(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

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

† 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 $\gamma$ and $\beta$ that used in conventional batch normalization, which computes $o_K = ((i_K - \mu_K)/\sigma_K) \times \gamma + \beta$ for the K-th input feature $i_K$, output feature $o_K$, feature mean $\mu_K$ and feature variance $\sigma_K$. 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, MLP$_\gamma(z, c)$ and MLP$_\beta(z, c)$, for each CBN layer, that conditionally produces $\gamma$ and $\beta$.

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 $C'_{(i,j)}$ 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.

![Figure 2: The detailed generator architecture of COCO-GAN for generating micro patches with a size of 32 x 32 pixels.](image)

(a) Generator Overall Architecture   (b) Generator Residual Block
Figure 3: The detailed discriminator architecture of COCO-GAN for discriminate macro patches with a size of \(64 \times 64\) 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

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).

With such an observation, we select to perform additional post-training on the check-point(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 $256 \times 256$ and a batch size 128). Aside from the original Adam optimizer that trains COCO-GAN with coordinates $c \in [-1, 1]$, 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 $c'' \in [-1.66, 1.66]$ (the distance between two consecutive micro patches is $2/(4-1) = 0.66$) 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 $384 \times 384$ pixels, whereas all of the training samples are of a smaller $256 \times 256$ resolution. The red box annotates the $256 \times 256$ 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 $128 \times 128$ 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
Due to file size limit, all images are compressed, please access the full resolution pdf from: https://goo.gl/5HLynv
Figure 9: Full images generated by COCO-GAN on CelebA $128 \times 128$ with (N8,M8,S8) setting.

(a) Selected generation samples.

(b) Random generation without calibration.

(c) Generated micro patches.

Due to file size limit, all images are compressed, please access the full resolution pdf from: https://goo.gl/5HLynyv
Figure 10: Full images generated by COCO-GAN on CelebA $128 \times 128$ 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
Figure 11: Full images generated by COCO-GAN on LSUN 256 × 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

![Micro Patches Interpolation](image1)

![Full-Images Interpolation](image2)

(a) CelebA (128 × 128).

![Micro Patches Interpolation](image3)

![Full-Images Interpolation](image4)

(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 $768 \times 256$ pixels, and micro patch size $64 \times 64$ 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 $64 \times 64$ 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.

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

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 $128 \times 128$ resolution.

References


