(Appendix) COCO-GAN: Generation by Parts via Conditional Coordinating

A. COCO-GAN during Testing Phase

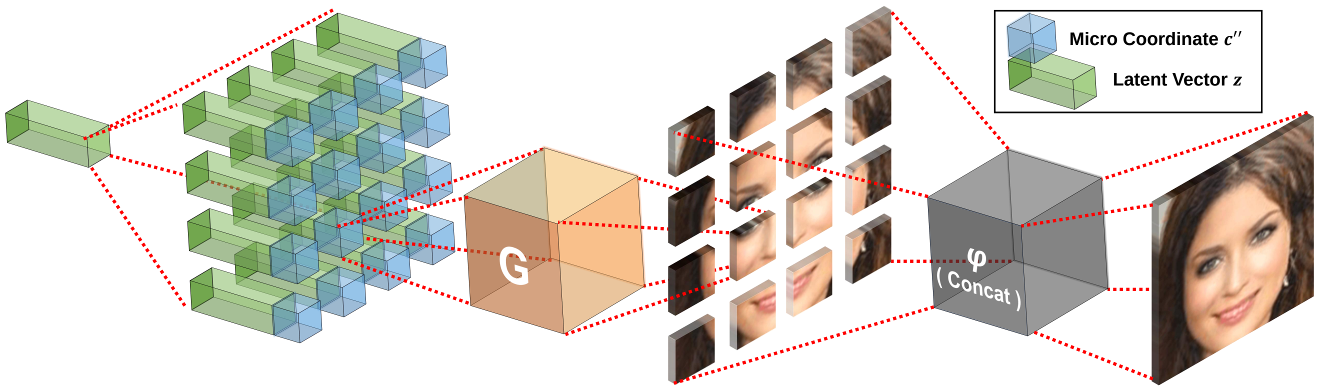


Figure 1: An overview of COCO-GAN during testing phase. The micro patches generated by G are directly combined into a full image as the final output.

B. Symbols

Group	Symbol	Name	Description	Usage
Model	G	Generator	Generates micro patches.	$s'' = G(z, c'')$
	D	Discriminator	Discriminates macro patches.	$D(\varphi(G(z, \boldsymbol{C''})))$
	A	Spatial prediction head	Predicts coordinate of a given macro patch.	$\hat{c}' = A(x')$
	Q	†Content prediction head	Predicts latent vector of a given macro patch.	$z_{est} = Q(s')$
Heuristic Function	φ	Merging function	Merges multiple s'' to form a s' or s.	$s' = \varphi(G(z, \boldsymbol{C''}))$
	ψ	Cropping function	Crops x' from x. Corresponding to φ.	$x' = \psi(x, c')$
Variable	z	Latent vector	Latent variable shared among s'' generation.	$s'' = G(z, c'')$
	z_{est}	†Predicted z	Predicted z of a given macro patch.	$L_Q = \mathbb{E}\,[\,\|z - z_{est}\|_1\,]$
	c'	Macro coordinate	Coordinate for macro patches on D side.	$L_S = \mathbb{E}\,[\,\|c' - \hat{c}'\|_2\,]$
	c''	Micro coordinate	Coordinate for micro patches on G side.	$s'' = G(z, c'')$
	$\hat{c}'$	Predicted c'	Coordinate predicted by A with a given x'.	$L_S = \mathbb{E}\,[\,\|c' - \hat{c}'\|_2\,]$
	$\boldsymbol{C''}$	Matrix of c''	The matrix of c'' used to generate $\boldsymbol{S''}$.	$s' = \varphi(G(z, \boldsymbol{C''}))$
Data	x	Real full image	Full resolution data, never directly used.	$x' = \psi(x, c')$
	x'	Real macro patch	A macro patch of x which D trains on.	$\text{adv}_{x'} = D(\psi(x, c'))$
	s'	Generated macro patch	Composed by s'' generated with $\boldsymbol{C''}$.	$\text{adv}_{s'} = D(s')$
	s''	Generated micro patch	Smallest data unit generated by G.	$s'' = G(z, c'')$
	$\boldsymbol{S''}$	Matrix of s''	Matrix of s'' generated by $\boldsymbol{C''}$.	$\boldsymbol{S''} = G(z, \boldsymbol{C''})$
	$\hat{s}'$	Interpolated macro patch	Interpolation between random x' and s'.	$\hat{s}' = \epsilon\, s' + (1-\epsilon)\, x'$, which $\epsilon \sim [0, 1]$
Loss	L_W	WGAN loss	The patch-level WGAN loss.	$L_W = \mathbb{E}\,[D(x')] - \mathbb{E}\,[D(s')]$
	L_{GP}	Gradient penalty loss	The gradient penalty loss to stabilize training.	$L_{GP} = \mathbb{E}\left[(\|\nabla_{\hat{s}'} D(\hat{s}')\|_2 - 1)^2\right]$
	L_S	Spatial consistency loss	Consistency loss of coordinates.	$L_S = \mathbb{E}\,[\|c' - A(x')\|_2]$
	L_Q	†Content consistency loss	Consistency loss of latent vectors.	$L_Q = \mathbb{E}\,[\|z - Q(s')\|_1]$
Hyper-parameter	α	Weight of L_S	Controls the strength of L_S (we use 100).	$\begin{cases} L_W + \lambda L_{GP} + \alpha L_S, & \text{for } D, \\ -L_W + \alpha L_S, & \text{for } G. \end{cases}$
	λ	Weight of L_{GP}	Controls the strength of L_{GP} (we use 10).	
Testing Only	s	Generated full image	Composed by s'' generated with $\boldsymbol{C''_{Full}}$.	$s = \varphi(G(z, \boldsymbol{C''_{Full}}))$
	$\boldsymbol{C''_{Full}}$	Matrix of c'' for testing	The matrix of c'' used during testing.	$s = \varphi(G(z, \boldsymbol{C''_{Full}}))$

† Only used in "Patch-Guided Image Generation" application.

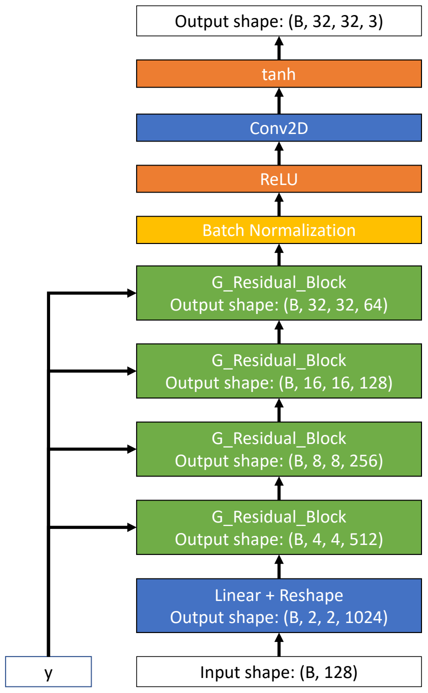
C. Experiments Setup and Model Architecture Details

Architecture. Our G and D design uses projection discriminator [6] as the backbone and adding class-projection to the discriminator. All convolutional and feed-forward layers of generator and discriminator are added with the spectral-normalization [7] as suggested in [9]. Detailed architecture diagram is illustrated in Figure 2 and Figure 3. Specifically, we directly duplicate/remove the last residual block if we need to enlarge/reduce the size of output patch. However, for (N8,M8,S8) and (N16,M16,S4) settings, since the model becomes too shallow, we keep using (N4,M4,S16) architecture, but without strides in the last one and two layer(s), respectively.

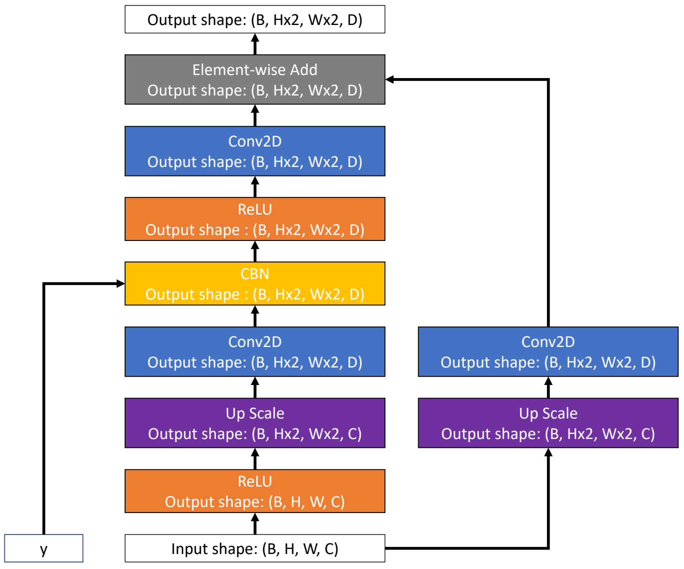
Conditional Batch Normalization (CBN). We follow the projection discriminator that employs CBN [3, 2] in the generator. The concept of CBN is to normalize, then modulate the features by conditionally produce γ and β that used in conventional batch normalization, which computes $o_K = ((i_K - \mu_K)/\sigma_K) * \gamma + \beta$ for the K-th input feature i_K, output feature o_K, feature mean μ_K and feature variance σ_K. However, in the COCO-GAN setup, we provide both spatial coordinate and latent vector as conditional inputs, which both having real values instead of common discrete classes. As a result, we create two MLPs, $\text{MLP}_\gamma(z, c)$ and $\text{MLP}_\beta(z, c)$, for each CBN layer, that conditionally produces γ and β.

Hyperparameters. For all the experiments, we set the gradient penalty weight $\lambda = 10$ and auxiliary loss weight $\alpha = 100$. We use Adam [5] optimizer with $\beta_1 = 0$ and $\beta_2 = 0.999$ for both the generator and the discriminator. The learning rates are based on the Two Time-scale Update Rule (TTUR) [4], setting 0.0001 for the generator and 0.0004 for the discriminator as suggested in [9]. We do not specifically balance the generator and the discriminator by manually setting how many iterations to update the generator once as described in the WGAN paper [1].

Coordinate Setup. For the micro coordinate matrix $\boldsymbol{C}''_{(i,j)}$ sampling, although COCO-GAN framework supports real-valued coordinate as input, however, with sampling only the discrete coordinate points that is used in the testing phase will result in better overall visual quality. As a result, all our experiments select to adopt such discrete sampling strategy. We show the quantitative degradation in the ablation study section. To ensure that the latent vectors z, macro coordinate conditions c', and micro coordinate conditions c'' share the similar scale, which z and c'' are concatenated before feeding to G. We normalize c' and c'' values into range $[-1, 1]$, respectively. For the latent vectors z sampling, we adopts uniform sampling between $[-1, 1]$, which is numerically more compatible with the normalized spatial condition space.

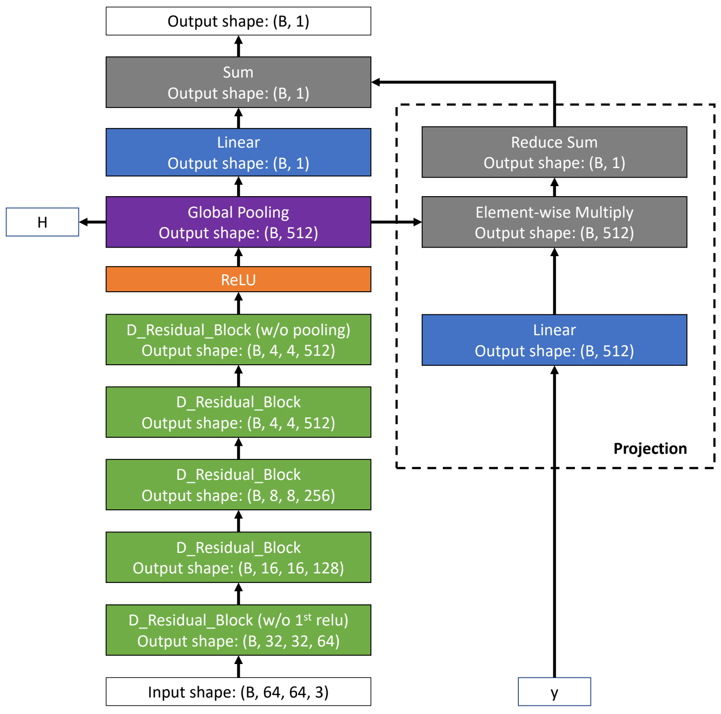

Output shape: (B, 32, 32, 3)
tanh
Conv2D
ReLU
Batch Normalization
G_Residual_Block Output shape: (B, 32, 32, 64)
G_Residual_Block Output shape: (B, 16, 16, 128)
G_Residual_Block Output shape: (B, 8, 8, 256)
G_Residual_Block Output shape: (B, 4, 4, 512)
Linear + Reshape Output shape: (B, 2, 2, 1024)
y
Input shape: (B, 128)

(a) Generator Overall Architecture

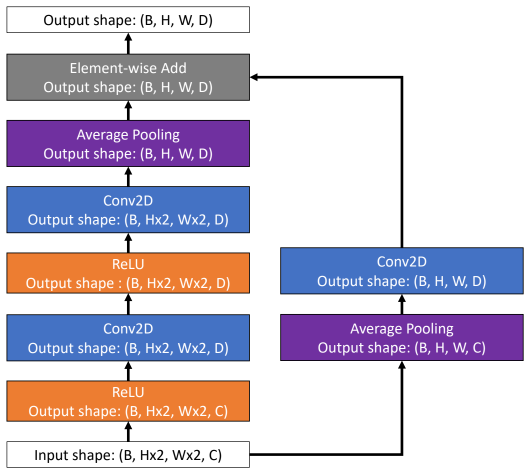

Output shape: (B, Hx2, Wx2, D)
Element-wise Add Output shape: (B, Hx2, Wx2, D)
Conv2D Output shape: (B, Hx2, Wx2, D)
ReLU Output shape : (B, Hx2, Wx2, D)
CBN Output shape : (B, Hx2, Wx2, D)
Conv2D Output shape: (B, Hx2, Wx2, D)
Conv2D Output shape: (B, Hx2, Wx2, D)
Up Scale Output shape: (B, Hx2, Wx2, C)
Up Scale Output shape: (B, Hx2, Wx2, C)
ReLU Output shape: (B, H, W, C)
y
Input shape: (B, H, W, C)

(b) Generator Residual Block

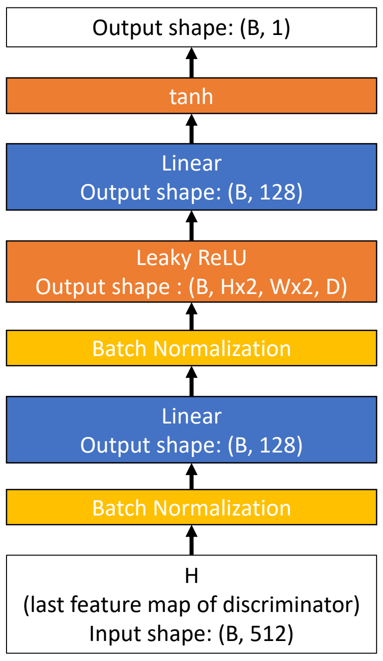
Figure 2: The detailed generator architecture of COCO-GAN for generating micro patches with a size of 32×32 pixels.



(a) Discriminator Overall Architecture


Output shape: (B, H, W, D)
Element-wise Add
Output shape: (B, H, W, D)
Average Pooling
Output shape: (B, H, W, D)
Conv2D
Output shape: (B, Hx2, Wx2, D)
ReLU
Output shape : (B, Hx2, Wx2, D)
Conv2D
Output shape: (B, Hx2, Wx2, D)
ReLU
Output shape: (B, Hx2, Wx2, C)
Input shape: (B, Hx2, Wx2, C)
Conv2D
Output shape: (B, H, W, D)
Average Pooling
Output shape: (B, H, W, C)

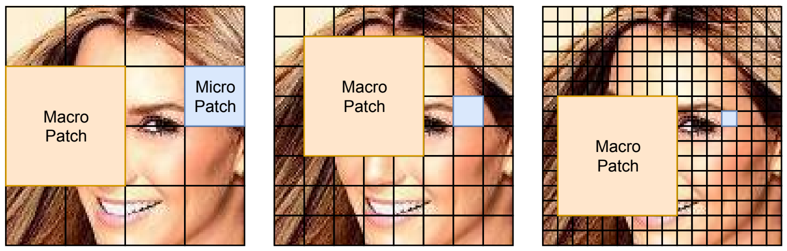
(b) Discriminator Residual Block



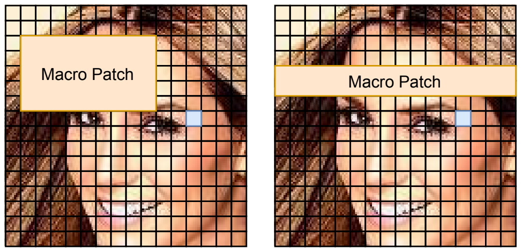
(c) Discriminator Auxiliary Head

Figure 3: The detailed discriminator architecture of COCO-GAN for discriminate macro patches with a size of 64×64 pixels. Both the content vector prediction head (Q) and the spatial condition prediction head use the same structure shown in (c).

D. Example of Coordinate Design



(a) Implementations used in this paper with (Left) P4x4, (Middle) P8x8 and (Right) P16x16.



(b) Other possible implementations (not used in this paper).

Figure 4: We showcase some of the coordinate systems: (a) implementations we used in our experiments, and (b) some of the other possible implementations. For instance, 3D cubic data may choose to use each of its face as a macro patch. Also, a recent work [8] shows that horizontal tiles are naturally suitable for indoor layout task on panoramas, which points out that using horizontal tiles as macro patch in panorama generation may be an interesting future direction.

E. Beyond-Boundary Generation: More Examples and Details of Post-Training

We show more examples of "Beyond-Boundary Generation" in Figure 6.

Directly train with coordinates out of the $[-1, 1]$ range (restricted by the real full images) is infeasible, since there is no real data at the coordinates outside of the boundary, thus the generator can exploit the discriminator easily. However, interestingly, we find extrapolating the coordinates of a manually trained COCO-GAN can already produce contents that seemingly extended from the edge of the generated full images (*e.g.*, Figure 5).

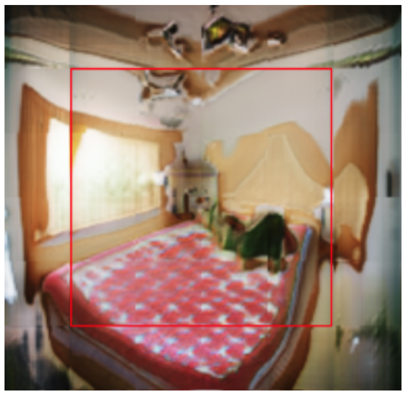


Figure 5: Without ***any extra*** training, original COCO-GAN can already perform slight extrapolations (*i.e.* the edge of the bed extends out of the normal generation area annotated with the red box), however, expectedly discontinuous on the edges.

With such an observation, we select to perform additional post-training on the checkpoint(s) of manually trained COCO-GAN (*e.g.*, (N4,M4,S64) variant of COCO-GAN that trained on LSUN dataset for 1 million steps with resolution 256×256 and a batch size 128). Aside from the original Adam optimizer that trains COCO-GAN with coordinates $\in [-1, 1]$, we create another Adam optimizer with the default learning rate setup (*i.e.*, 0.0004 for D and 0.0001 for G). The additional optimizer trains COCO-GAN with additional coordinates along with the original coordinates. For instance, in our experiments, we extend an extra micro patch out of the image boundary, as a result, we train the model with $c'' \in \left[-1.\overline{66}, 1.\overline{66}\right]$ (the distance between two consecutive micro patches is $2/(4-1) = 0.\overline{66}$) and $c' \in [-2, 2]$ (the distance between two consecutive micro patches is $2/((4-1)-1) = 1$). We only use the new optimizer to train COCO-GAN until the discontinuity becomes patches becomes invisible. Note that we do not train the spatial prediction head A with coordinates out of $[-1, 1]$, since our original model has a tanh activation function on the output of A, which is impossible to produce predictions out of the range of $[-1, 1]$.

We empirically observe that by only training the first-two layers of the generator (while the whole discriminator at the same time) can largely stabilize the post-training process. Otherwise, the generator will start to produce weird and mottled artifacts. As the local textures are handled by later layers of the generator, we decide to freeze all the later layers and only train the first-two layers, which controls the most high-level representations. We flag the more detailed investigation on the root-cause of such an effect and other possible solutions as interesting future research direction.

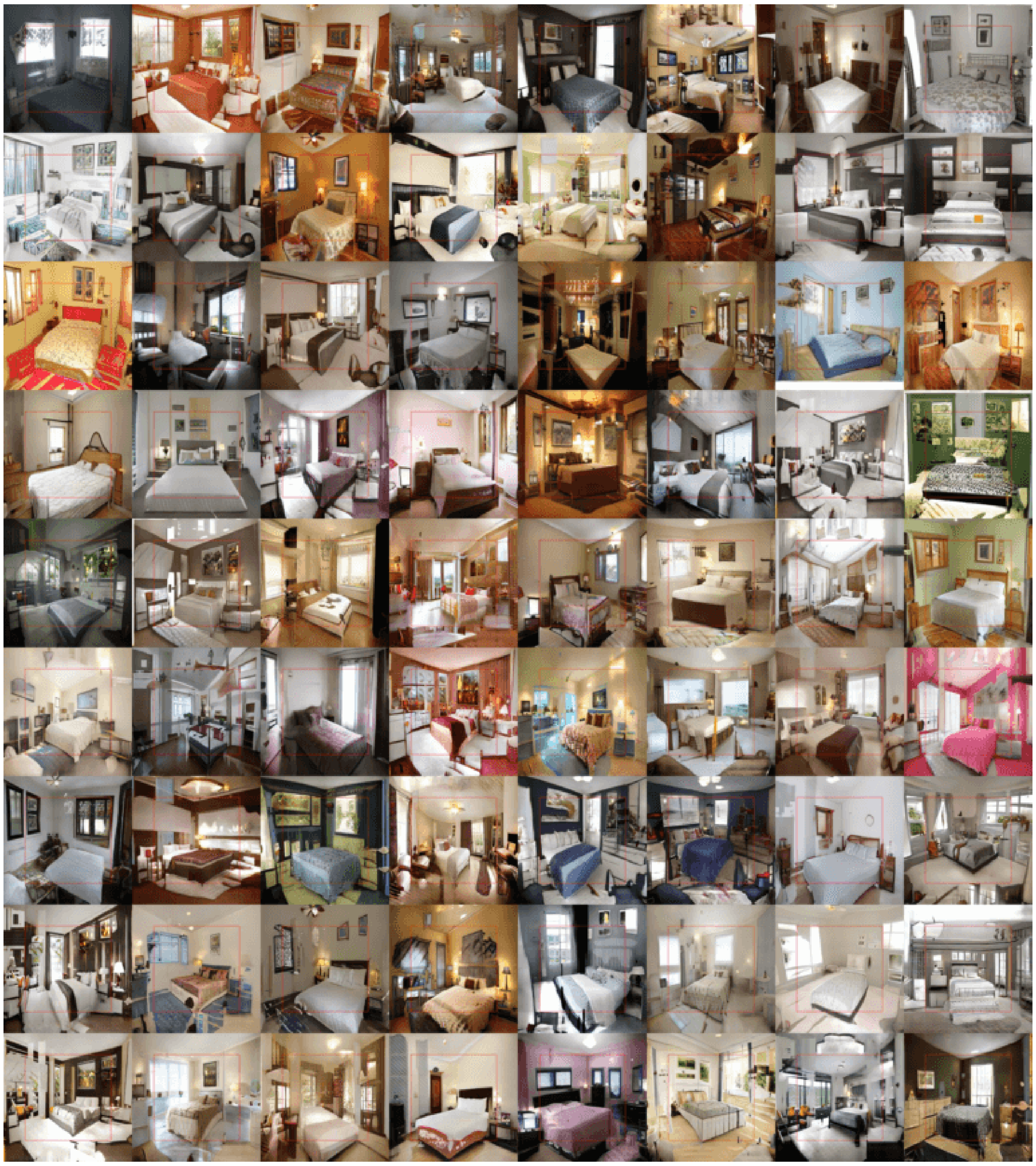
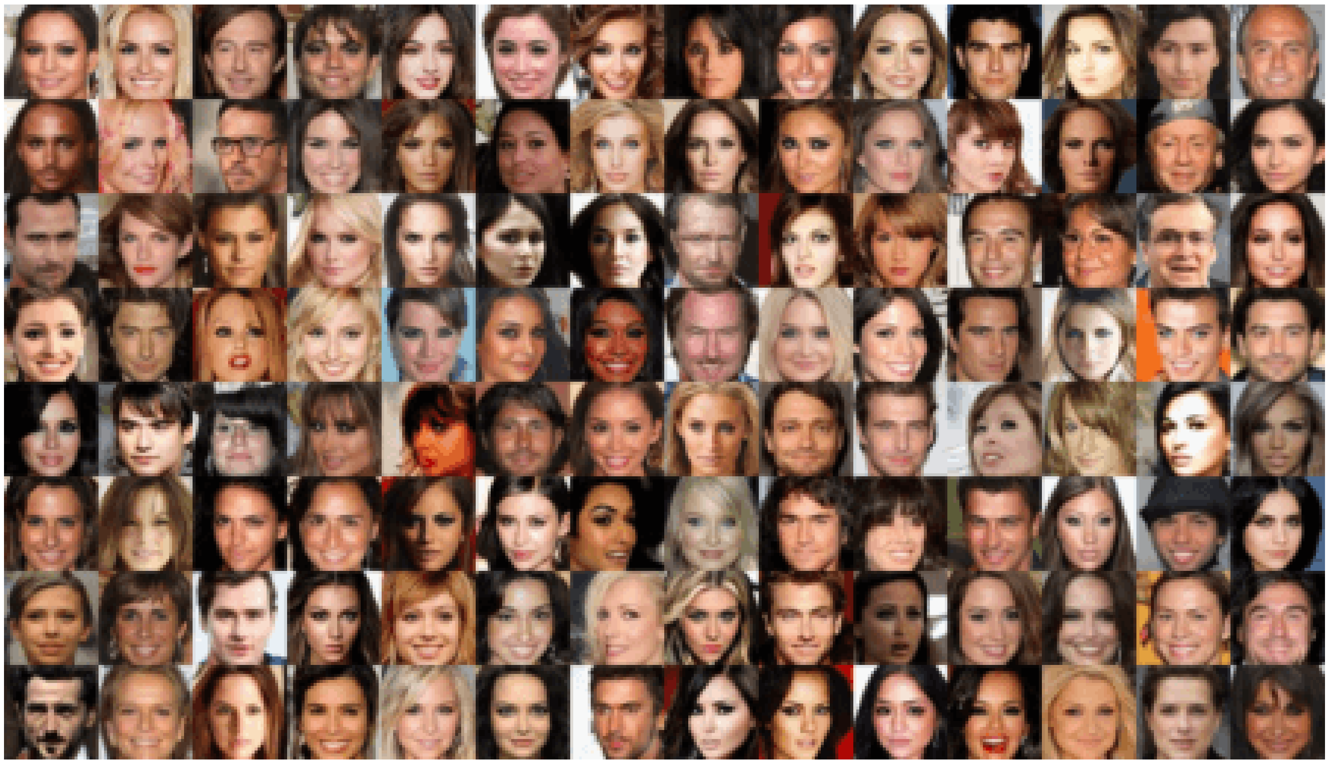
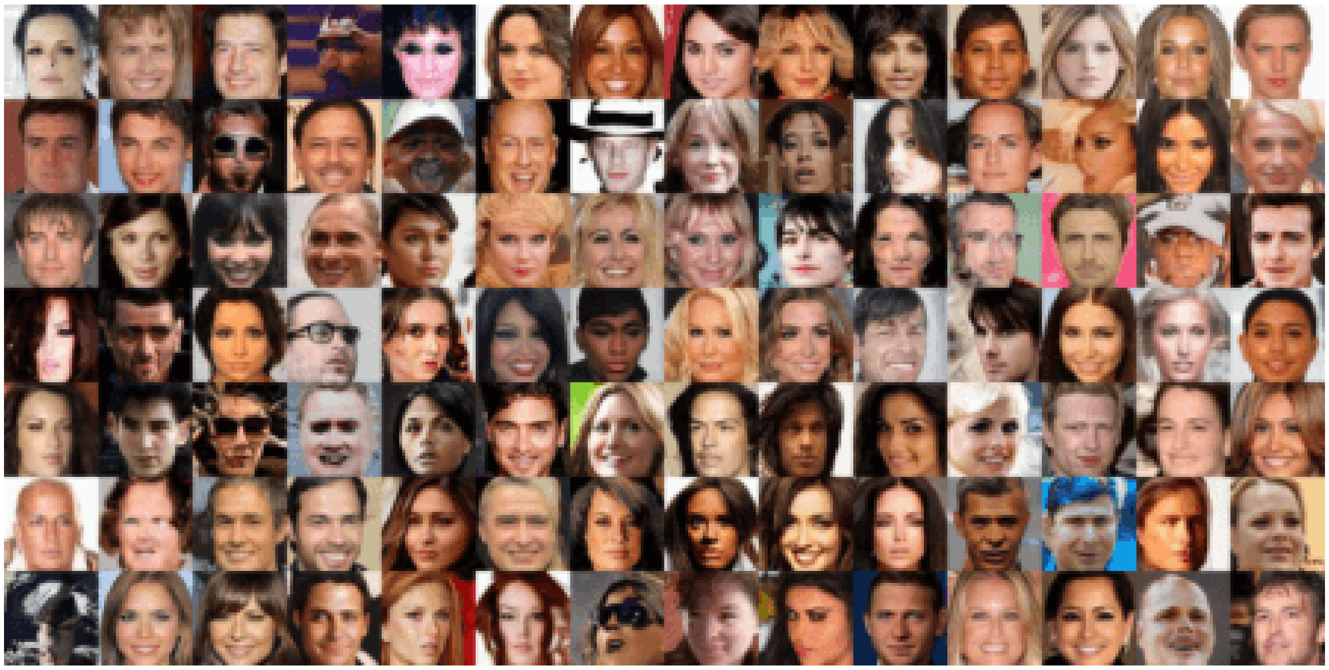


Figure 6: “Beyond-Boundary Generation” generates additional contents by extrapolating the learned coordinate manifold. Note that the generated samples are 384×384 pixels, whereas ***all*** of the training samples are of a smaller 256×256 resolution. The red box annotates the 256×256 region for regular generation without extrapolation.

F. More Full Image Generation Examples


(a) Selected generation samples.


(b) Random generation without calibration.

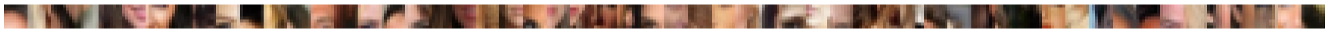
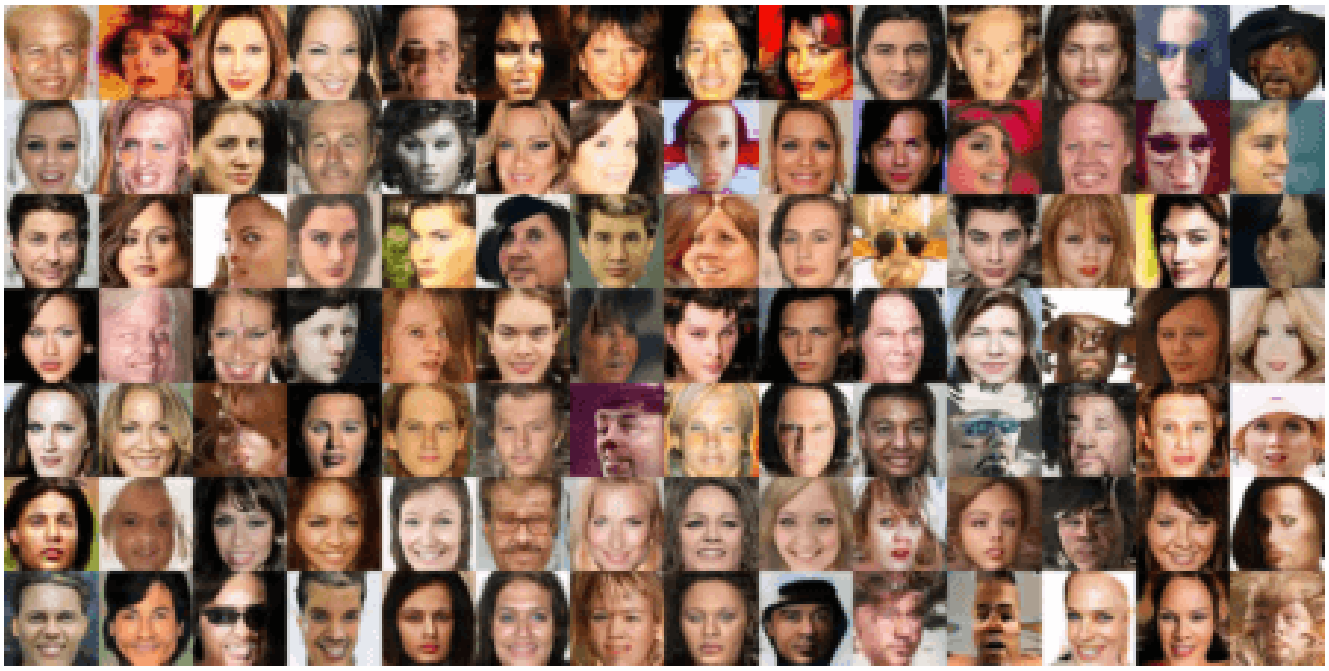

(c) Generated micro patches.

Figure 7: Full images generated by COCO-GAN on CelebA 128×128 with (N2,M2,S32) setting.

Due to file size limit, all images are compressed, please access the full resolution pdf from: https://goo.gl/5HLynv

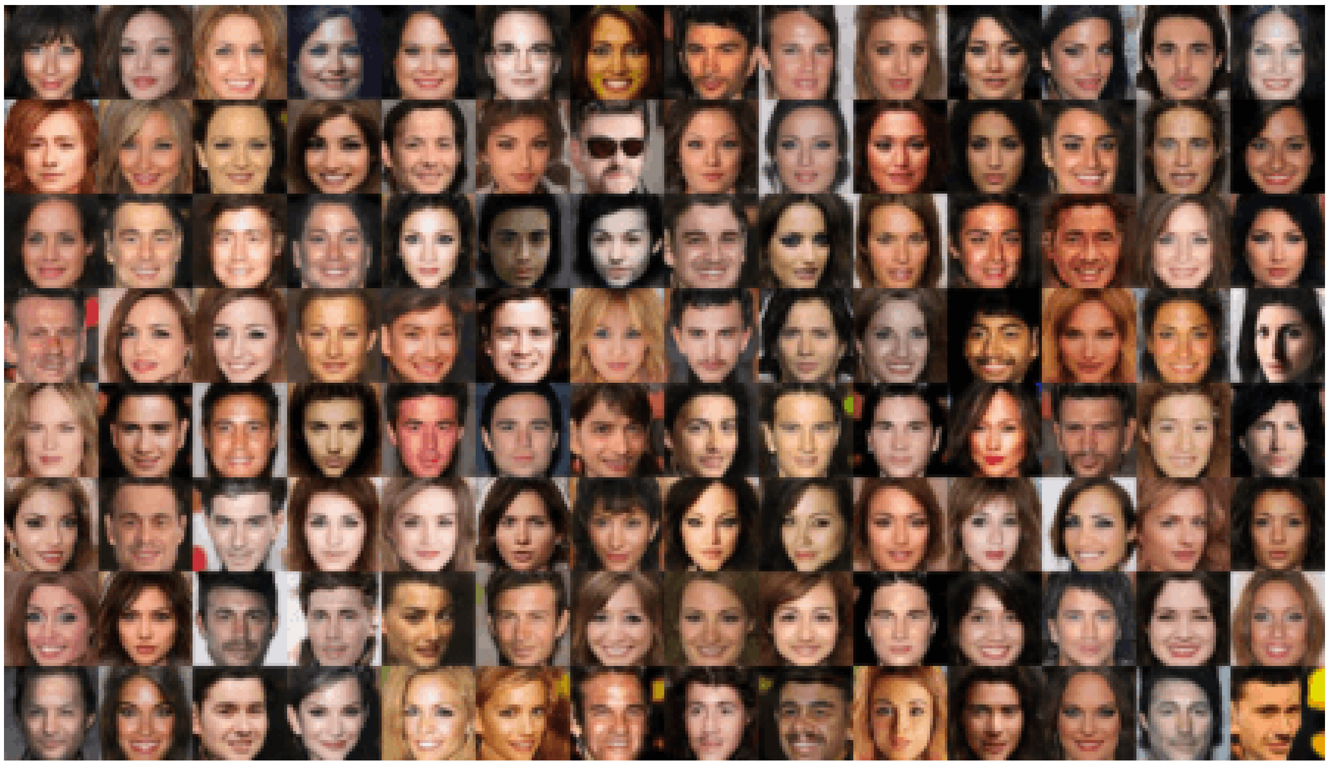
(a) Selected generation samples.

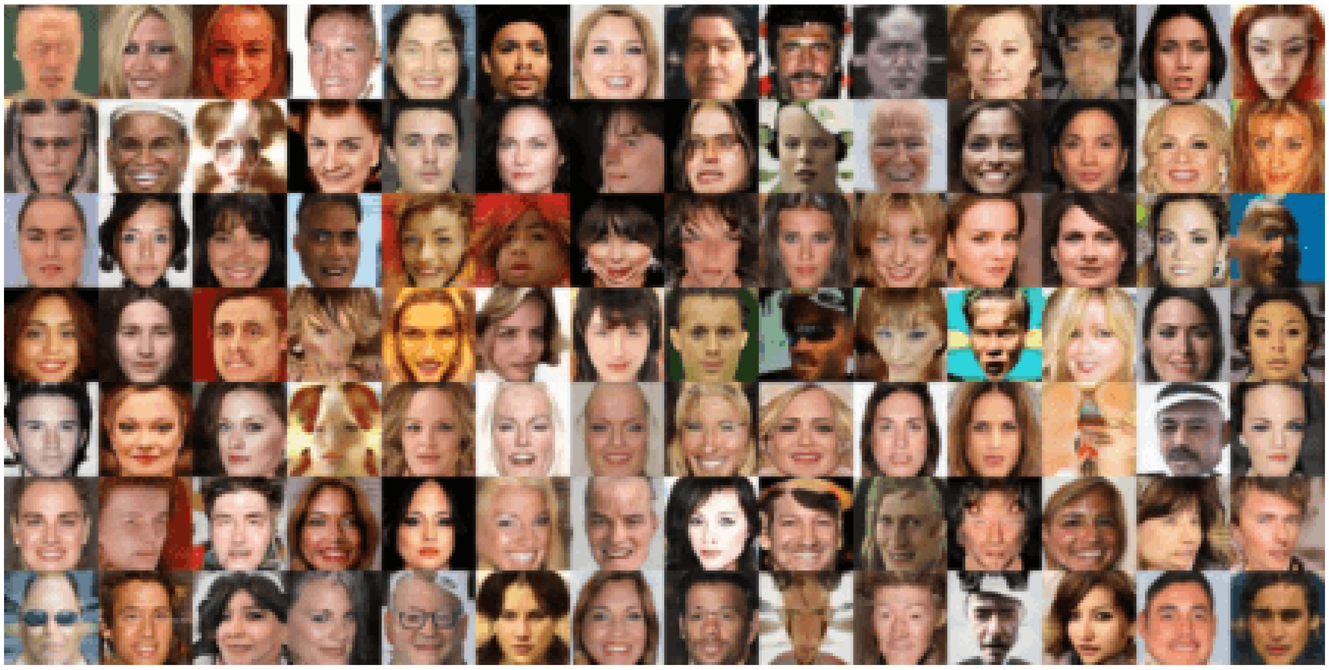

(b) Random generation without calibration.

(c) Generated micro patches.

Figure 8: Full images generated by COCO-GAN on CelebA 128 × 128 with (N4,M4,S16) setting.

Due to file size limit, all images are compressed, please access the full resolution pdf from: https://goo.gl/5HLynv

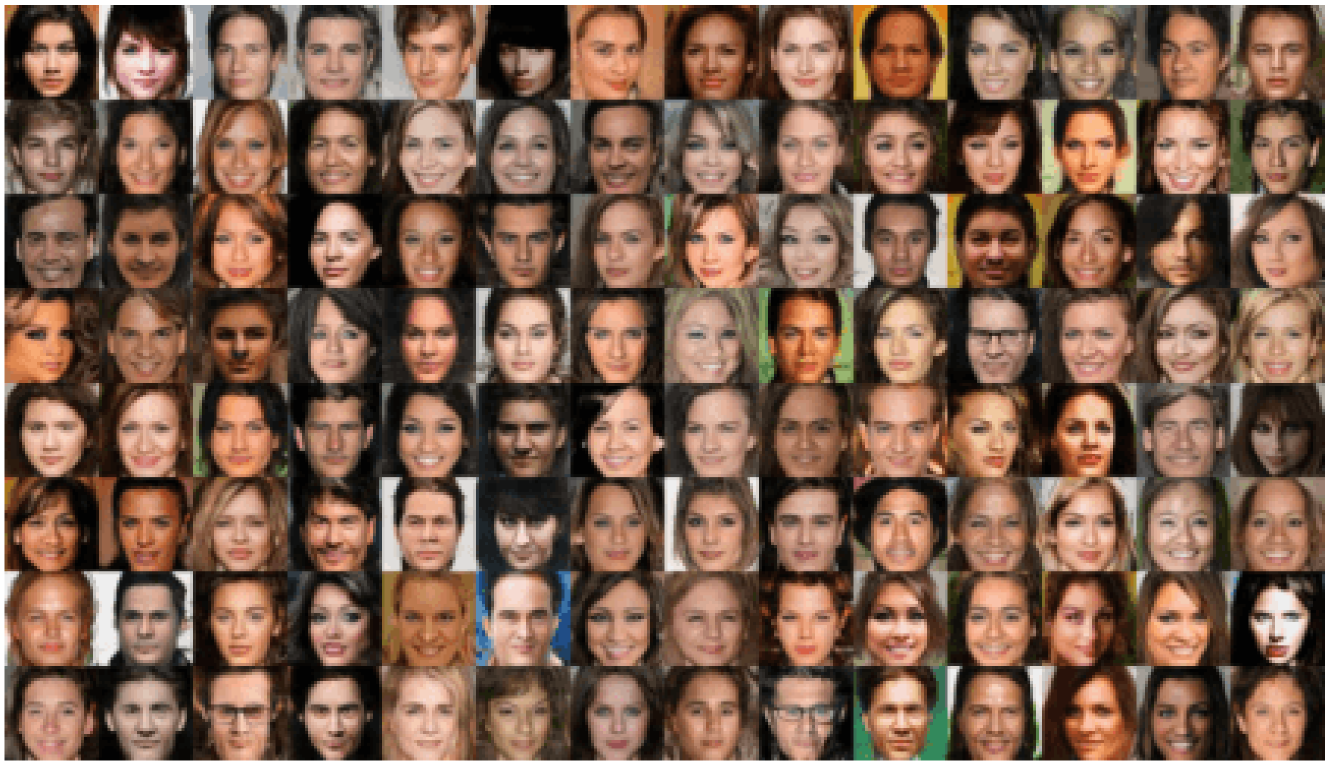

(a) Selected generation samples.


(b) Random generation without calibration.

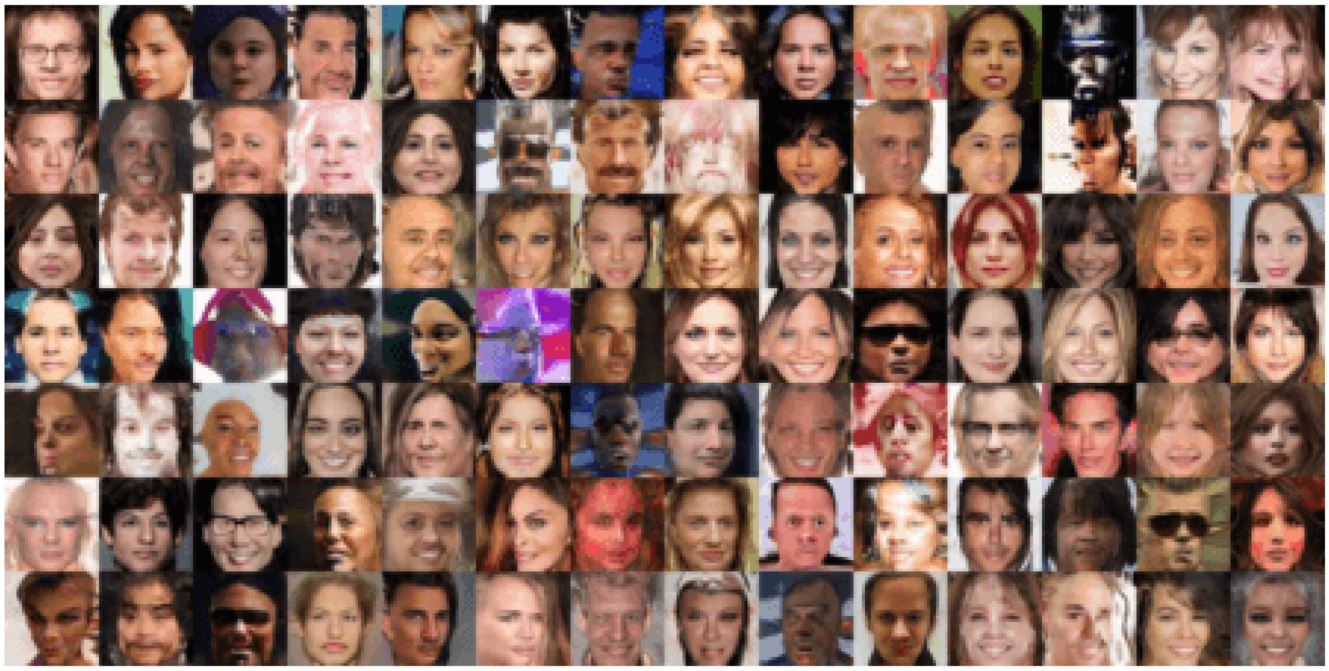
(c) Generated micro patches.

Figure 9: Full images generated by COCO-GAN on CelebA 128×128 with (N8,M8,S8) setting.

Due to file size limit, all images are compressed, please access the full resolution pdf from: https://goo.gl/5HLynv



(a) Selected generation samples.

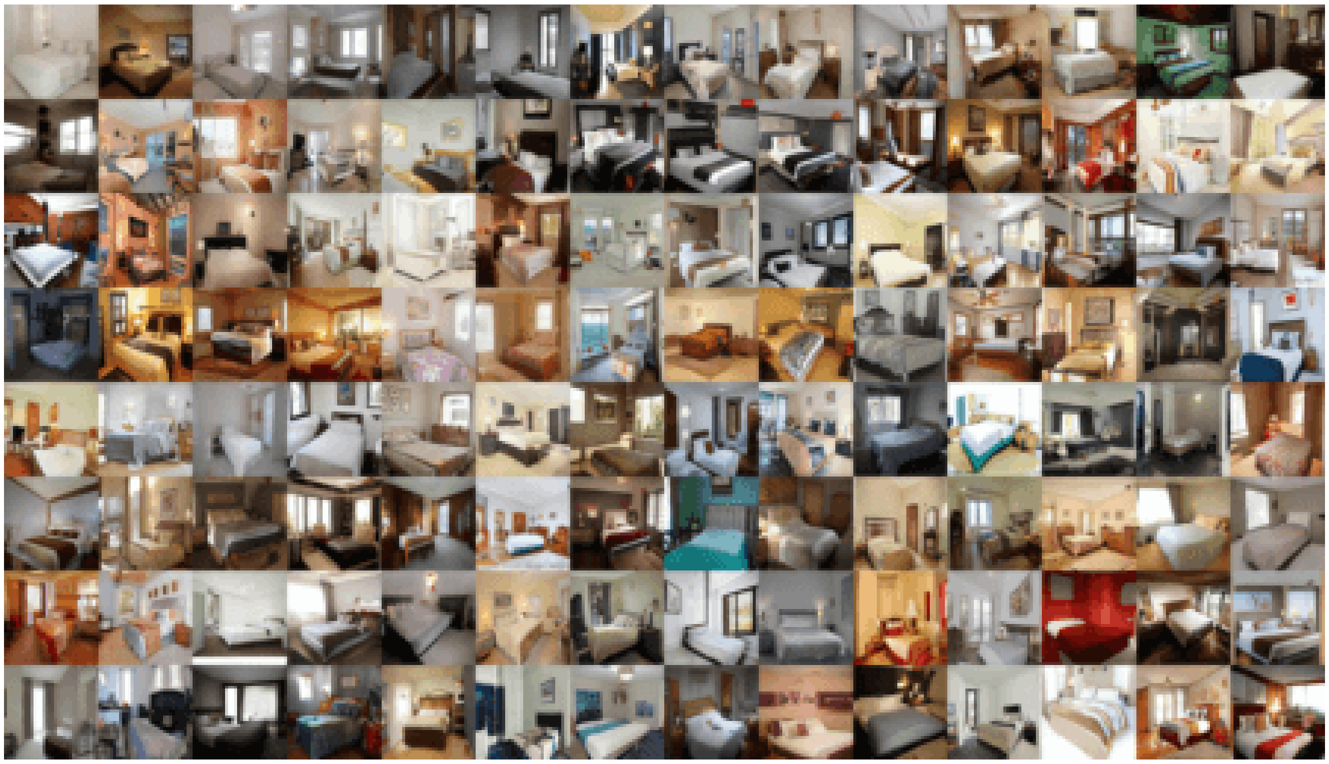


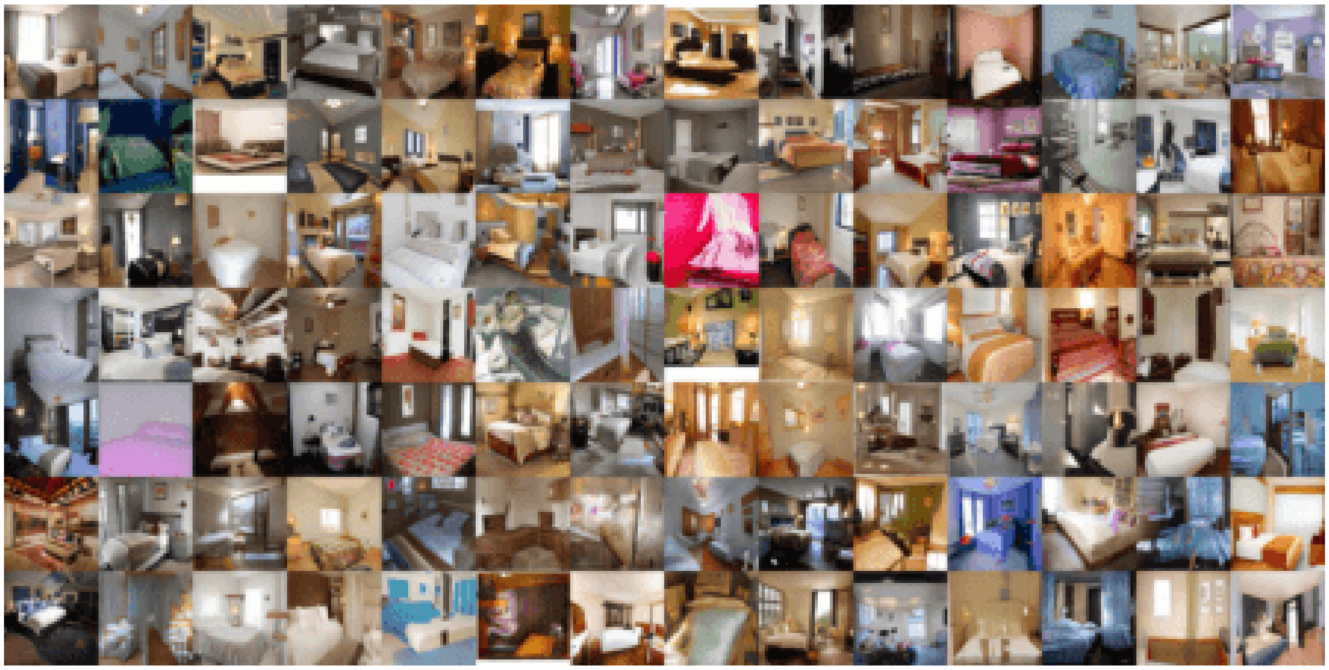
(b) Random generation without calibration.

(c) Generated micro patches.

Figure 10: Full images generated by COCO-GAN on CelebA 128×128 with (N16,M16,S4) setting.

Due to file size limit, all images are compressed, please access the full resolution pdf from: https://goo.gl/5HLynv


(a) Selected generation samples.

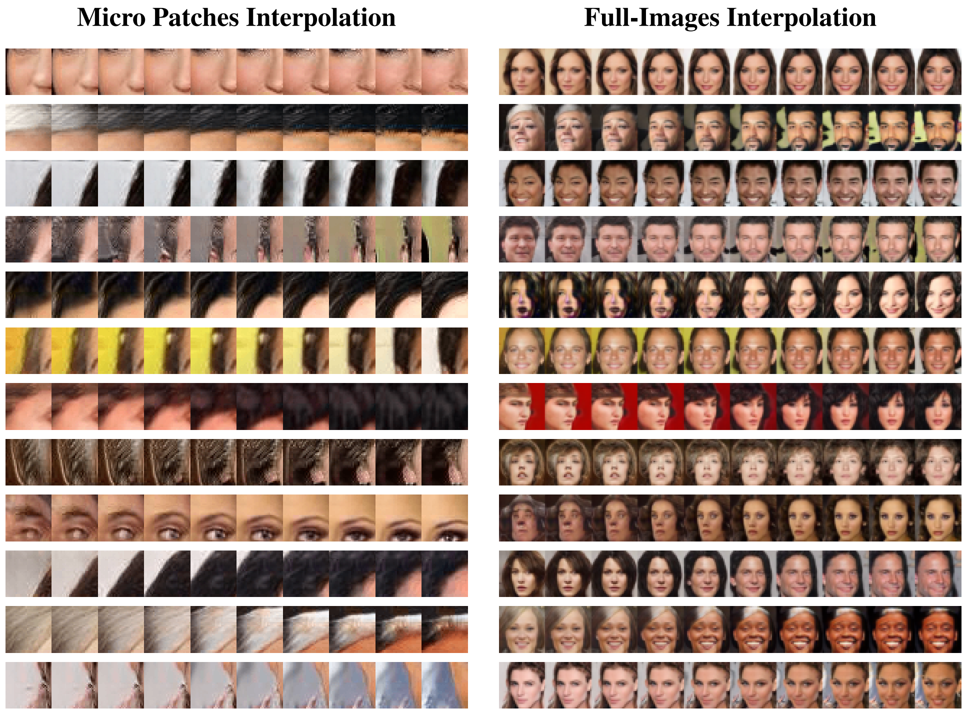

(b) Random generation without calibration.

(c) Generated micro patches.

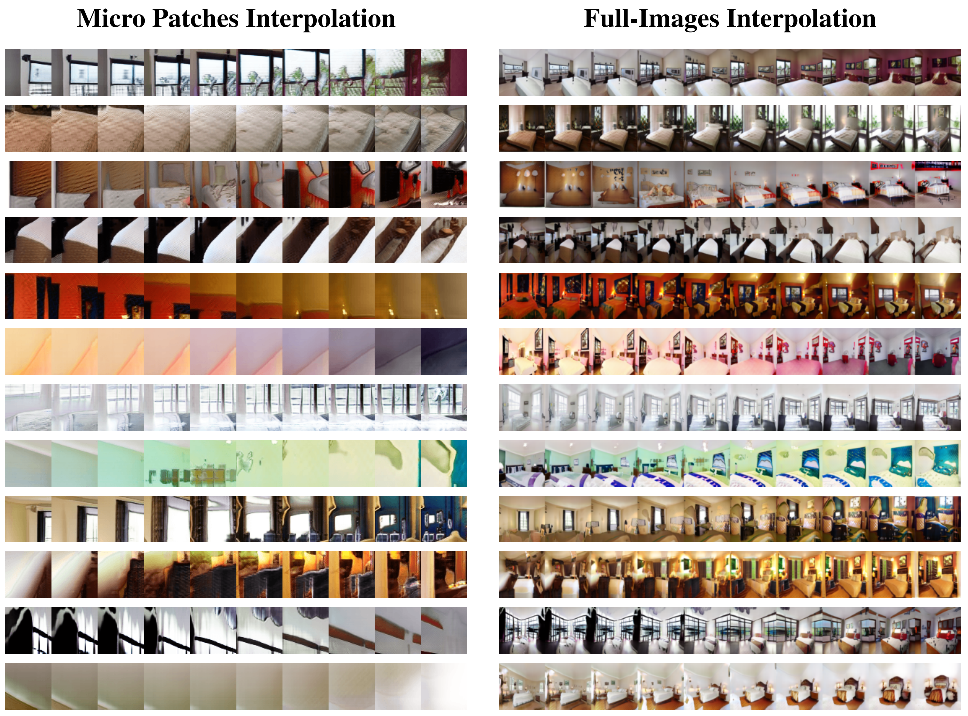
Figure 11: Full images generated by COCO-GAN on LSUN 256 $\times$ 256 with (N4,M4,S64) setting.

Due to file size limit, all images are compressed, please access the full resolution pdf from: https://goo.gl/5HLynv

G. More Interpolation Examples



(a) CelebA (128 × 128).



(b) LSUN (bedroom category) (256 × 256).

Figure 12: More interpolation examples. Given two latent vectors, COCO-GAN generates the micro patches and full images that correspond to the interpolated latent vectors.

H. More Panorama Generation Samples

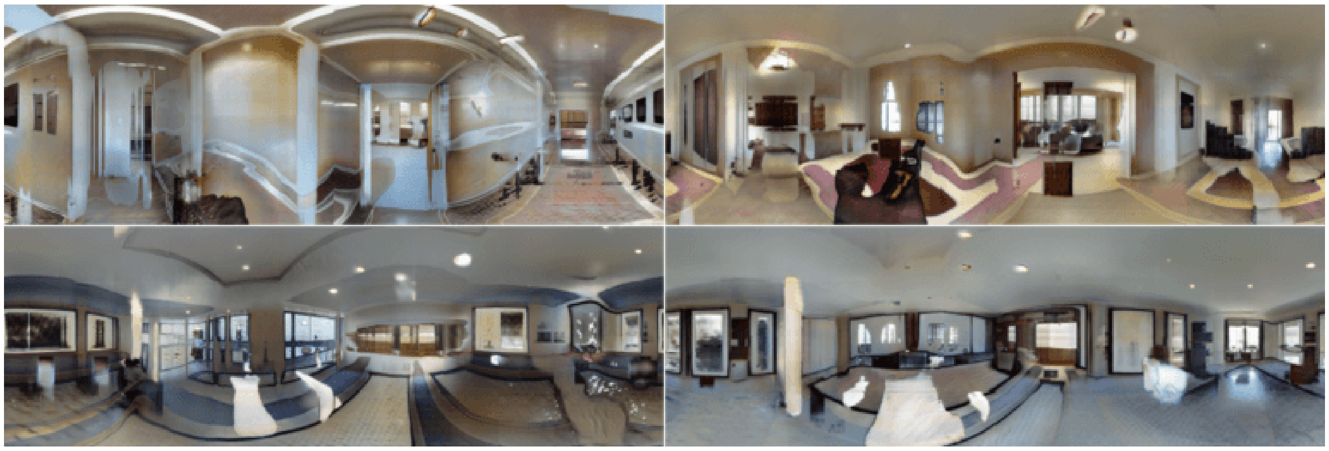


Figure 13: More examples of generated panoramas. All samples possess the cyclic property along the horizontal direction. Each sample is generated with a resolution of 768×256 pixels, and micro patch size 64×64 pixels.

I. Spatial Coordinates Interpolation

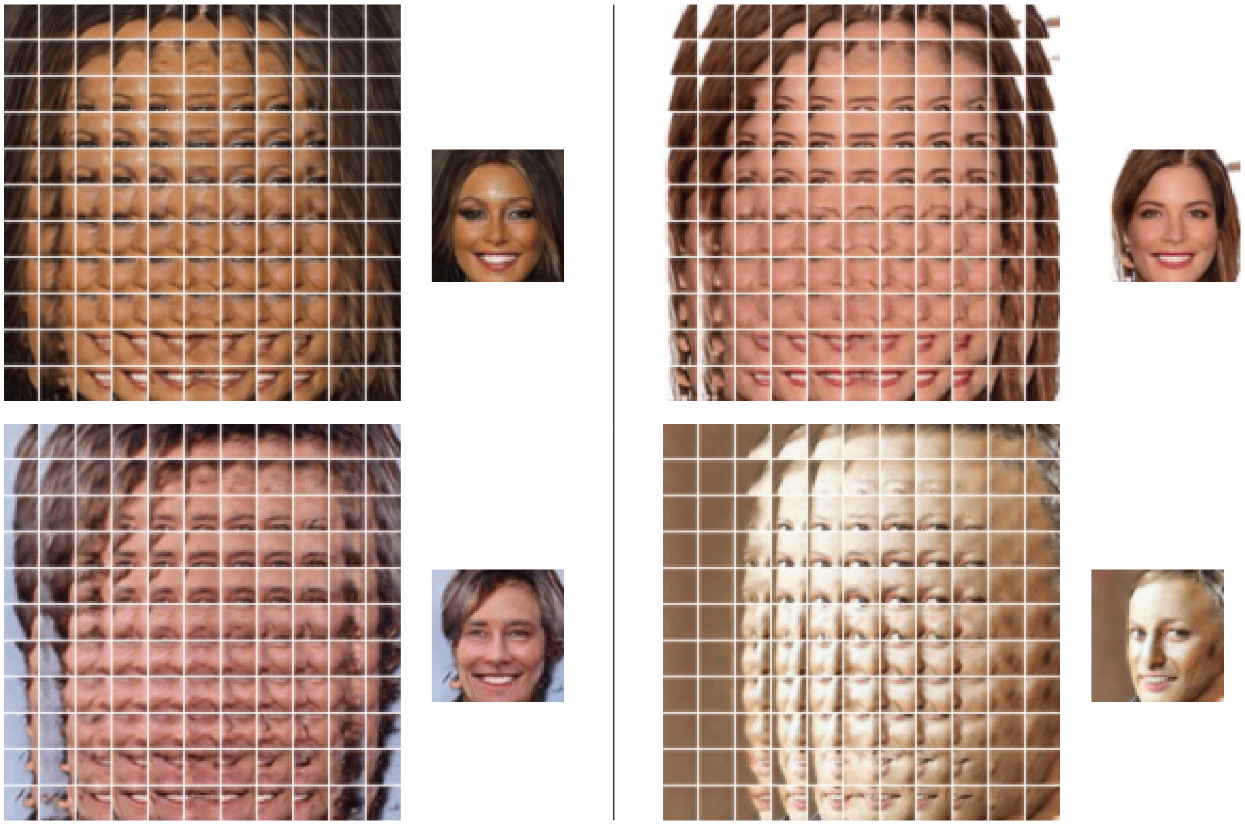


Figure 14: Spatial interpolation shows the spatial continuity of the micro patches. The spatial conditions are interpolated between range $[-1, 1]$ of the micro coordinate with a fixed latent vector.

J. Ablation Study

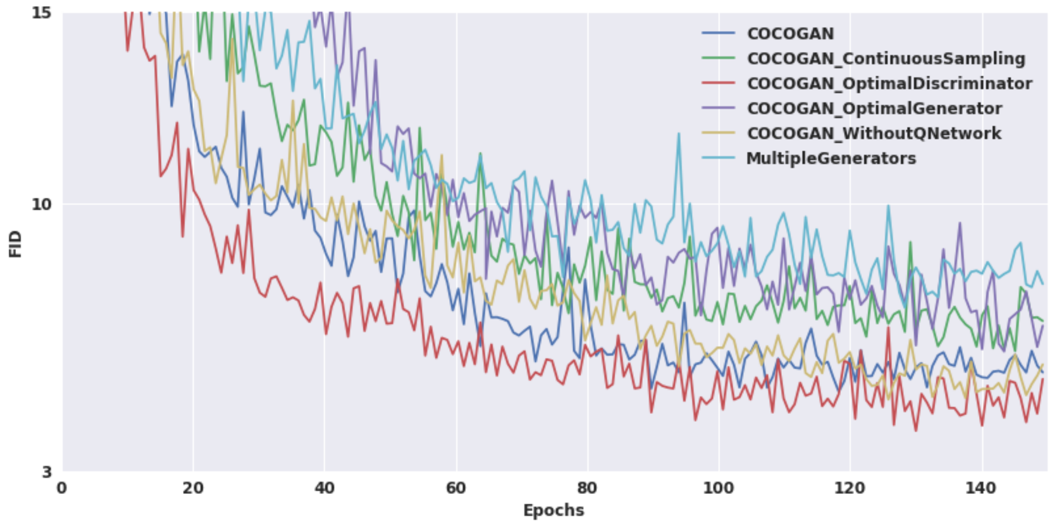
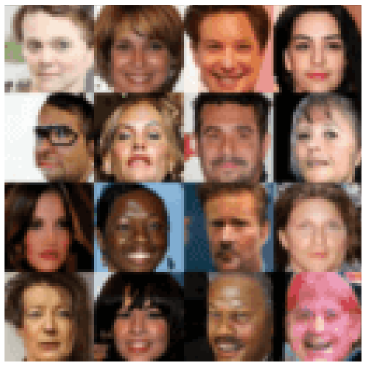
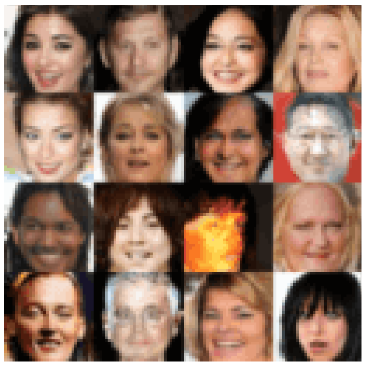
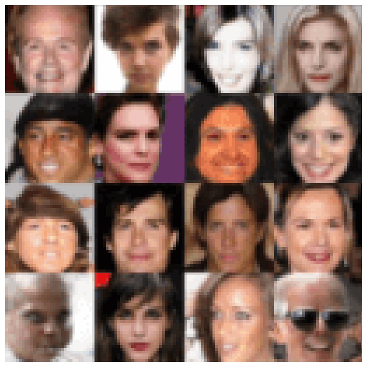


Figure 15: FID score curves of different variants of COCO-GAN in CelebA 64×64 setting. Combined with Figure 16, the results do not show significant differences in quality between COCO-GAN variants. Therefore, COCO-GAN does not pay significant trade-off for the conditional coordinate property.


(a) COCO-GAN (ours).


(b) COCO-GAN (cont sampling).

(c) COCO-GAN (optimal D).


(d) COCO-GAN (optimal G).

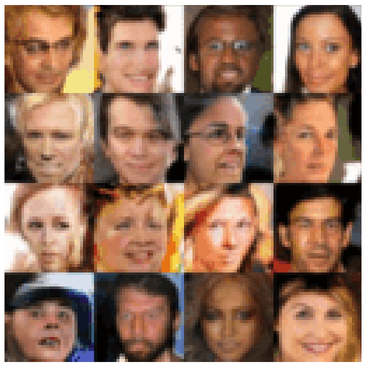
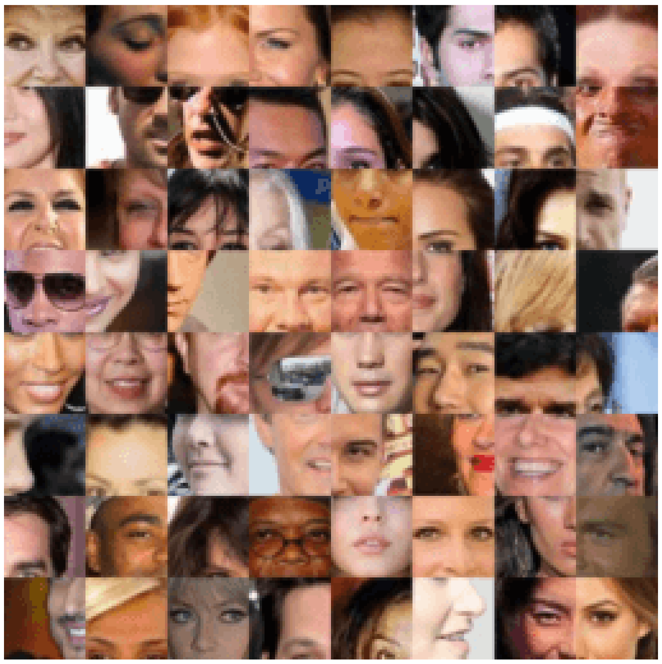
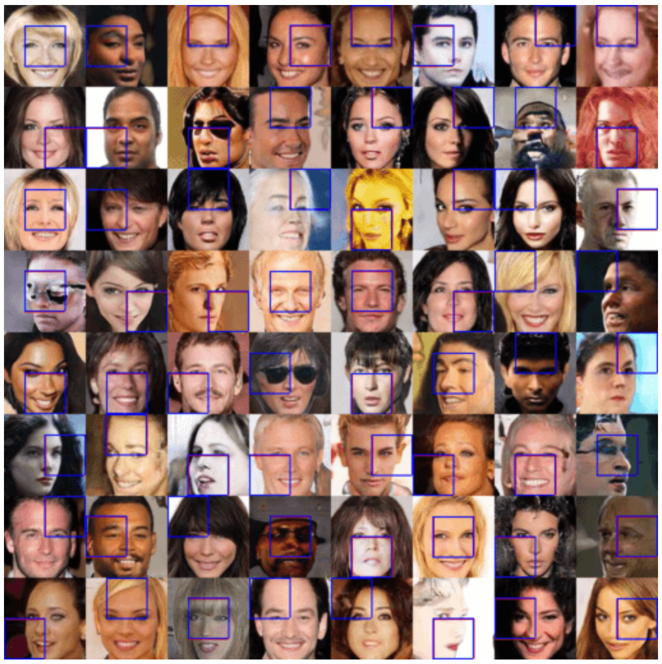

(e) Multiple generators.

Figure 16: Some samples generated by different variants of COCO-GAN. Note that each set of samples is extracted at the epoch when each model variant reaches its lowest FID score. We also provide more samples for each of the variants at different epochs via following : https://goo.gl/Wnrppf.

K. Patch-Guided Image Generation

(a) (CelebA 128×128) Real full images.


(b) (CelebA 128×128) Real macro patches.


(c) (CelebA 128×128) Patch-guided full image generation.

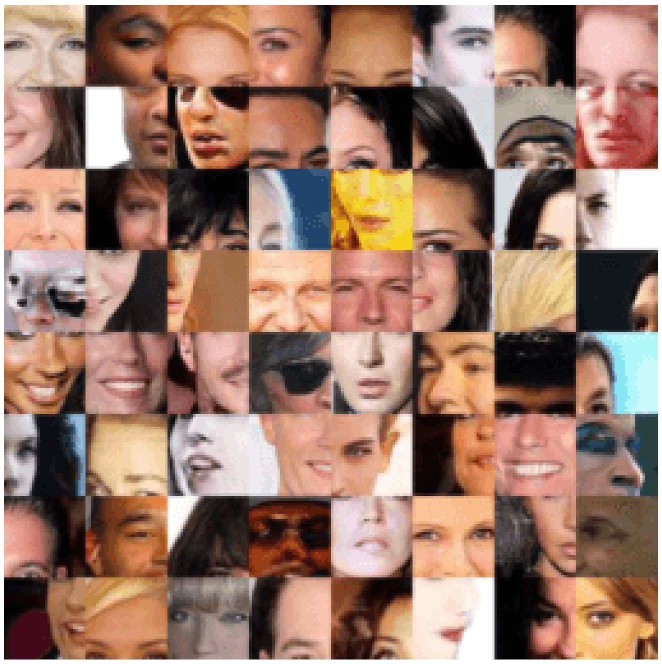
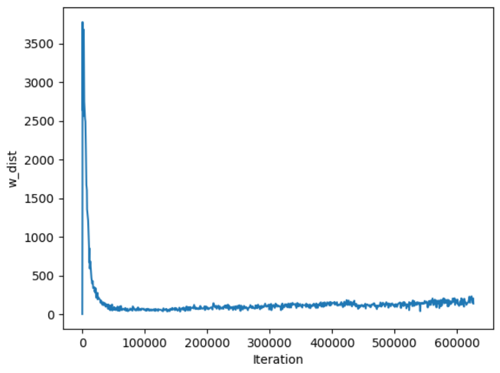
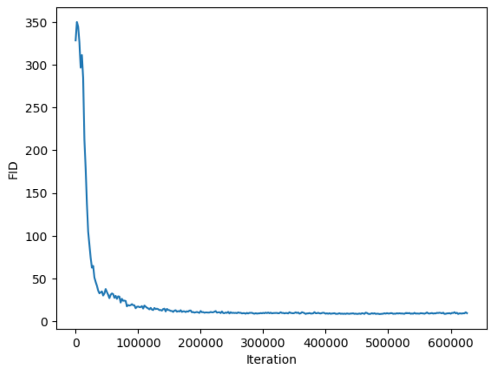

(d) (CelebA 128×128) Patch-guided macro patch generation.

Figure 17: Patch-guided image generation can loosely retain some local structure or global characteristic of the original image. (b) shows the patch-guided generated images based on z_{est} estimated from (a). The blue boxes visualize the predicted spatial coordinates $A(x')$, while the red boxes indicates the ground truth coordinates c'. Since the information loss of cropping the macro patches from real images is critical, we do not expect (b) to be identical to the original real image. Instead, the area within blue boxes of (b) should be visually similar to (a), in the meanwhile, (b) should be globally coherent.

L. Training Indicators



(a) Wasserstein distance



(b) FID

Figure 18: Both Wasserstein distance and FID through time show that the training of COCO-GAN is stable. Both two figures are logged while training on CelebA with 128×128 resolution.

References

[1] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 214–223, 2017. 2

[2] de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6597–6607, 2017. 2

[3] Dumoulin, V., Shlens, J., and Kudlur, M. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016. 2

[4] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6629–6640, 2017. 2

[5] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 2

[6] Miyato, T. and Koyama, M. cgans with projection discriminator. *CoRR*, abs/1802.05637, 2018. 2

[7] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. 2

[8] Sun, C., Hsiao, C.-W., Sun, M., and Chen, H.-T. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. *arXiv preprint arXiv:1901.03861*, 2019. 4

[9] Zhang, H., Goodfellow, I. J., Metaxas, D. N., and Odena, A. Self-attention generative adversarial networks. *CoRR*, abs/1805.08318, 2018. 2