Realistic Full-Body Anonymization with Surface-Guided GANs (Supplementary)

Håkon Hukkelås Morten Smebye Rudolf Mester Frank Lindseth
Norwegian University of Science and Technology
hakon.hukkelas@ntnu.no

Contents

A. Additional Experimental Details

We describe the general setup of the COCO-Body model here and state specific changes to DeepFashion in Sec. A.3.

All experiments were performed with Pytorch v1.11 [22] with 2x Tesla V100-32GB GPUs. FID and PPL is computed with torch fidelity [20], where we modify the code to support conditional image synthesis.

I. Network Architecture and Hyperparameters

We use equalized learning rate [8] for all trainable parameters and use the Adam [12] optimizer with batch size=32, learning rate=0.001, $\beta_1 = 0.0, \beta_2 = 0.99$ and $\epsilon = 10^{-8}$. Note that we set the learning rate to 0.002 for Config E. Each model is trained until the discriminator has seen 12M images. All networks are trained with mixed precision using the implementation provided by Pytorch [22].

Generator Architecture The generator architecture is a U-Net architecture [24] following design principles of StyleGAN2 [11]. There are 5 residual blocks in the encoder and 5 in the decoder, where each block has two 3x3 convolutions. The residual blocks has the following number of output channels (Config E): $[128, 256, 512, 512, 512, 512, 512, 256, 128, 64]$. Convolutions of the same block has the same number of output channels. For other configs, we straightforward scale down the reported output channels to match the number of parameters reported in the main paper. Bilinear upsampling/downsampling is performed after each block (except the end/start of the encoder/decoder). We use 1×1 residual skip connections between the first 4 and last 4 blocks of the encoder/decoder. Each convolution in the residual blocks (except reisdual/u-net skip connections) has the following order of operations; SAM feature modulation $\rightarrow$ convolution $\rightarrow$ LeakyReLU (slope=.2) [17] $\rightarrow$ normalization.

The input image resolution is 288×160 and the minimum feature map resolution is 36×20. For inference, we use the exponential moving average [28] of the generator, where we use $\beta = 0.9977$. For all models (except those with V-SAM, INADE [25], Co-mod [30] or StyleGAN [11]), we linearly project the latent variable to a $1 \times 32 \times 32$, then add it to the output of the encoder.

FPN-Discriminator The FPN-discriminator consists of 6 residual blocks, with the following number of output channels (Config E) for each: $[128, 256, 512, 512, 512, 512]$, which results in 34M parameters. For the FPN-Head, we add a 1×1 convolution that linearly transforms the output of each residual block to the number of output channels (16 for CSE embedding, 26 for semantic segmentation). These feature maps are then bilinearly upsampled and added.

Loss function We use the non-saturating adversarial loss [3] and regularize the discriminator with epsilon penalty [8] and r1-regularization [18]. We mask the r1-regularization by M, similar to [29, 7], and apply it every 16th gradient step (known as "lazy regularization" [11]). The total loss is given by

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda_{EP} \cdot \mathcal{L}_{EP} + \lambda_{GP} \cdot \mathcal{L}_{GP} + \lambda_{CSE} \mathcal{L}_{\text{CSE}} \tag{1}$$

Table 1: All quantitative metrics for every model in the main paper, where the leftmost column specify which table each model corresponds to. Note that some models are repeated in the table between different comparisons. Number of layers in the mapping network (n) is 6 if not stated otherwise.

		Main Metrics				Face Metric	PSNR Affine Transformation		
	Model	LPIPS ↓	LPIPS Diversity ↑	FID ↓	PPL ↓	FID ↓	Translation ↑	Hflip ↑	Rotation ↑
Table 1	A (Baseline)	0.237	0.162	7.4	26.7	13.4	22.0	20.4	19.3
	B (A+ CSE Supervision)	0.220	0.140	5.8	19.0	9.1	23.1	21.7	20.3
	C, SAM	0.219	0.143	5.6	19.2	7.4	23.8	21.4	20.7
	D, V-SAM	0.220	0.166	5.2	13.7	7.4	26.1	22.5	21.4
	E, Larger D/G	0.211	0.161	4.8	15.1	6.8	26.2	22.1	21.0
Table 2	D, V-SAM n=0	0.221	0.155	5.4	24.9	7.7	25.9	22.0	21.1
	D, V-SAM n=2	0.221	0.164	5.4	19.7	8.0	26.1	21.9	21.3
	D, V-SAM n=4	0.221	0.161	5.5	19.8	7.9	26.0	22.0	21.1
	D, V-SAM n=6	0.220	0.166	5.2	13.7	7.4	26.1	22.5	21.4
Table 3	B + SPADE [21]	0.223	0.150	5.9	20.6	9.7	22.5	20.7	19.8
	B + CLADE [26]	0.221	0.138	5.7	16.9	8.9	22.9	21.3	20.1
	B + INADE [25]	0.223	0.140	5.8	19.5	9.4	24.1	20.9	20.2
	B + StyleGAN [11]	0.220	0.155	5.7	48.2	9.4	25.5	21.6	20.9
	B + CoMod	0.221	0.154	5.5	17.9	17.5	24.5	21.6	20.6

where $\lambda_{EP} = 0.001$, $\lambda_{GP} = 5$, and $\lambda_{CSE} = 0.1$. λ_{EP} is identical to Progressive Growing GAN [8]. λ_{GP} and λ_{CSE} are determined by a rough hyperparameter search, where we tested $\lambda_{CSE} \in [.1, .2, .5, 1, 2, 5, 10, 50]$ and $\lambda_{GP} \in [0.1, 0.5, 5, 10, 20, 100]$. We did the hyperparameter search only on Config B. No other hyperparameter search is done, unless stated otherwise.

Data Augmentation We use a limited amount of data augmentation, but find that it significantly improves quality of generated samples. We adapt the augmentation pipeline of StyleGAN2-ada [9] and modify the Pytorch implementation to support conditional image synthesis. The pipeline includes general geometrical transformations, color transformations, rotation, and horizontal flip. We significantly limit the amount of augmentation done, *e.g.* we rotate by a maximum of $9°$ left/right and translate by a maximum of 5% of the image width/height. This is to prevent augmentations leaking to our generator as we do not use adaptive augmentation training [9]. Following the Pytorch implementation, we use the following parameters; rotate=0.5, rotate_max=.05, xint=.5, xint_max=0.05, scale=.5, scale_std=.05, aniso=0.5, aniso_std=.05, xfrac=.5, xfrac_std=.05, brightness=.5, brightness_std=.05, contrast=.5, contrast_std=.1, hue=.5, hue_max=.05, saturation=.5, saturation_std=.5, imgfilter=.5, imgfilter_std=.1.

II. Face Quantitative Evaluation

The quantitative evaluation of the face region was done by cropping the face region, upsampling the region to $(299, 299)$ and compute FID for every training sample in the dataset. The face region was detected with DSFD [14] using an open source Pytorch implementation [6] where the highest scoring and largest face region was extracted.

III. DeepFashion CSE-Dataset

The DeepFashion CSE-dataset is derived from the Inshop Clothes Retrieval Benchmark of DeepFashion [16], where we have annotated each image with a CSE embedding. Each image is automatically annotated with a pretrained model from detectron2 [27], specifically R-101-FPN-DL-s1x. For each image, we select the highest scoring detected instance. We remove any image that has no detections with a confidence score larger than 0.8. The filtered dataset results in 40,625 training images and 10,275 validation images, where each image is downsampled to 384×256 using bilinear sampling. We will include the train/val split that we randomly selected.

DeepFashion Decoder-Only Architecture The decoder-only generator is similar to the generator of StyleGAN2 [11]. The generator consists of 6 residual blocks with two 3×3 convolutions each, where each block is followed by bilinear upsampling. In total, the generator has 43.5 M parameters, where each residual block has the following number of output channels: $[768, 768, 384, 192, 96, 48]$. The starting resolution is 12×8 and the output is 384×256.

We use the same discriminator as for COCO-Body, except that we scale the number of parameters to 42.4M. Otherwise, we use identical hyperparameters as for the COCO-Body dataset.

B. COCO Anonymization Details

The anonymization framework consists of two stages, detection and generation.

The detection network is a pre-trained CSE [19] network [1] from detectron2 [27]. We detect pixel-to-surface correspondences on the entire image (can contain several identities), then crop every person out. For each instance detected, we crop by first finding the minimum bounding box that contains the detected surface, then expand this surface to a rectangular shape that has a similar aspect ratio of the target resolution (288×160). In addition, we ensure that 30% of the resulting crop contains "background" (*i.e*. not the surface). We zero-pad the image if there is no possible crop that fits in the original image.

We crop the CSE-embedding similarly. The cropped image and the embedding is then resampled to 288×160 with bilinear up/downsampling. The segmentation map S outputted by CSE is 1 for every pixel belonging to the surface. We equally resample S and dilate S depending on the number of pixels in the image that belongs to the surface. This is to ensure that we remove clothing and other accessories on the human body, as CSE primarily detects the body. Then, we zero-out all pixels that are belongs to the surface, indicated by the dilated segmentation S. The resulting partial image is the input of our surface-guided generator.

The final image is naively stitched together. For each instance, the generated image is resampled to the original resolution, then all pixels in the original image belonging to the dilated surface is replaced by with the generated ones.

I. COCO-Body Dataset

We generate the COCO-Body dataset by using the detection procedure described above on COCO [15] train2014 and val2014. Specifically, for every image in the COCO dataset, we detect instances in the image. For each instance, we find the cropped image (described above) and include the image in the dataset if either; (1) the detected surface has more than 80% geodisic point similarity [4] to the ground truth DensePose-CSE dataset [19], or (2) the instance has a confidence score higher than 99.5%. Out of these detections, we filter out all images that are not in the aspect ratio range $[0.4, 4]$ (height/width), images that cover an area smaller than 144×80 pixels, or images that contains more than 25% zero-padding (in image area).

After filtering the detections, we are left with $43,053$ images from COCO train2014, and $10,777$ in val2014. We use this train/validation split for our experiments. The dataset is published open source at github.com/hukkelas/full_body_anonymization.

[1] Specifically, R-101-FPN-DL-s1x.

C. Additional Quantitative Results

Table 1 includes all metrics for every model that is reported in the main paper.

I. Truncation Trick

The truncation trick is an established technique to sample latent vectors from a truncated distribution to improve generated image quality at the cost of diversity of samples [1]. A similar strategy is adopted for the intermediate latent space of StyleGAN [10], which is shown to be more effective than truncating in z [13].

We adopt a similar strategy [10, 13] for the conditional latent space of V-SAM. We find the mass of center; $\bar{\omega_i} = \mathbb{E}_{z \sim P(z)}[f_\omega(\omega_i, z)]$. Then, we evaluate the effect of interpolation and clamping the latent by the following apporaches;

- **Interpolate** ω: $\bar{\omega} + t_i \cdot (\omega - \bar{\omega})$,
- **Clamp** ω: $\text{clamp}(\omega, \bar{\omega} - t_c, \bar{\omega} + t_c)$,
- **Interpolate** z: $t_i \cdot z$,
- **Clamp** z: $\text{clamp}(z, -t_c, t_c)$

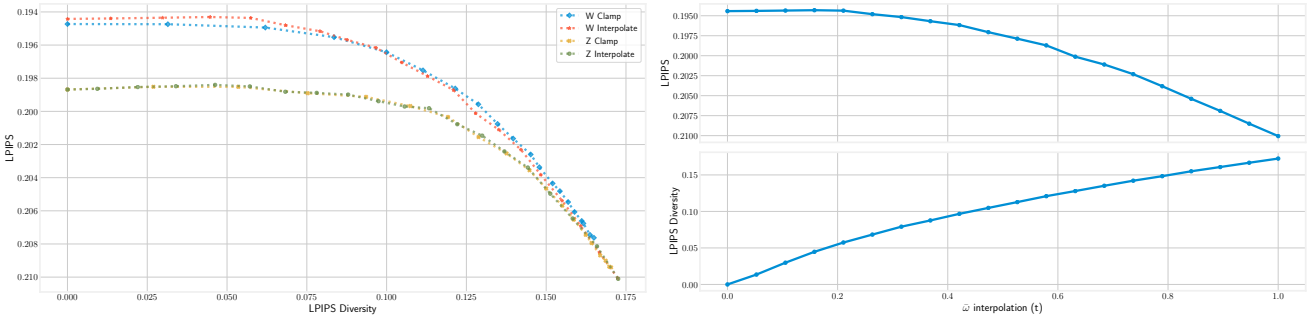
where $t_i \in \mathbb{R}, t_i \in [0, 1]$, and $\text{clamp}(x, x_{\text{lower}}, x_{\text{upper}})$ sets all elements of x that are outside the range of $[x_{\text{lower}}, x_{\text{upper}}]$ to x_{lower} or x_{upper}.

We observe that truncating the intermediate latent space ω yields a better tradeoff between quality and diversity, compared to truncating in z (Figure 1).

Interpolating in ω The disentangled representation of V-SAM allows for natural transitions between different synthesized styles. Figure 2 reflects that linearly interpolating between two randomly sampled ω yields a natural transition. Observe that the interpolation path is smooth, where different features of the synthesized person is gradually changed. Note that we observe a discontinuous transition (*e.g*. Row 5 in Figure 2) for some combinations of conditional pose and sampled ω. We speculate that a deeper mapping network can further improve the disentanglement and mitigate this non-smooth transition.

II. Effect of Anonymization for Computer Vision

Anonymizing Data for Evaluation (PASCAL VOC) Following the same approach as for COCO (described in the main paper), we show that surface-guided anonymization strongly improves over traditional techniques for evaluation purposes (Table 2). Furthermore, we note a slight gap between surface-guided anonymization and the original dataset (50.3 vs 51.8 AP). We believe this originates primarily from errors in detection, as we use a confidence threshold of 0.1.



(a) (b)

Figure 1: Analysis of truncation techniques for V-SAM. **(a)** Compares different truncation strategies applied to the intermediate latent space ω or z. Truncation in ω yields a better trade off between diversity and image quality. **(b)** Evaluates the effect of ω interpolation , where we interpolate a sampled latent towards $\bar{\omega}$. Increasing the truncation (t) gradually improves image quality at the cost of sample diversity.

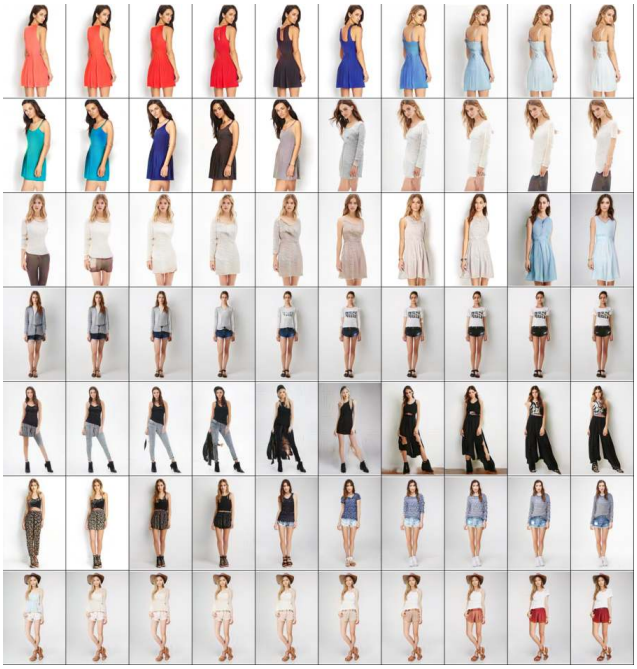


Figure 2: Example of latent interpolation in the intermediate latent space ω. The images are generated with linear interpolation between two randomly sampled ω, starting from 0 (leftmost column) to 1 (rightmost column). Each row is generated with different latents. No latent truncation is done.

Anonymizing data for Training Surface-guided anonymization slightly improves over traditional anonymization when using the anonymized data for training (Table 3, Table 4). Our naive anonymization of COCO without modifying object labels (*e.g.* not removing the class "tie") introduces ambiguities in the training objective, which we believe significantly degrades training performance. This issue is not as severe for pixelation, as these objects are still present in the image. Furthermore, we note that the CSE detector has several false positives and negatives which directly impacts the training objective. For example, false positives yields highly corrupted images with surface-guided anonymization.

Table 2: Object Detection AP on the PASCAL VOC validation set [2]. The results are from a **pre-trained** Faster R-CNN [23] R50-C4 from detectron2 [27].

Validation Dataset	$AP_{50:95}$ ↑	AP_{50} ↑	AP_{75} ↑
Original	51.8	80.3	56.5
Mask Out	47.9	75.2	51.4
8×8 Pixelation	47.2	75.1	50.8
16×16 Pixelation	48.5	76.7	52.0
Ours	**50.3**	**78.9**	**54.2**

D. Additional Qualitative Results

I. COCO-Body

Random examples from the COCO-Body dataset are given in Figure 3, Figure 4, Figure 5, and Figure 6.

II. DeepFashion

Random examples from the DeepFashion-CSE dataset are given in Figure 7, Figure 8, and Figure 9.

III. COCO

Random examples from the COCO dataset are given in Figure 10, Figure 11, Figure 12, Figure 13, Figure 14 and Figure 15.

Table 3: Instance segmentation mask AP on the COCO validation set [15]. The results are from a Mask R-CNN [5] R50-FPN-3x from detectron2 [27] trained on different anonymized datasets. The validation set is not anonymized.

Training Dataset	$AP_{50:95} \uparrow$	$AP_{50} \uparrow$	$AP_{75} \uparrow$	$AP_s \uparrow$	$AP_m \uparrow$	$AP_l \uparrow$	$AP_{Person} \uparrow$
Original	37.1	58.7	39.8	18.8	39.6	53.3	47.8
Mask Out	34.6	55.2	36.7	16.7	36.5	50.0	44.7
8×8 Pixelation	34.8	55.7	37.0	16.9	**36.7**	50.3	46.5
16×16 Pixelation	**35.0**	55.9	37.3	17.3	37.1	50.4	**46.9**
Ours	**35.0**	**56.0**	**37.5**	**17.7**	**36.7**	**50.6**	46.5

Table 4: Object Detection AP on the PASCAL VOC validation set [2]. The results are from a Faster R-CNN [23] R50-C4 from detectron2 [27] trained on different anonymized datasets. The validation set is not anonymized.

Training Dataset	$AP_{50:95} \uparrow$	$AP_{50} \uparrow$	$AP_{75} \uparrow$
Original	51.8	80.3	56.5
Mask Out	**51.3**	80.0	55.7
8×8 Pixelation	51.2	79.9	55.03
16×16 Pixelation	51.2	79.8	55.7
Ours	**51.3**	**80.1**	**56.0**

References

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*, 2019.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Riza Alp Guler, Natalia Neverova, and Iasonas Kokkinos. DensePose: Dense human pose estimation in the wild. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[6] Håkon Hukkelås. A high-performance pytorch implementation of face detection models, including retinaface and dsfd, 2020.

[7] Håkon Hukkelås, Frank Lindseth, and Rudolf Mester. Image inpainting with learnable feature imputation. *arXiv preprint arXiv:2011.01077*, 2020.

[8] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[9] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020.

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[11] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*, 2019.

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

[13] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, pages 3929–3938, 2019.

[14] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: Dual shot face detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[16] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[17] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[18] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018.

[19] Natalia Neverova, David Novotny, Marc Szafraniec, Vasil Khalidov, Patrick Labatut, and Andrea Vedaldi. Continuous surface embeddings. *Advances in Neural Information Processing Systems*, 33, 2020.

[20] Anton Obukhov, Maximilian Seitzer, Po-Wei Wu, Semen Zhydenko, Jonathan Kyl, and Elvis Yu-Jing Lin. High-fidelity performance metrics for generative models in pytorch, 2020. Version: 0.3.0, DOI: 10.5281/zenodo.4957738.

[21] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2019.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[25] Zhentao Tan, Menglei Chai, Dongdong Chen, Jing Liao, Qi Chu, Bin Liu, Gang Hua, and Nenghai Yu. Diverse semantic image synthesis via probability distribution modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7962–7971, 2021.

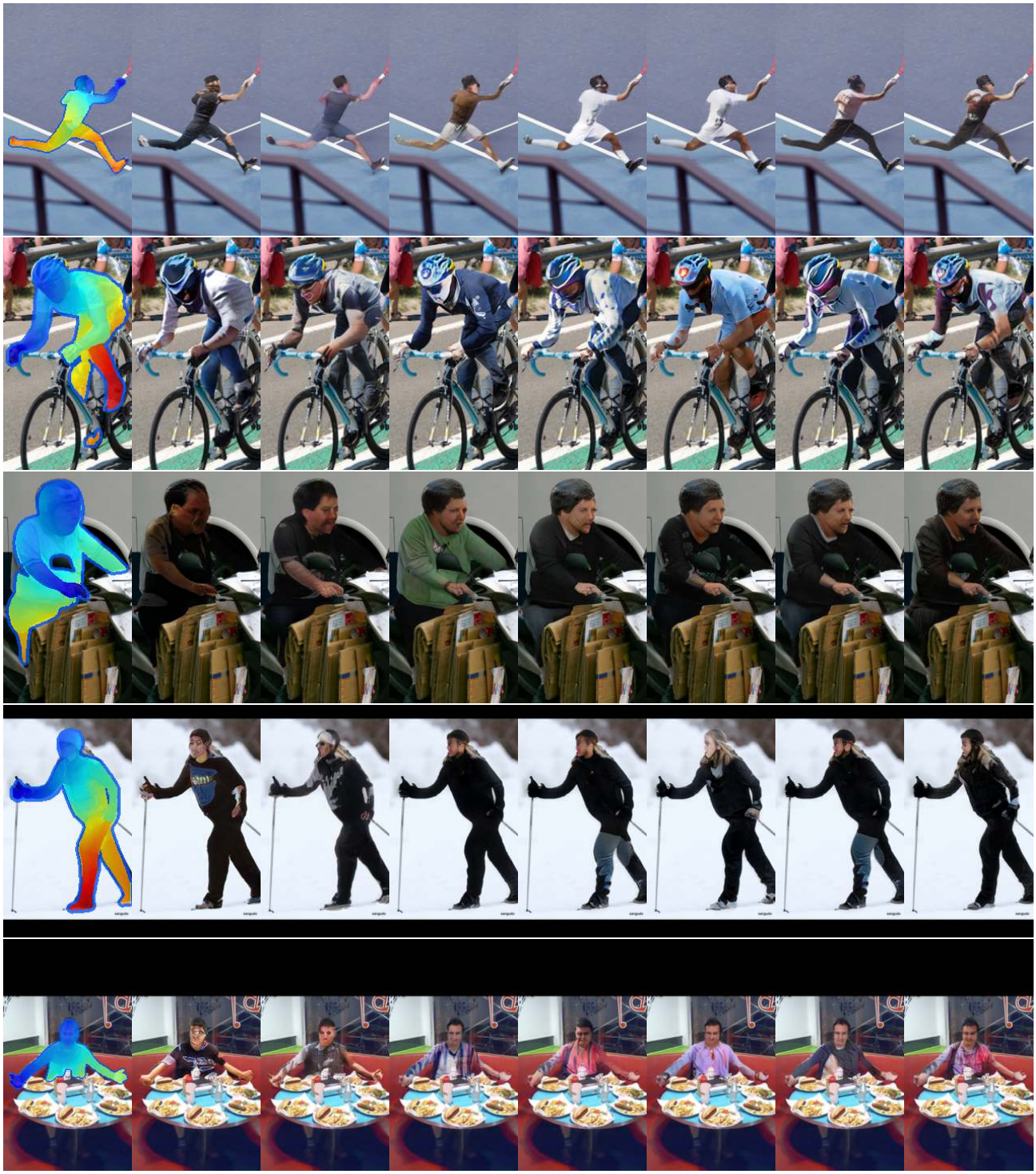
[26] Zhentao Tan, Dongdong Chen, Qi Chu, Menglei Chai, Jing Liao, Mingming He, Lu Yuan, Gang Hua, and Nenghai Yu. Efficient semantic image synthesis via class-adaptive normalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[27] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[28] Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *International Conference on Learning Representations*, 2019.

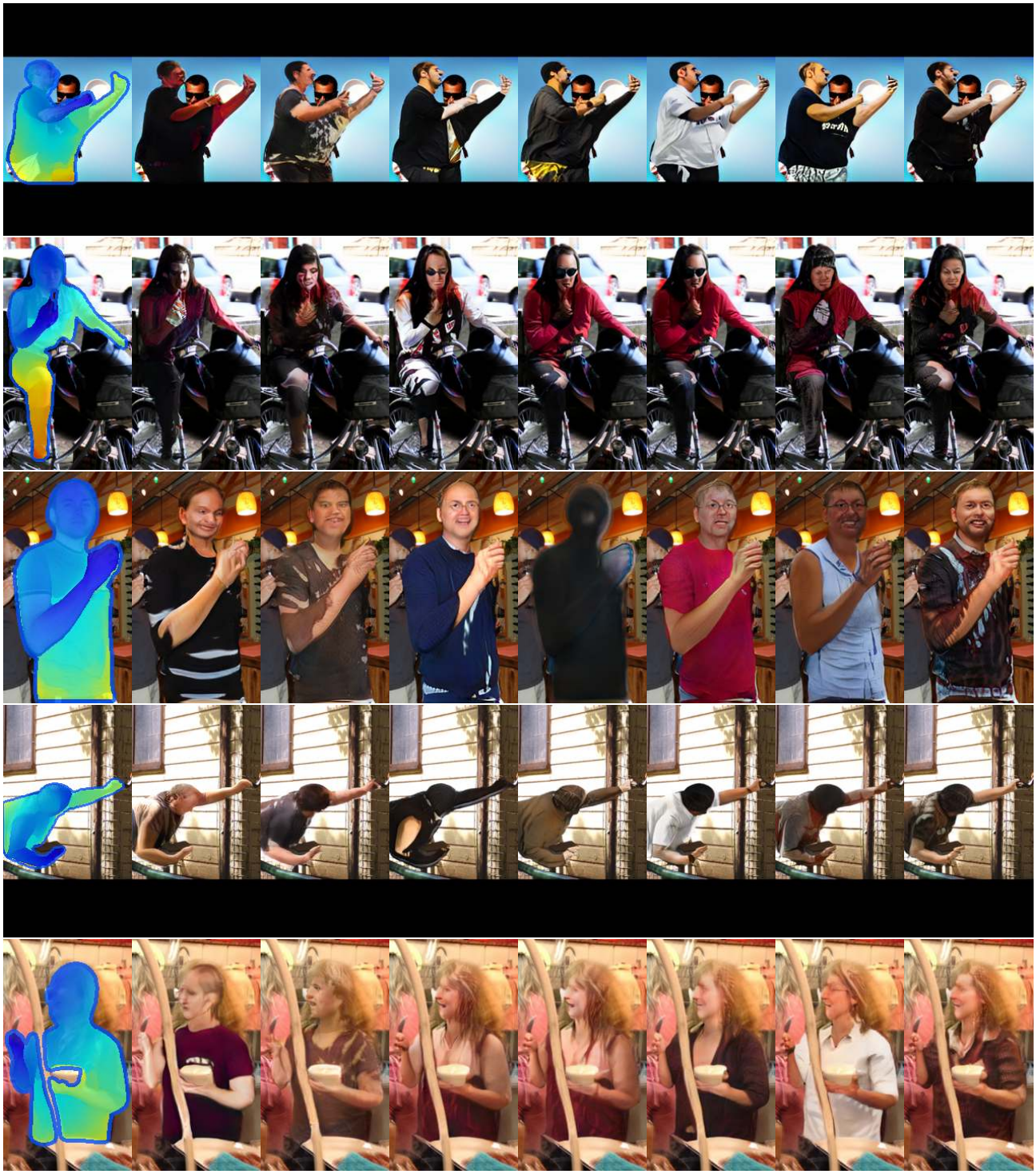
[29] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[30] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I-Chao Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. In *International Conference on Learning Representations*, 2021.



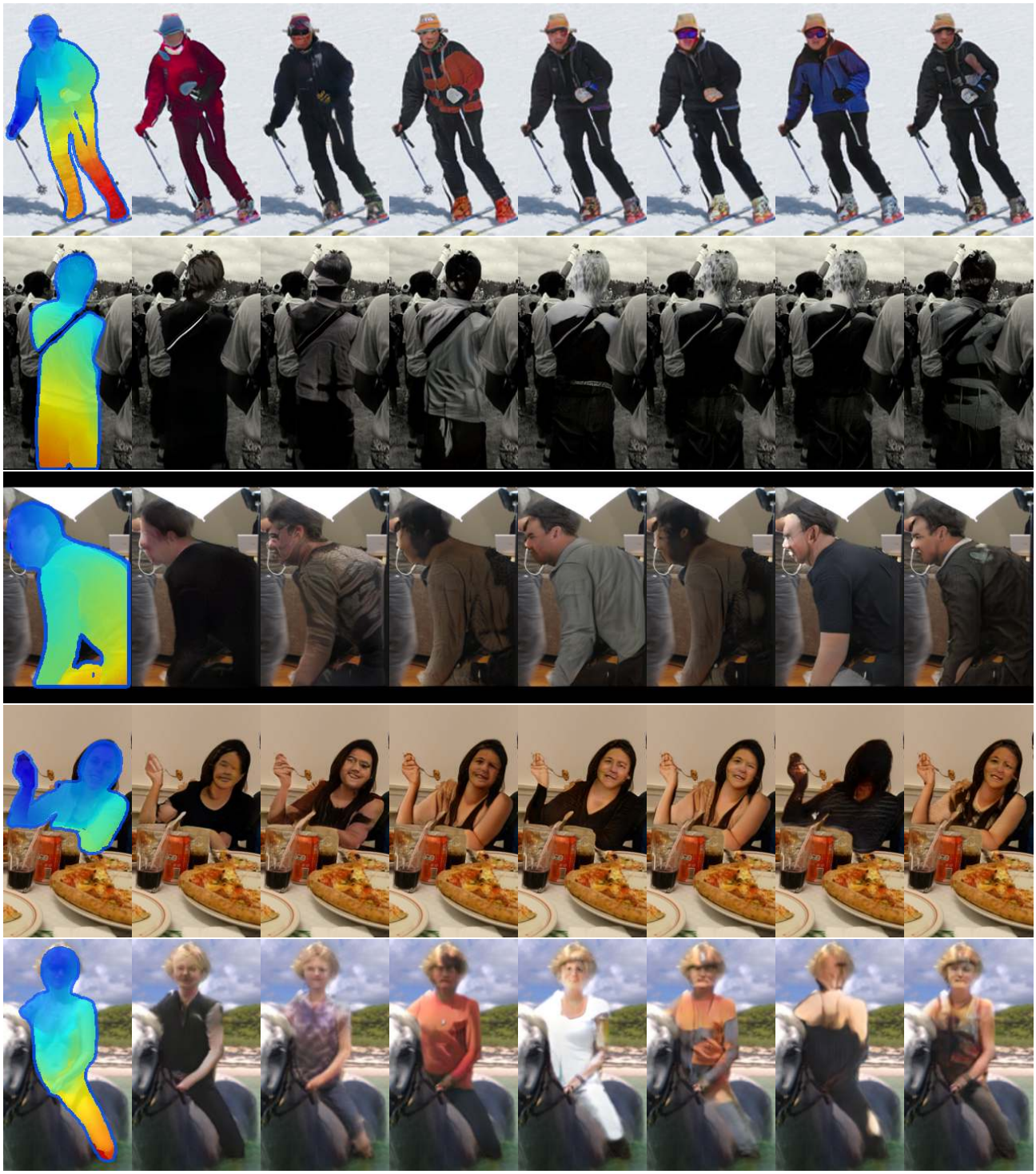
(a) Original (b) SPADE [21] (c) Config D (d) Config E (no truncation) (e) Config E (no truncation) (f) Config E (no truncation) (g) Config E (no truncation) (h) Config E (truncated)

Figure 3: Diverse synthesized images from COCO-Body. We truncate the normal of SPADE and Config D, and show diverse results for Config E. Note that the images are randomly sampled and the latent variable for (d-g) is randomly sampled with no latent truncation.



(a) Original (b) SPADE [21] (c) Config D (d) Config E (no truncation) (e) Config E (no truncation) (f) Config E (no truncation) (g) Config E (no truncation) (h) E (truncated)

Figure 4: Diverse synthesized images from COCO-Body. We truncate the normal of SPADE and Config D, and show diverse results for Config E. Note that the images are randomly sampled and the latent variable for (d-g) is randomly sampled with no latent truncation.



(a) Original (b) SPADE [21] (c) Config D (d) Config E (no truncation) (e) Config E (no truncation) (f) Config E (no truncation) (g) Config E (no truncation) (h) E (truncated)

Figure 5: Diverse synthesized images from COCO-Body. We truncate the normal of SPADE and Config D, and show diverse results for Config E. Note that the images are randomly sampled and the latent variable for (d-g) is randomly sampled with no latent truncation.

(a) Original (b) SPADE [21] (c) Config D (d) Config E (no truncation) (e) Config E (no truncation) (f) Config E (no truncation) (g) Config E (no truncation) (h) E (truncated)

Figure 6: Diverse synthesized images from COCO-Body. We truncate the normal of SPADE and Config D, and show diverse results for Config E. Note that the images are randomly sampled and the latent variable for (d-g) is randomly sampled with no latent truncation.

Original Truncated ω

Figure 7: Diverse synthesized images from DeepFashion-CSE. (b) shows generated results with a truncated latent, and all other columns use no truncation. Each column is synthesized with the same latent variable z. All images and latent variables are randomly sampled. Zoom in for details.

Original Truncated ω

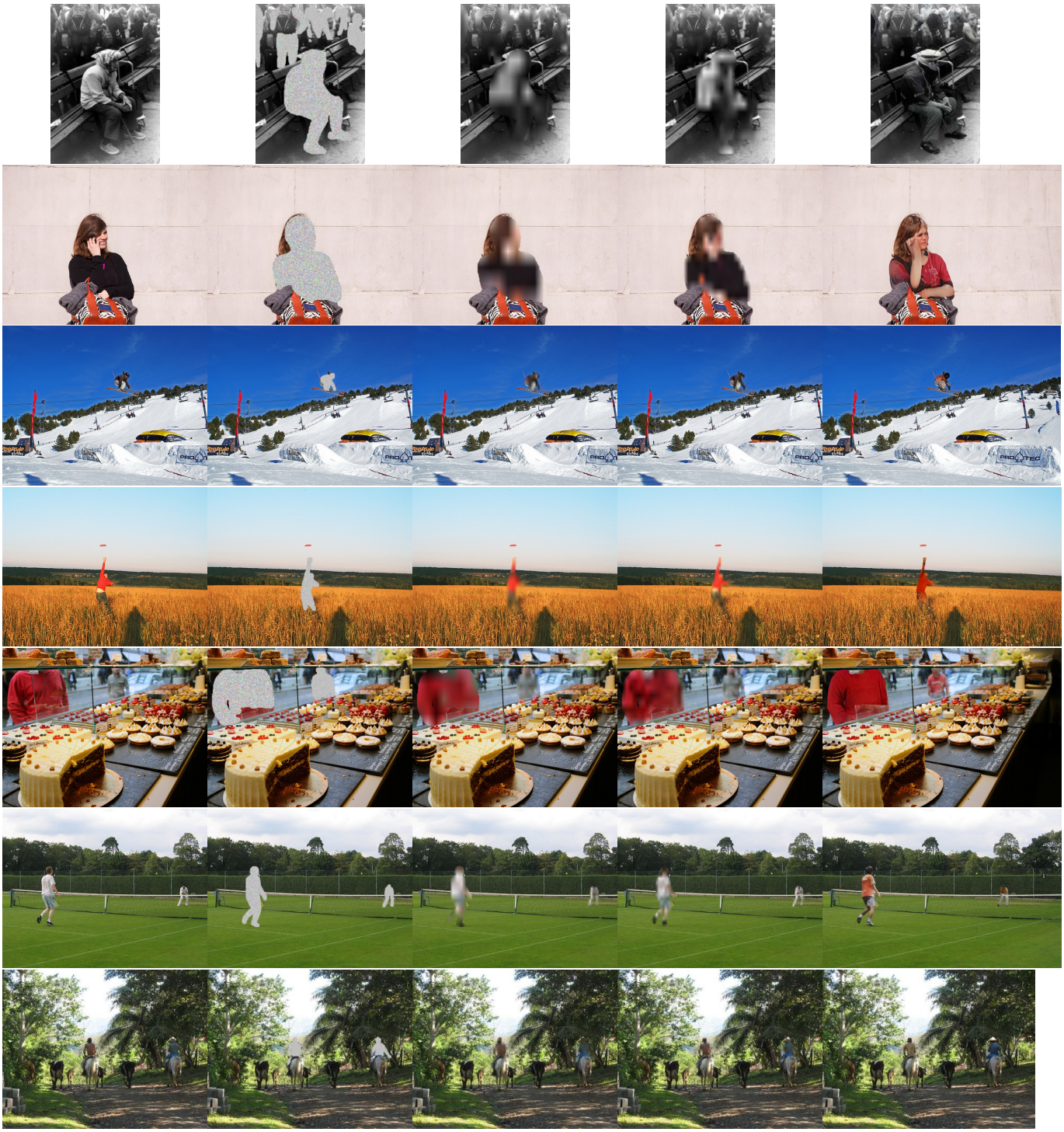
Figure 8: Diverse synthesized images from DeepFashion-CSE. (b) shows generated results with a truncated latent, and all other columns use no truncation. Each column is synthesized with the same latent variable z. All images and latent variables are randomly sampled. Zoom in for details.

Original Truncated ω

Figure 9: Diverse synthesized images from DeepFashion-CSE. (b) shows generated results with a truncated latent, and all other columns use no truncation. Each column is synthesized with the same latent variable z. All images and latent variables are randomly sampled. Zoom in for details.

Original | Mask Out | Pixelation 8×8 | Pixelation 16×16 | Ours

Figure 10: Randomly sampled images from the anonymized COCO [15] validation dataset. Each image is sampled with latent truncation (t=0), see Appendix C for details. Zoom in for details.



Original | Mask Out | Pixelation 8×8 | Pixelation 16×16 | Ours

Figure 11: Randomly sampled images from the anonymized COCO [15] validation dataset. Each image is sampled with latent truncation (t=0), see Appendix C for details. Zoom in for details.

Original | Mask Out | Pixelation 8×8 | Pixelation 16×16 | Ours

Figure 12: Randomly sampled images from the anonymized COCO [15] validation dataset. Each image is sampled with latent truncation (t=0), see Appendix C for details. Zoom in for details.

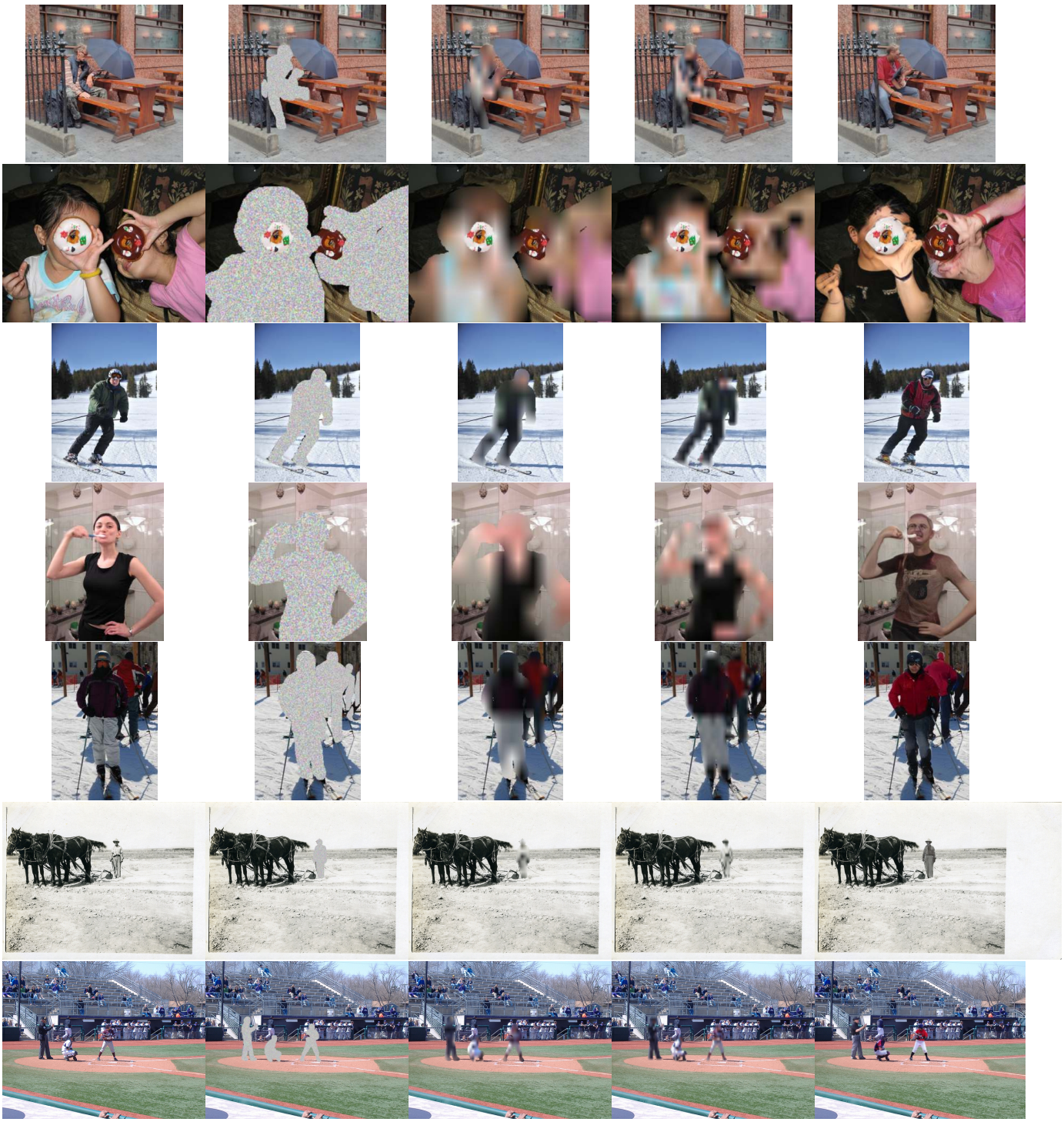
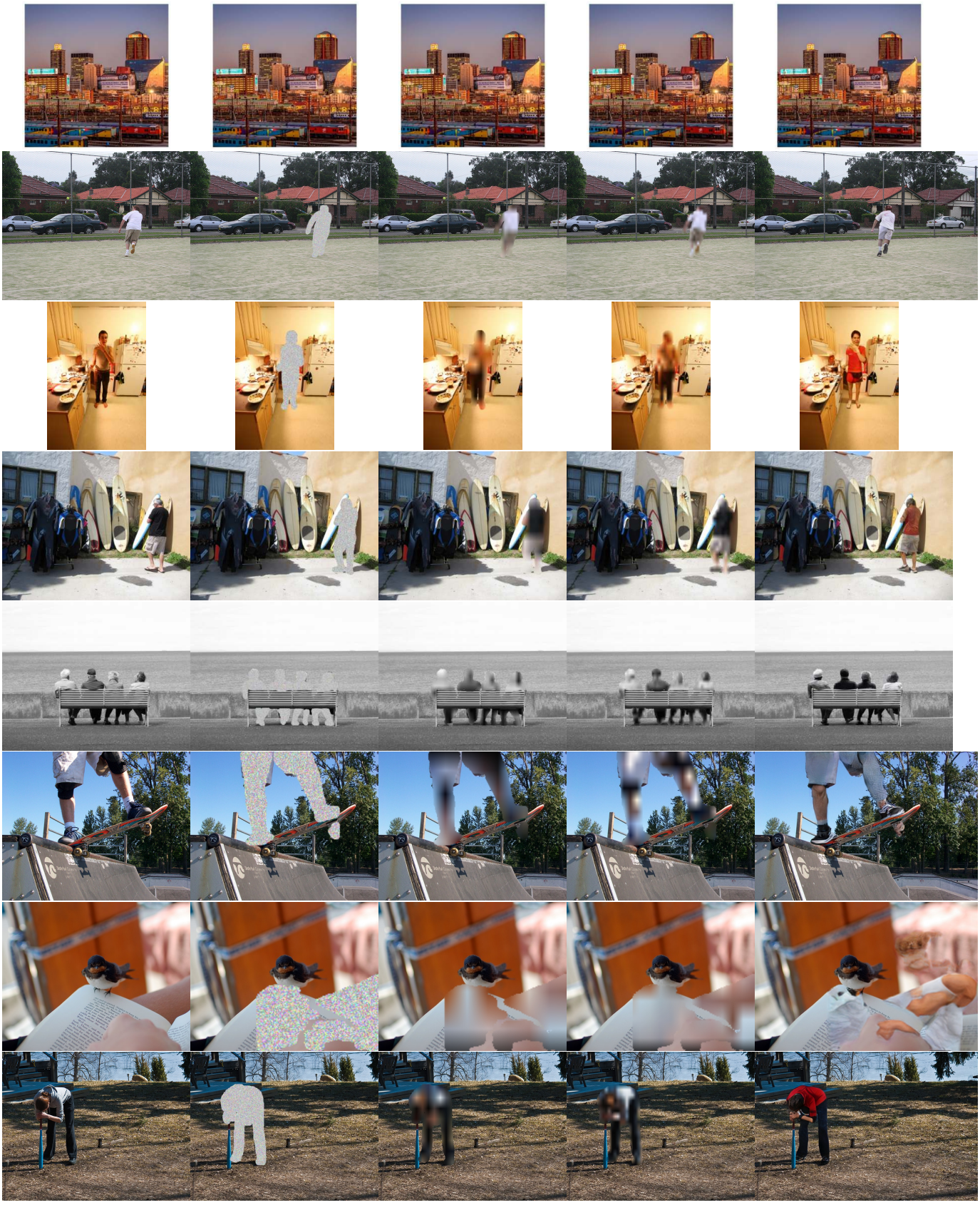
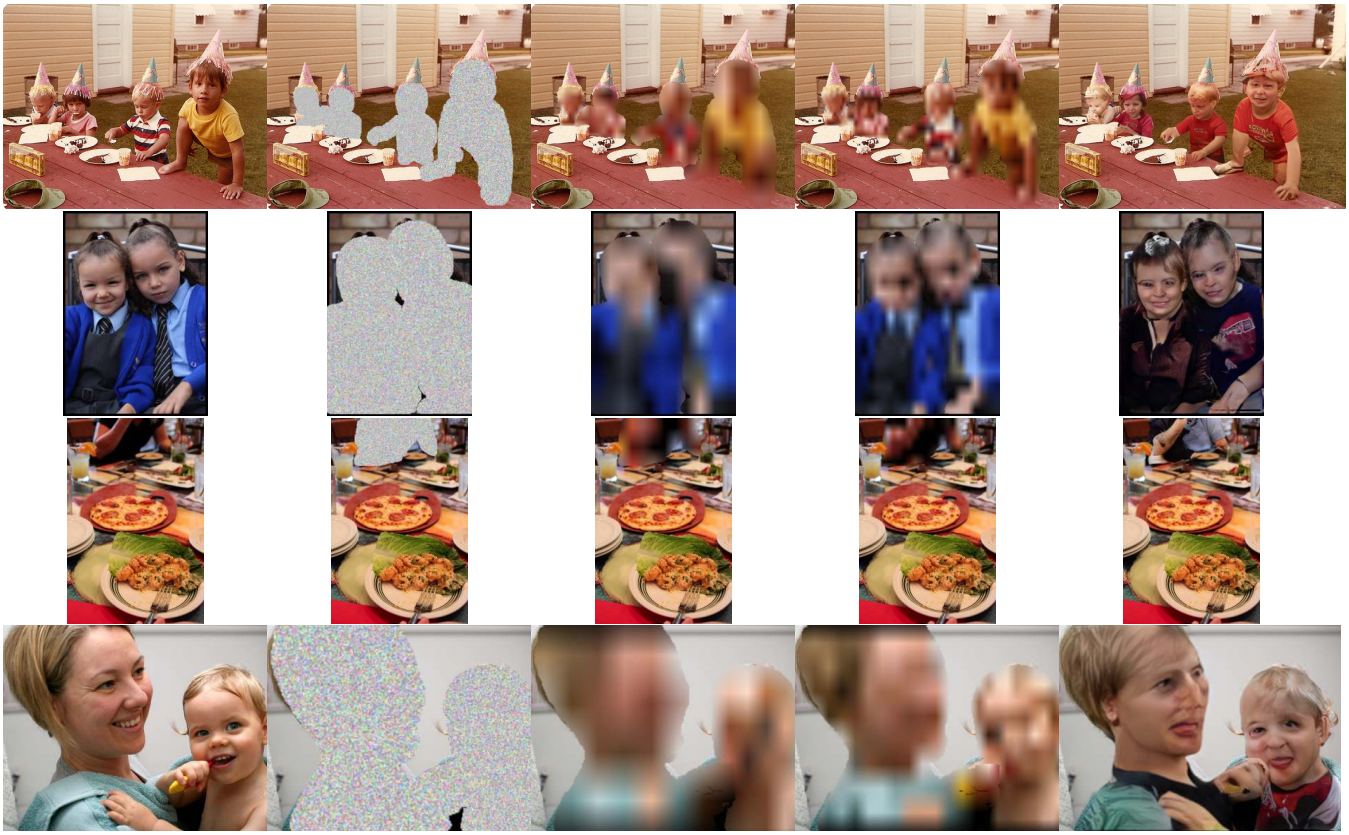


Figure 13: Randomly sampled images from the anonymized COCO [15] validation dataset. Each image is sampled with latent truncation (t=0), see Appendix C for details. Zoom in for details.



Original | Mask Out | Pixelation 8×8 | Pixelation 16×16 | Ours

Figure 14: Randomly sampled images from the anonymized COCO [15] validation dataset. Each image is sampled with latent truncation (t=0), see Appendix C for details. Zoom in for details.



Original | Mask Out | Pixelation 8×8 | Pixelation 16×16 | Ours

Figure 15: Randomly sampled images from the anonymized COCO [15] validation dataset. Each image is sampled with latent truncation (t=0), see Appendix C for details. Zoom in for details.