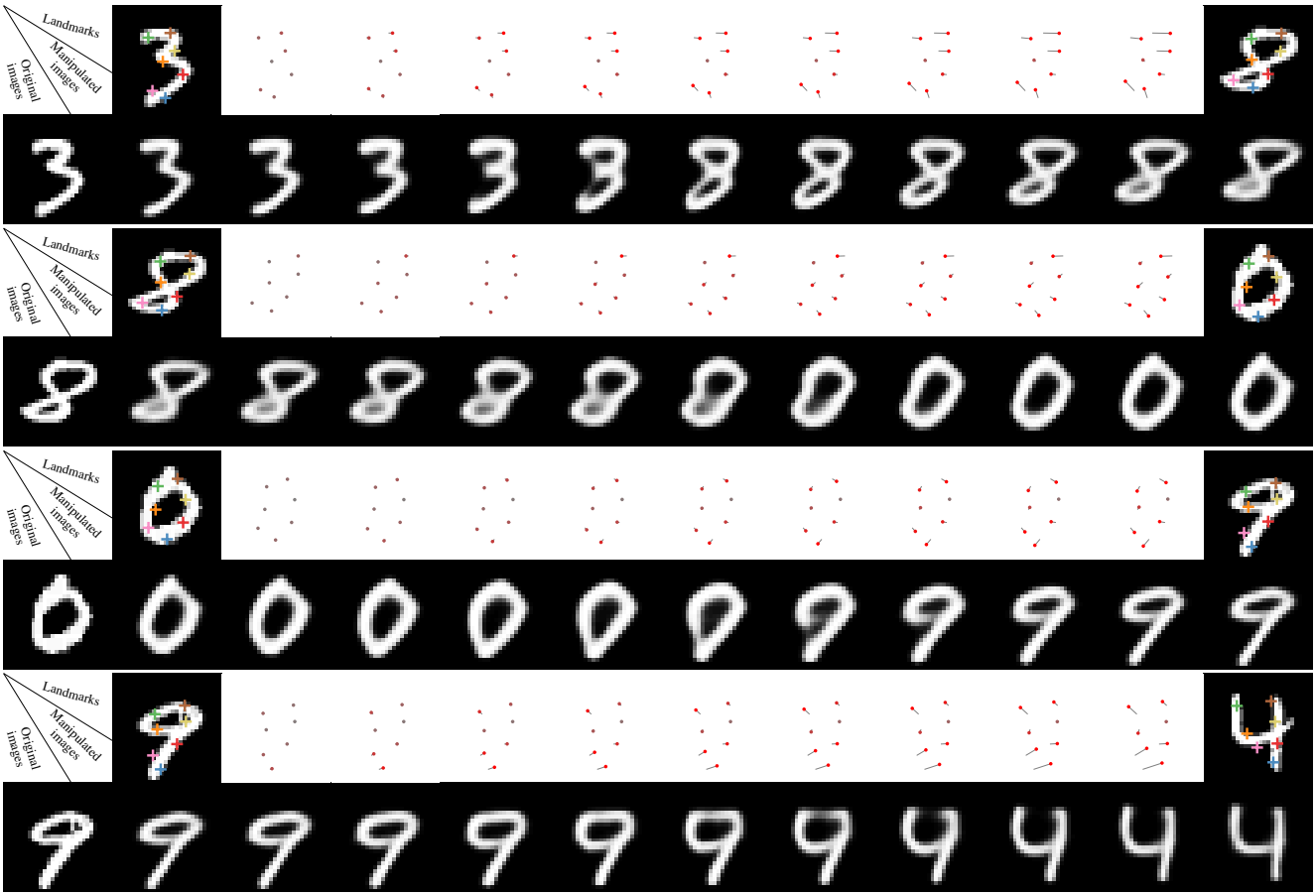


Figure 20: Geometric digital morphing using our discovered landmarks and image decoding module. The landmark coordinates are linearly interpolated between the source digit and the target digit. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images of the source digit; **last column**: discovered landmarks and reconstructed images of the target digit; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at `videos/mnist-morphing` for the morphing process.

Figure 21: Continued from Figure 20. Geometric digital morphing using our discovered landmarks and image decoding module. The landmark coordinates are linearly interpolated between the source digit and the target digit. **1st column**: input images; **2nd column**: discovered landmarks and reconstructed images of the source digit; **last column**: discovered landmarks and reconstructed images of the target digit; **other columns**: the red dots denote the target landmark locations (gray dots means not too much offset regarding the original landmarks), the gray lines denote the synthetic adjustment of landmarks, and the facial images are the decoder outputs. Best viewed in zoom mode. Videos are available at `videos/mnist-morphing` for the morphing process.

C. Details and more results on Human3.6M

On the Human3.6M dataset, we train our landmark discovery model on six actions: waiting, posing, greeting, directions, discussions, and walking. We report quantitative results on predicting the 32 annotated landmarks[5] (acquired by wearable markers) using our models trained for the mix of all six actions and each independent action. We also show qualitative results of the mixed-action model (trained for all six actions).

C.1. Optical flow as self-supervision for equivariance

The Human3.6M training data is in the video format. We can calculate the optical flows between nearby frames and take them as self-supervision for the equivariance constraint defined in (8). Following the same notations, the two frames are $\mathbf{I}$ and $\mathbf{I}'$, and the optical flows define the transformation $g(\cdot, \cdot)$. In particular, we use the Farneback method in OpenCV to compute the dense optical flows at 5-frame intervals. We then accumulate two optical flow fields to calculate the optical flows at 10-frame intervals. The 10-frame-interval optical flow fields in addition to the random TPS transform are used in the equivariance constraint.

Note that using optical flow as the self-supervision for the equivariance constraint can push the landmarks to the background region. This phenomenon occurs because locating landmarks at image regions with weaker optical flows can result in low equivariance loss. To prevent trivial landmarks, we encourage the landmarks to be discovered at locations with strong optical flows. Let $\mathbf{O}_{\mathrm{x}}$ and $\mathbf{O}_{\mathrm{y}}$ be the x, y components of the optical flow map from the current image to another frame. The flow magnitude map is $\mathbf{O}_{\mathrm{n}}(u, v) = (\mathbf{O}_{\mathrm{x}}(u, v))^2 + (\mathbf{O}_{\mathrm{y}}(u, v))^2$. Recall that $\tilde{\mathbf{R}}$ is the multi-channel heatmap computed from the landmark locations. We encourage $\mathbf{O}_{\mathrm{n}}$ and $\tilde{\mathbf{R}}$ to have a significant correlation. In particular, loss to encourage

$$L_{\text{flow-prefer}} = -\frac{\sum_{v=1}^{H}\sum_{u=1}^{W}\mathbf{O}_{\mathrm{n}}(u, v)\sum_{k=1}^{K}\tilde{\mathbf{R}}_k(u, v)}{\sum_{v=1}^{H}\sum_{u=1}^{W}\mathbf{O}_{\mathrm{n}}(u, v)} \tag{16}$$

We use the same loss weight λ_{eqv} for $L_{\text{flow-prefer}}$ as L_{eqv}.

Using the above formulation, we can reduce the ground landmark prediction error from 4.91 (without optical flows) to 4.14 (with optical flows). For all experiments on Human3.6M, we use the optical flow as self-supervision for equivariance as described above.

C.2. Quantitative results

We compare our model with Thewlis et al. [59]'s unsupervised landmark discovery method regarding the annotated-landmark prediction accuracy. Both models discover 16 landmarks. The whole training set is used to train the linear mapping from the discovered landmarks to the annotated ones. As discussed in the main paper, we do not expect the two unsupervised methods to distinguish the frontal and back views. Thus, in the evaluation, we compute the errors against the original landmark annotations and its left-right-flipped counterpart[6], and then we choose the minimum value as the final error. Note that, when flipping the landmark annotations, the landmarks for the whole body are flipped simultaneously. As to the linear regressor training, we propose the following training strategy.

1. Figure out the rough orientations of the human body, heuristically. If more than 2/3 of the left-hand side annotated landmarks are to the right of the right-hand side annotated landmarks, the human is in the frontal view.
2. Train the regressor using the images with the landmark annotations in the frontal view. The other images are ignored in this step.
3. Use the aforementioned evaluation protocol to determine if the landmark annotations on other images should be flipped or not. The model is then retrained with all the training images.
4. Repeat the step 3 until the model is converged.

As shown in Table 4, our method outperforms Thewlis et al. [59]'s method significantly. We also report the results obtained by Newell et al. [39]'s supervised stacked hourglass network using their off-the-shelf pretrained 16-landmark model. Both unsupervised methods perform worse than the supervised stacked hourglass network. However, our model is unsupervised, and our neural network architectures are also smaller. We believe that our results show the potential of unsupervised methods for discovering complicated object structures.

[5] Some markers are close to each other (e.g., on each foot, there are two markers), so the effective locations annotated by the markers are less than 32 (around 16).

[6] For examples, we swap the coordinates of the left-shoulder landmark and the right-shoulder landmark.

Methods		Mixed Actions	Waiting	Posing	Greeting	Directions	Discussion	Walking
Unsupervised	Thewlis et al. [59]	7.51	7.54	8.56	7.26	6.47	7.93	5.40
	Ours (discovered landmarks)	**4.14**	**5.01**	**4.61**	**4.76**	**4.45**	**4.91**	**4.61**
Supervised	Hourglass Newell et al. [39]	**2.16**	**1.88**	**1.92**	**2.15**	**1.62**	**1.88**	**2.21**

Table 4: Comparison with unsupervised and supervised methods for annotated landmark prediction on the Human 3.6M testing sets. The error is in % regarding the edge length of the image.

C.3. Qualitative results

We train our model and Thewlis et al. [59]'s model on all six chosen actions and perform the annotated-landmark prediction. Figure 22 shows the side-by-side comparison. In general, our method visually outperforms Thewlis et al. [59]'s. Figure 23 shows landmark discovery examples, where our method outperforms Thewlis et al. [59]'s method very significantly.

See next page for the figure.

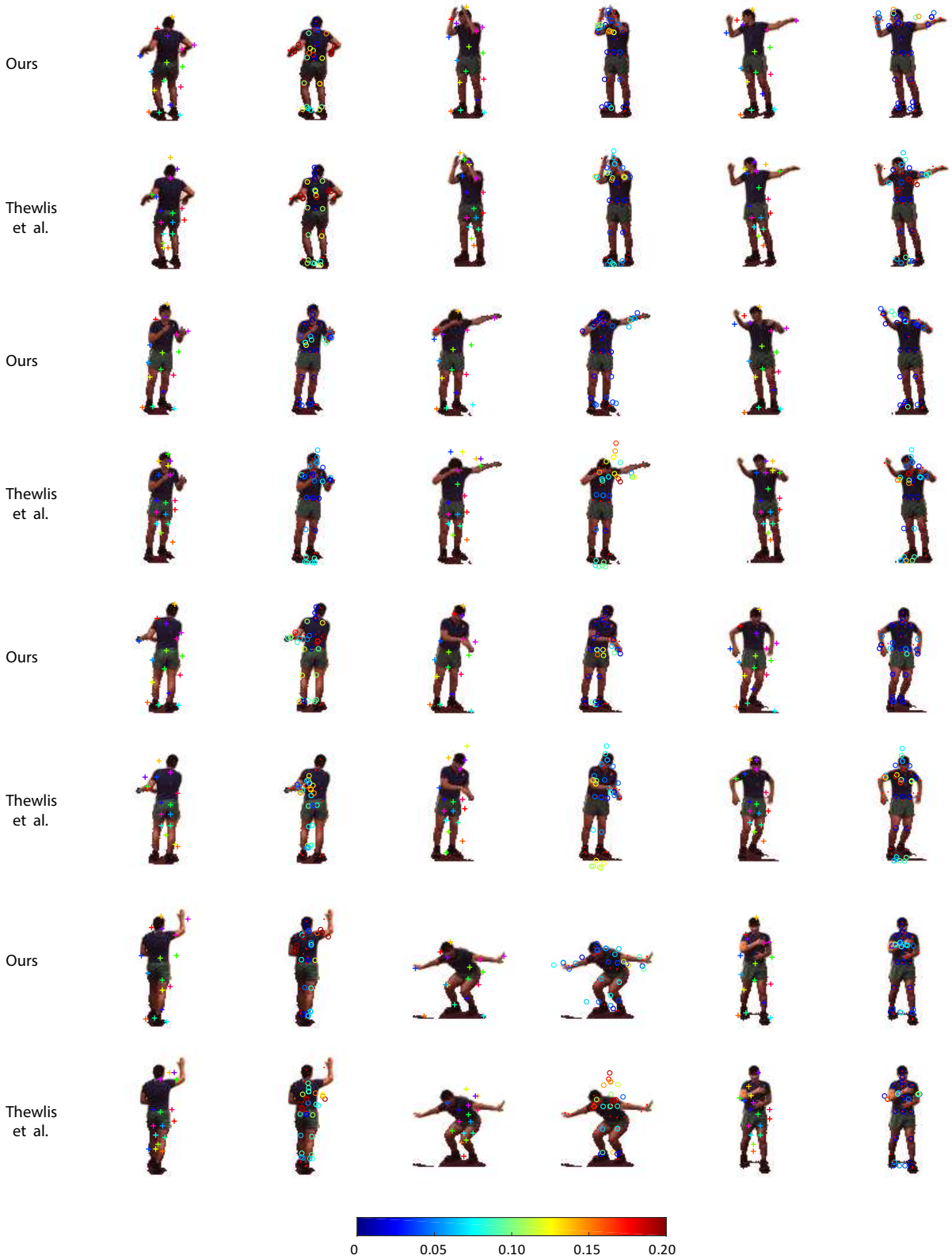


Figure 22: Prediction of 32 annotated landmarks on Human 3.6M. Colorful cross: discovered landmark; Red dot: annotated landmark; Circle: regressed landmark, whose color represent its distance to the annotated land-marks. See the color bar for the distance (i.e., prediction error). Our method shows more deep blue circles (for example, the image in second row second column, the image in last row second column), which means more landmarks with low error compared with Thewlis et al. [59]

Figure 23: Discovering 16 landmarks on Human3.6M testing set. We illustrate some cases when our method outperforms Thewlis et al. [59]'s very significantly.

D. Results on animals of mixed species

On the animal-with-attributes (AwA) dataset [28], we choose the profile images from five animal categories (antelope, deer, moose, horse, zebra) and try to detect landmarks on these mixed species of animals. As shown in Figure 24, even though multiple species of animals with different appearance are mixed, our method can still find several consistent landmarks. For example, the yellow cross is always on the hoof, the orange cross always above the back and the light green cross at the buttock. The landmarks are consistently detected despite the significant variations in species, pose and individual appearance.

See next page for the figure.

Figure 24: Discovering 10 landmarks on mixed animal images of five different species: antelope, deer, moose, horse, zebra

E. More qualitative results on human faces, cat heads, cars, and shoes

In this section, we show more result of landmark discovery and ground truth landmark prediction compared with Thewlis et al. [59]. All the shown images are randomly sampled from the test set.

E.1. CelebA

Figure 25: Discovering 10 landmarks on CelebA, the detected landmarks are highly aligned with facial features such as mouth corner, eyes corner and nose

Figure 26: Discovering 30 landmarks on CelebA

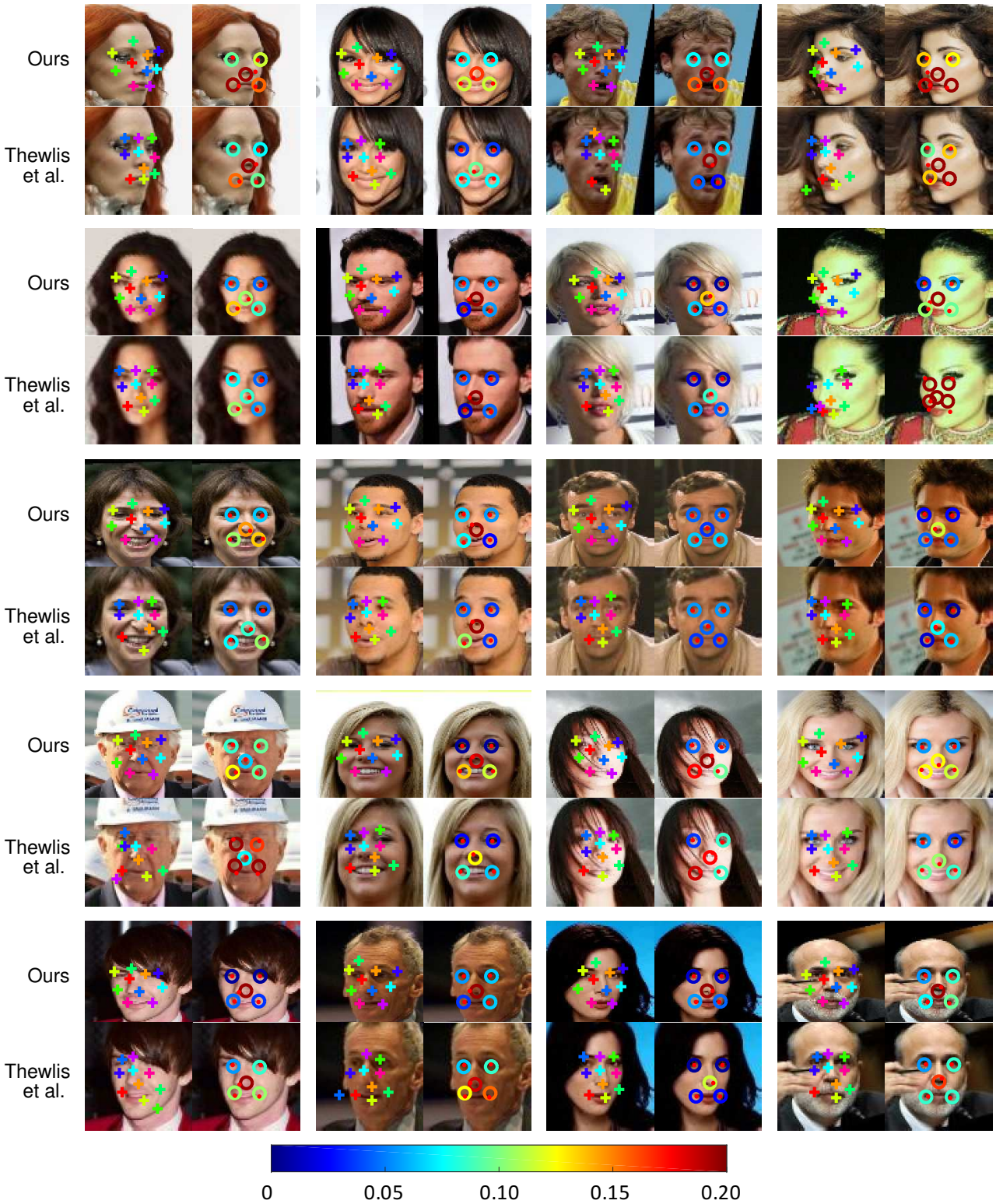


Figure 27: Prediction of 5 annotated landmarks on CelebA. Colorful cross: discovered landmark; Red dot: annotated landmark; Circle: regressed landmark, whose color represent its distance to the annotated land-marks. See the color bar for the distance (i.e., prediction error). Our method shows more deep blue circles (for example, the image in first row first column, the image in first row fourth column), which means more landmarks with low error compared with Thewlis et al. [59]

Figure 28: Discovering 30 landmarks on unaligned CelebA images.

E.2. AFLW

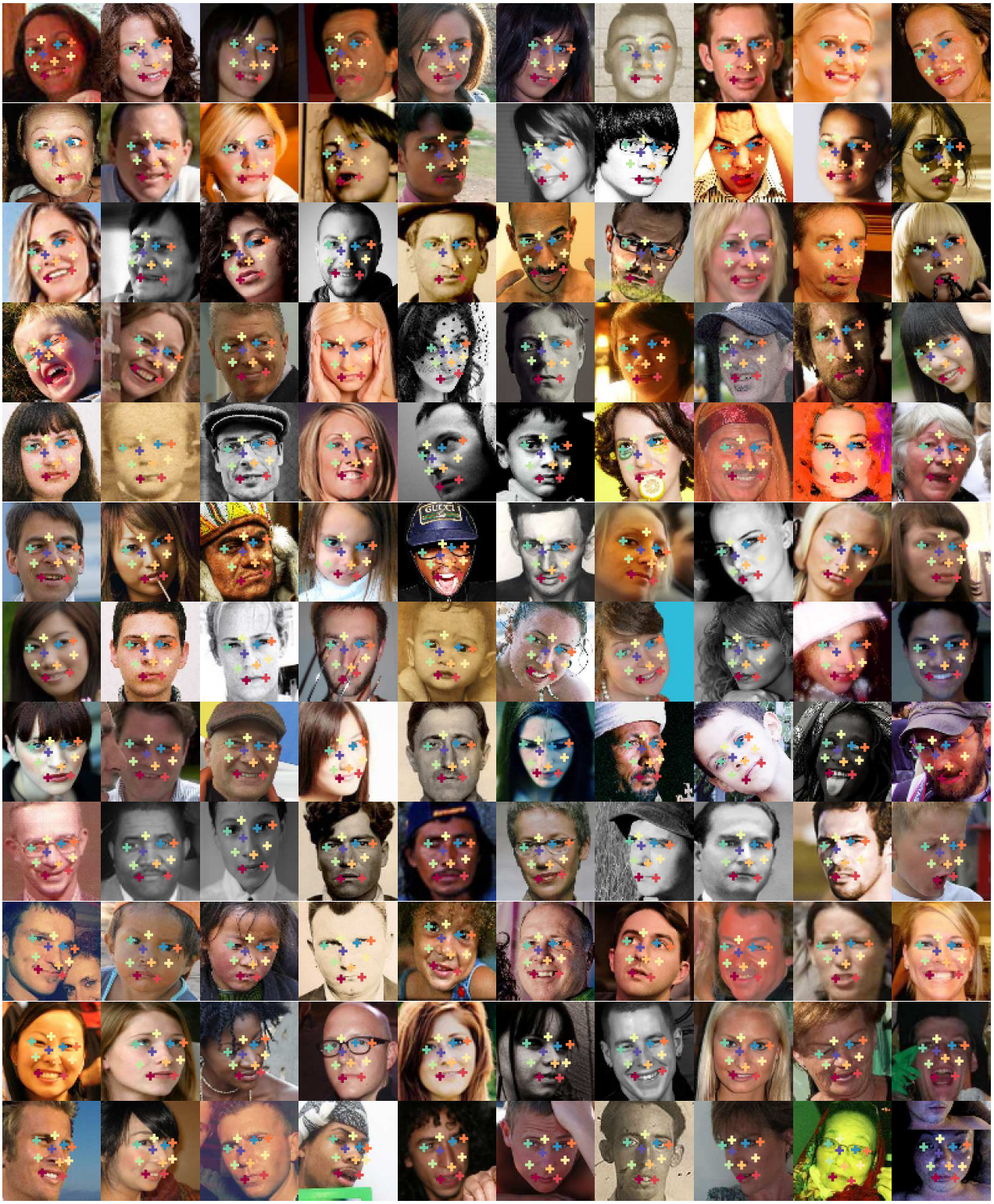


Figure 29: Discovering 10 landmarks on AFLW

Figure 30: Discovering 30 landmarks on AFLW

E.3. Cat heads

Figure 31: Discovering 10 landmarks on Cat Head dataset, our method find some landmarks on the nose, eyes and base of earlobe

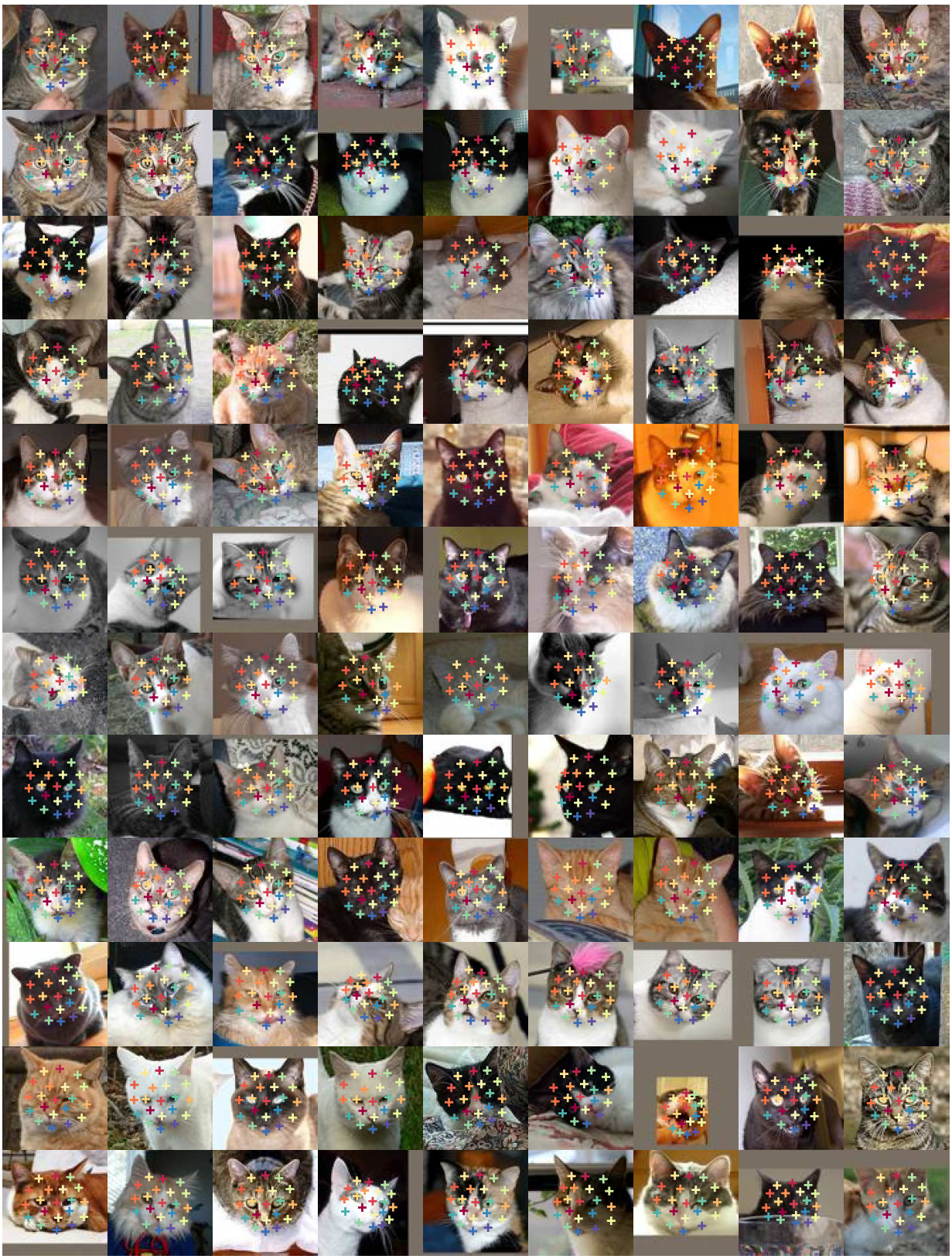


Figure 32: Discovering 20 landmarks on Cat Head dataset

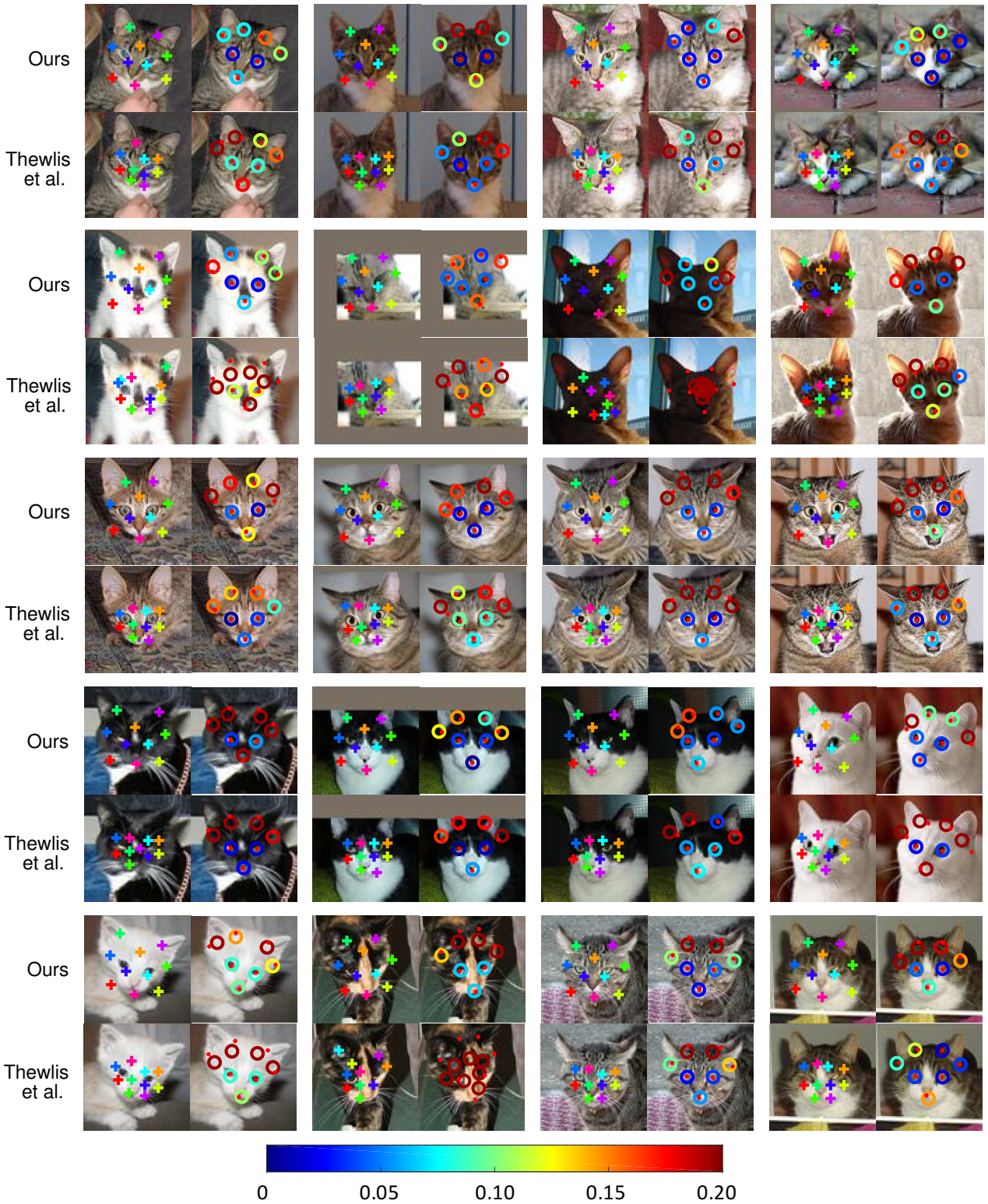


Figure 33: Prediction of 7 annotated landmarks on cat head. Colorful cross: discovered landmark; Red dot: annotated landmark; Circle: regressed landmark, whose color represent its distance to the annotated land-marks. See the color bar for the distance (i.e., prediction error).Our method shows more deep blue circles (for example, the image in fourth row third column, the image in second row first column), which means more landmarks with low error compared with Thewlis et al. [59]

E.4. Cars

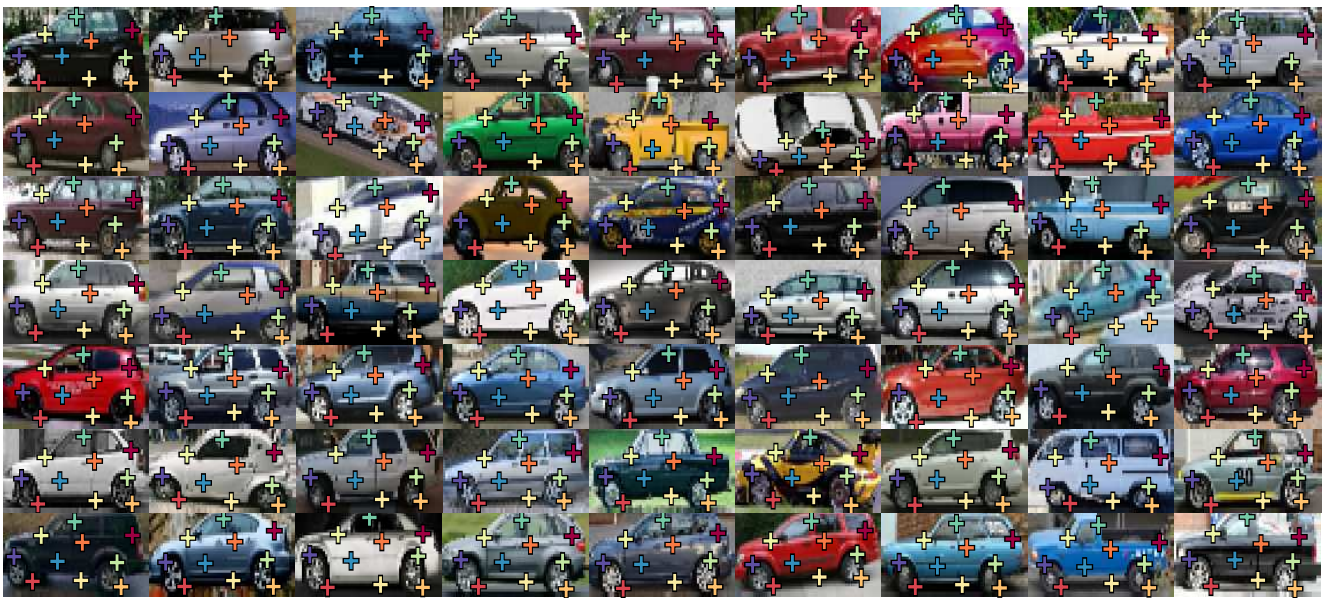


Figure 34: Discovering 10 landmarks on PASCAL-VOC 3D Car dataset

Figure 35: Discovering 24 landmarks on PASCAL-VOC 3D Car dataset

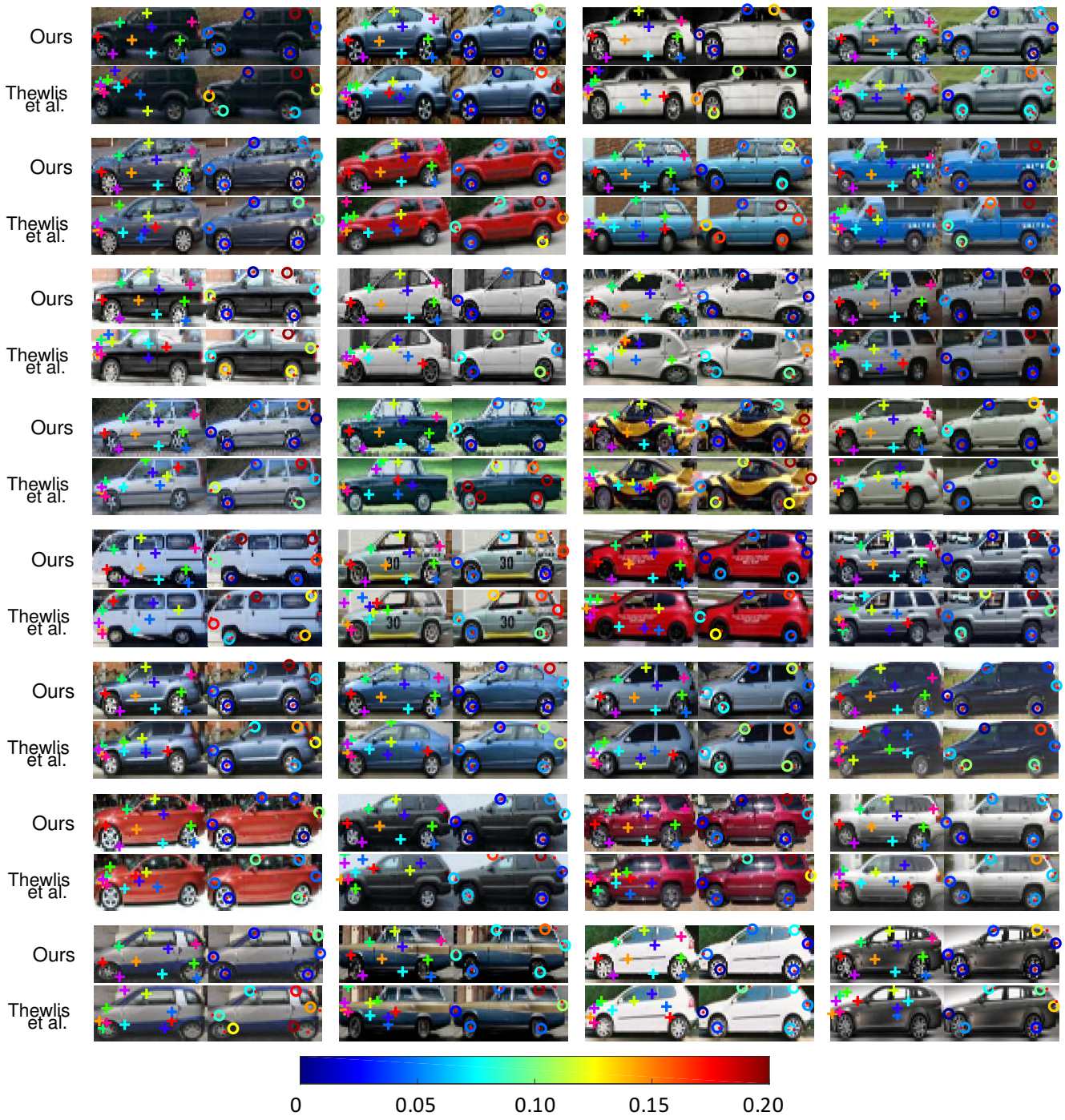


Figure 36: Prediction of 6 annotated landmarks on car. Colorful cross: discovered landmark; Red dot: annotated landmark; Circle: regressed landmark, whose color represent its distance to the annotated land-marks. See the color bar for the distance (i.e., prediction error).Our method shows more deep blue circles (for example, the image in last row first column, the image in last row last column), which means more landmarks with low error compared with Thewlis et al. [59]

E.5. Shoes

Figure 37: Landmark discovery results of our model on shoes using 8 landmarks

F. Ablative study

F.1. Number of labeled samples for annotated-landmark prediction

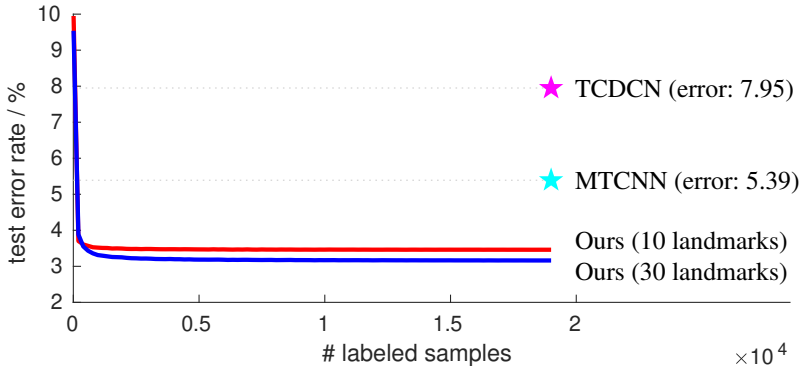


Figure 38: Prediction errors of manually-annotated landmarks on the MAFL testing set when using different numbers of labeled samples to train the linear regressor.

Taking our model as a detector of manually annotated landmarks, we find that less than 200 samples are enough for our model to achieve less than 4% mean error on the MAFL testing set, which is better than the performance of two popular off-the-shelf models. This result suggests that it is possible to train a high-accurate landmark detector using only a few labeled sample when sufficient unlabeled samples are given to train our unsupervised model. We show its performance versus the number of labeled samples in Figure 38.

F.2. Evolution of detection confidence map during training

Figure 39 shows the detection confidence maps of an input image at different training stage of our model. In the beginning, the heatmap shows random values over the whole image. As a result, the landmarks, defined as the mean coordinates weighted by the confidence maps, are all at the center of the image. As the training goes on, the values gradually becomes spatially concentrated. With the separation loss defined in (7), the peaked value of each channel of the confidence map can move to a different location. Without the separation loss, every channel can have a peaked value at the center of the images, resulting in degenerate landmarks.

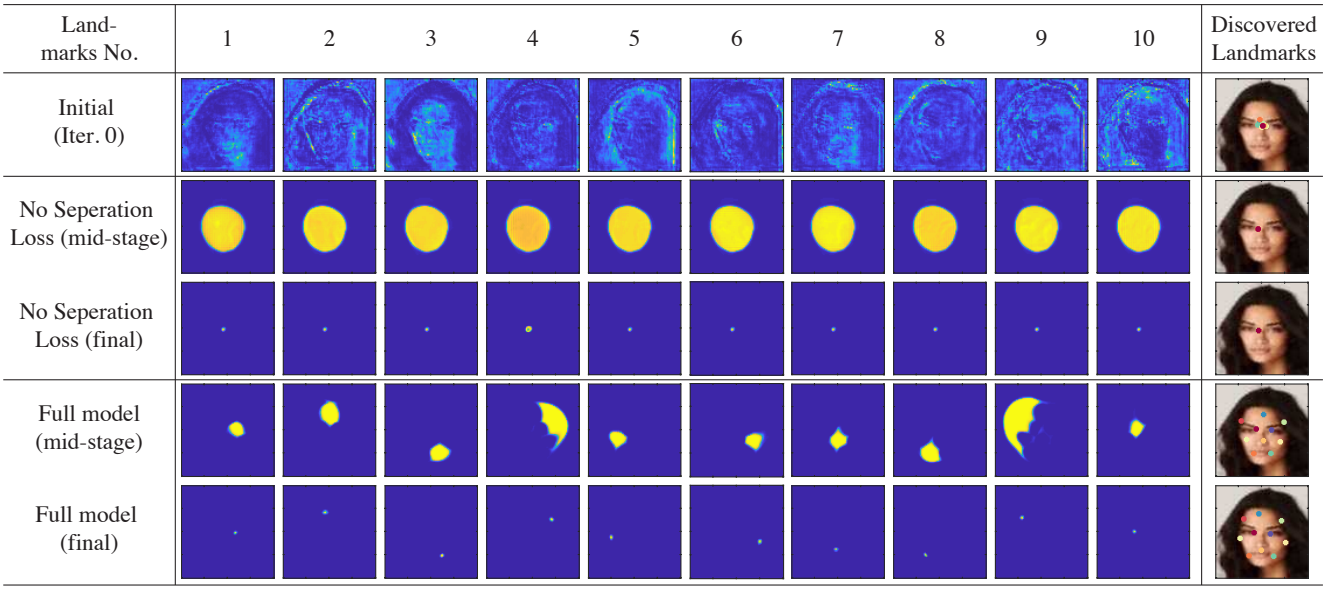


Figure 39: The evolution of the detection confidence map $\mathbf{D}$ during training. The training of a 10-landmark CelebA model is monitored.

F.3. TPS control points

For the random TPS in our equivariance constraint (defined in (8)), we both use the regular-grid control points and take the discovered landmarks in the current iteration as the control points. The two sets of control points are alternatively used in each optimization iteration with 7:3 chance. We do not exhaustively tune the ratio and keep it the same in all experiments.

As shown in Figure 5, the performance of our model is fairly insensitive to this ratio when the other hyper-parameters are fixed. However, introducing the discovered landmarks as the TPS control points does benefit.

grid:landmark	10:0	1:9	3:7	5:5	7:3	9:1
GT prediction error / %	4.17	3.86	3.46	3.38	3.46	3.41

Table 5: Impact of the ratio between two types of TPS control points (i.e., regular grid and discovered landmarks). Our 10-landmark CelebA model is trained under different ratios of the TPS control points. The evaluation metric is the ground truth (GT) landmark prediction errors with the CelebA training set for linear regressor training.

Note that the discovered landmarks are clustered at the center of the image (see discussion about the separation constraint in Section 3.2) and cannot serve as good TPS control points. As a result, we use only the regular-grid control points in the beginning and start to apply the previously mentioned ratio after training the model for several thousands of iterations.

G. Implementation details

G.1. Data preprocessing

The main paper and this supplementary materials report results on several datasets. Table 6 summarizes the image size we used for each each dataset. In our landmark discovery formulation, we need to perform random TPS to calculate the equivariance constraint in (8). It requires the image to have large enough margins so that the foreground will not be out of image due to the random transformation. Table 6 also summarizes the image size after padding with the edge values.

Dataset	Image size	Padded size	Remark
CelebA, AFLW, Cat Heads, Shoes	80×80	96×96	-
Profile Cars from PASCAL 3D	64×64	96×96	visualized as $W : H = 2 : 1$
Animals from AwA	64×64	80×80	-
Human3.6M	128×128	192×192	-
MNIST	28×28	56×56	-

Table 6: Data processing parameters for different dataset.

For different datasets, we crop the foreground images and prepare input images as follows.

CelebA We started from the cropped-and-aligned images (218×178) in CelebA dataset, scaled them to 100×100 pixels and then cropped the 80×80 center patches.

AFLW The dataset provides annotated bounding boxes. We enlarged the bounding boxes on each image with a margin on the top ($1/4$ height of the original bounding box), margins on the left and right ($1/8$ width of the original bounding box) so that the cropped facial images look similar to the CelebA data. The cropped images were also scaled to 80×80.

Cat Heads The dataset provides ground truth landmarks on the cat head. We figured out the bounding box for cropping each image according to the landmarks and then scaled the cropped images to 80×80.

Shoes We scaled the shoe images (102×136) to 64×64, and padded the images to be 80×80 with white margins. We linearly transformed the color value range from $[0, 1]$ to $[0.1, 0.9]$ to avoid a huge amount of saturated responses for the output layer with the sigmoid activation. The color was scaled back for visualization.

Cars The PASCAL 3D dataset provides annotations for the orientation and landmarks of several objects. We cropped the profile car images according to the bounding box of the ground truth landmarks. Slight margins are added to the bounding box, and the cropped image is scaled to a square image without preserving the aspect ratio.

Animals from AwA We manually annotated bounding boxes and orientation labels (i.e., left, right, frontal, back) for several types of animals. For each rectangular bounding box, we enlarged its shorter edge to make the box square, cropped the patch, and scaled it to 64×64. We ignored the images with frontal and back views, and we flipped the right-facing animals horizontally so that all animals in the image face to the left.

Human3.6M The human body was cropped using a square bounding box from the original video frames. We use the 3D landmarks provided in this dataset, acquired by wearable markers, as side information to roughly align the scales and foot locations of the human bodies in different images. The cropped square images are scaled to 128×128 pixels. We use the provided segmentation masks, obtained by an off-the-shelf unsupervised background removal method, to mask out the background image with gray color. For visualization, we show the gray color as white for the printing clarity.

Type	Kernel size	Output channels	Output size
Input	n/a	3	80
Convolution	3	16	80
Convolution	3	16	80
Pooling	2	16	40
Convolution	3	32	40
Convolution	3	32	40
Pooling	2	32	20
Convolution	3	64	20
Convolution	3	64	20
Pooling	2	64	10
Convolution	3	64	10
Convolution	3	64	10
Convolution	3	64	10
Convolution	3	64	10
Upsampling	2	64	20
Convolution	3	64	20
Convolution	3	64	20
Upsampling	2	64	40
Convolution	3	32	40
Convolution	3	32	40
Upsampling	2	32	80
Convolution	3	16	80
Convolution	3	?*	80

(a) 64×64 (padded to 80×80): hourglass_ℓ

Type	Kernel size	Output channels	Output size
Input	n/a	3	80
Convolution	3	32	80
Convolution	3	64	80
Pooling	2	64	40
Convolution	3	128	40
Pooling	2	128	20
Convolution	3	256	20
Pooling	2	256	10
Convolution	3	512	10
Convolution	3	512	10
Upsampling	2	512	20
Convolution	3	256	20
Upsampling	2	128	40
Convolution	3	128	40
Upsampling	2	64	80
Convolution	3	64	80
Convolution	3	64	80

(b) 64×64 (padded to 80×80): hourglass_f

Type	Kernel size	Output channels	Output size
Input	n/a	3	96
Convolution	3	32	96
Convolution	3	32	96
Pooling	2	32	48
Convolution	3	64	48
Convolution	3	64	48
Pooling	2	64	24
Convolution	3	64	24
Convolution	3	64	24
Pooling	2	64	12
Convolution	3	64	12
Convolution	3	64	12
Upsampling	2	64	24
Convolution	3	64	24
Convolution	3	64	24
Upsampling	2	64	48
Convolution	3	64	48
Convolution	3	64	48
Upsampling	2	64	96
Convolution	3	32	96
Convolution	3	?*	96

(c) 64×64 or 80×80 (padded to 96×96): hourglass_ℓ

Type	Kernel size	Output channels	Output size
Input	n/a	3	96
Convolution	3	32	96
Convolution	3	64	96
Pooling	2	64	48
Convolution	3	128	48
Pooling	2	128	24
Convolution	3	256	24
Pooling	2	256	12
Convolution	3	512	12
Convolution	3	512	12
Upsampling	2	512	24
Convolution	3	256	24
Upsampling	2	128	48
Convolution	3	128	48
Upsampling	2	64	96
Convolution	3	64	96
Convolution	3	64	96

(d) 64×64 or 80×80 (padded to 96×96): hourglass_f

Type	Kernel size	Output channels	Output size
Input	n/a	?**	64
Convolution	3	64	64
Convolution	3	64	64
Pooling	2	64	32
Convolution	3	128	32
Pooling	2	128	16
Convolution	3	256	16
Pooling	2	256	8
Convolution	3	512	8
Pooling	2	512	4
Convolution	3	512	4
Convolution	3	512	4
Upsampling	2	512	8
Convolution	3	512	8
Upsampling	2	512	16
Convolution	3	256	16
Upsampling	2	128	32
Convolution	3	128	32
Upsampling	2	64	64
Convolution	3	64	64
Convolution	3	3	64

(e) 64×64: hourglass_d

Type	Kernel size	Output channels	Output size
Input	n/a	?**	80
Convolution	3	64	80
Convolution	3	64	80
Pooling	2	64	40
Convolution	3	128	40
Pooling	2	128	20
Convolution	3	256	20
Pooling	2	256	10
Convolution	3	512	10
Pooling	2	512	5
Convolution	3	512	5
Convolution	3	512	5
Upsampling	2	512	10
Convolution	3	512	10
Upsampling	2	512	20
Convolution	3	256	20
Upsampling	2	128	40
Convolution	3	128	40
Upsampling	2	64	80
Convolution	3	64	80
Convolution	3	3	80

(f) 80×80: hourglass_d

Type	Kernel size	Output channels	Output size
Input	n/a	3	192
Convolution	3	32	192
Pooling	2	32	96
Convolution	3	64	96
Pooling	2	64	48
Convolution	3	128	48
Pooling	2	128	24
Convolution	3	256	24
Pooling	2	256	12
Convolution	3	512	12
Pooling	2	512	6
Convolution	3	512	6
Convolution	3	512	6
Upsampling	2	512	12
Convolution	3	512	12
Upsampling	2	512	24
Convolution	3	256	24
Upsampling	2	256	48
Convolution	3	128	48
Upsampling	2	128	96
Convolution	3	64	96
Convolution	3	?***	96

(g) 128×128 (padded to 192×192): hourglass_ℓ, hourglass_f

Type	Kernel size	Output channels	Output size
Input	n/a	?**	64
Convolution	3	64	64
Pooling	2	64	32
Convolution	3	128	32
Pooling	2	128	16
Convolution	3	256	16
Pooling	2	256	8
Convolution	3	512	8
Pooling	2	512	4
Convolution	3	512	4
Pooling	3	512	4
Convolution	2	512	8
Convolution	3	512	8
Upsampling	2	512	16
Convolution	3	256	16
Upsampling	2	256	32
Convolution	3	128	32
Upsampling	2	128	64
Convolution	3	64	64
Upsampling	2	128	64
Convolution	3	32	128
Convolution	3	3	128

(h) 128×128: hourglass_d

?*: decided by the number of landmarks. ?**: decided by the number of landmarks and the dimension of the shared feature space. ?***: this layer is used only for hourglass_ℓ, and the number of channels is decided by the number of landmarks. ?****: decided by the number of landmarks for hourglass_ℓ, $= 32$ for hourglass_f, and $= 3$ for hourglass_d. Each skip-link in **(a)**, **(b)**, **(c)**, **(d)**, and **(f)** consists of three convolutional layers of 3×3 kernels. Each skip-link in **(e)**, **(g)**, and **(h)** consists of two convolutional layers of 3×3 kernels.

Figure 40: Architectures of our hourglass-style networks. **(a)**,**(b)**, **(c)**, **(d)**: For 80×80 and 64×64 images, the outputs of hourglass_ℓ and hourglass_f have the same size as the image. **(g)**: For 128×128 images, the outputs of hourglass_ℓ and hourglass_f are 64×64.

G.2. Network architectures

In Section 3 of the main paper, we describe the key architectures of our model and leave some details unspecified. This section describes the detailed neural network architectures.

Figure 40 summarizes the hourglass-like architectures that we used for images of different sizes. The image padding is explaned and specified in Appendix G.1. In general, an hourglass architecture has mirrored encoding (high-resolution to low-resolution) and decoding (low-resolution and high-resolution) architectures. Skip-links, made up of convolutional layers, create shortcuts from the encoding feature maps and decoding feature maps of the same resolution. The responses of the skip-links are fused with the main stream decoding responses using element-wise addition. We use the max-pooling to reduce the feature map size for encoding, and we upsample feature maps by the nearest interpolation for decoding. For each linear layer, a batch normalization layer is followed, and LeakyReLU [34] is the default activation function.

Based on the neural network architectures in Figure 40, a convolutional layer of 1×1 kernels calculates the raw detection score map of $K+1$ channels (recall that K is the number of landmarks). For image output, the last convolutional layer's responses ($\in \mathbb{R}$) are mapped to $(0,1)$ with a sigmoid function for the reconstructed color intensity and to $(0,+\infty)$ by $\log(1+2\exp(z))/2$ for the pixel-wise standard deviation, respectively.

G.3. Training strategy

We use Adam with an initial learning rate of 0.001 to optimize the neural network parameters. We set the training batch size to 16 or 32[7] for 80×80 and 60×60 images and 8 for 128×128 images. The learning rate starts from 10^{-3} and decreases to 10^{-4} and 10^{-5} later. For color images, we do random brightness and contrast jittering.

For batch normalization, the global mean and variance are computed using a random subset of the training set when the neural network training is done. Note that using the running average during training for the global mean and variance can hurt the performance.

To implement the equivariance constraint in (8), we use random TPS transformation to obtain warped input images (paired with the original images) in each training iteration. Taking the normalized image height and width as 1, the random transformation parameters are:

- **Global affine component.** Uniform random translation in ±0.15; Gaussian random rotation with the standard deviation of $10°$; Gaussian random scaling in the base-2 logarithm scale with the standard deviation of 1.25.
- **Local TPS.** Gaussian random translation with the standard deviation of 0.1 (regarding the regular-grid control points) or 0.05 (regarding the landmark control points).

For the reconstruction loss weight λ_{recon} in (15), we find it helpful to start with a small value and increase it to its final value at a later stage. At the early training stage, the discovered landmarks change significantly for each iteration, and the latent descriptor of landmarks inputted to the decoder also varies a lot. The model parameters of the decoder and the gradients from it can change too drastically if the reconstruction loss weights is large, which would harm the training of both the landmark detector and the landmark-based image decoder. As a particular training strategy, we increase the value of λ_{recon} by $\times10$ twice during the training. Table 7 in Appendix G.4 summarizes the detailed hyper-parameters.

For the L2 reconstruction loss L_{recon}, we scale the image pixel values to $[0,1]$. The loss is explicitly defined as the negative logarithm likelihood (NLL) to draw the input image $\mathbf{I}$ from the Gaussian distribution centered at the reconstructed image $\tilde{\mathbf{I}}$ the with a fix variation $\sigma_{\text{color}}=0.05$. More specifically,

$$L_{\text{recon}}=\frac{1}{\sigma_{\text{color}}^2}\|\mathbf{I}-\tilde{\mathbf{I}}\|_F^2+\ln\left(2\pi\sigma_{\text{color}}^2\right). \tag{17}$$

The value of the loss weight λ_{recon} is based on this definition.

[7] There is no significant difference in the performance when using either 16 or 32 as the training batch size.

G.4. Hyper-parameters

Table 7 summarizes the dataset-specific hyper-parameters. Note that our model is not sensitive to minor changes of the hyper-parameters, but adjusting the hyper-parameters for each dataset can improve the performance slightly.

Dataset	# landmark	1st LR decay iter.	2nd LR decay iter.	Initial λ_{recon}	1st λ_{recon} increase iter.	2nd λ_{recon} increase iter.	λ_{conc}	σ_{sep}	λ_{sep}	λ_{eqv}	C
CelebA	10	100K	200K	0.01	100K	200K	100	0.06	16	10^4	8
CelebA	30	100K	200K	0.1	100K	200K	100	0.04	10	10^4	8
AFLW	10	100K	200K	0.1	100K	200K	100	0.06	16	10^4	8
AFLW	30	100K	200K	0.0001	100K	200K	100	0.04	10	10^4	8
Cat	10	100K	200K	0.0001	100K	200K	100	0.08	20	10^4	8
Cat	20	100K	200K	0.0001	100K	200K	100	0.05	10	10^4	8
Car	10	40K	80K	0.001	40K	50K	100	0.08	200	10^4	8
Car	24	40K	80K	0.001	40K	50K	100	0.05	200	10^4	8
Animal	10	20K	50K	0.001	40K	50K	100	0.08	20	10^4	2
Shoes	8	100K	20K	0.01	100K	200K	100	0.05	20	10^4	8
Human	16	100K	200K	0.1	100K	200K	100	0.06	20	10^4	8

Table 7: The hyper-parameters for our models on different datasets. **When computing loss $L_{\text{conc}}, L_{\text{sep}}, L_{\text{eqv}}$, the coordinates is first *normalized with respect to the image edge length* (i.e., the square root of the image area). All hyper-parameters (including, λ_{conc}, σ_{seq}, λ_{seq}, λ_{eqv}) are set according the normalized landmark coordinates.**

G.5. Details about face generations using unsupervised landmarks

Figure 11 in the main paper shows results of generating facial images conditioned on our discovered landmarks. In this experiment, we fix our landmark discovery module and use it to detect landmarks on training images. For the decoding module, we take the detected landmarks as a given input condition and map an isotropic Gaussian random variable to the latent part of the image representation. Inspired by [46], we first use deconvolutional layers to get a feature map from the random variable and use convolutional layers to get another map of the same size from the reconstructed detection confidence map. We use element-wise multiplication and channel-wise concatenation to fuse the two into one feature map as the input of the decoding neural network. Thus, the way of calculating the input feature map of the decoding module is not the same as our landmark discovery model.

We use the boundary equilibrium GAN (BEGAN) [3] framework to design the discriminator, which encourages the decoder to generate realistic images. More concretely, an autoencoder is trained as the energy function to distinguish the real and generated images. To make sure the generated images are consistent with the landmark condition, we first use our landmark discovery module to detect landmarks on them. We then take the L1 distance between them and those from the corresponding real images (i.e., input training images for getting the landmarks) as an extra training loss for the decoding module.

This experiment is mainly to show that our discovered landmark is accurate and meaningful enough for controllable image generation.