Attentive Normalization for Conditional Image Generation Supplementary Material

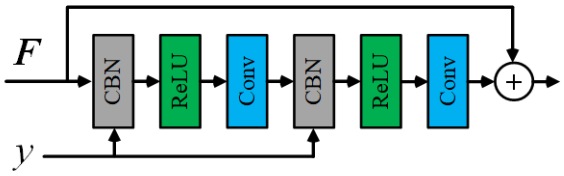


Figure 1. Default residual block used in the class-conditional image generation.

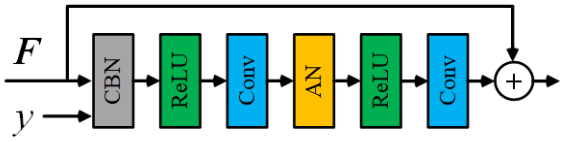


Figure 2. Residual block using attentive normalization.

1. Implementation of Attentive Normalization

Attentive normalization consists of the semantic layout learning module and subsequent regional normalization. The semantic layout learning module includes self-sampling regularization, similar to that of self-attention [5] except that we need an additional convolutional layer (with n filters) to simulate the semantic entities and avoid computing pair-wise relationship between every two feature points from the input. The implementation of regional normalization is given in Algorithm. 1.

2. Network Architectures

Class-conditional Image Generation Our used GAN framework is based on [3, 5] and we incorporate our designed attentive normalization (AN) onto it. Both the generator and discriminator mainly contain the residual blocks [2]. Construction of a standard residual block in this task is given in Figure 1, and the custom residual block with our AN is designed as Figure 2. The detailed structure of the used GAN model is given as follows.

Generator: $z \rightarrow$ FC ($4 \times 4 \times 1024$) $\rightarrow$ ResBlock up 1024 $\rightarrow$ ResBlock up 512 $\rightarrow$ ResBlock up 256 (AN) $\rightarrow$ ResBlock up 128 $\rightarrow$ ResBlock up 64 $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ Conv3 $\times$ 3 $\rightarrow$ Tanh

Discriminator: $x \rightarrow$ ResBlock down 64 $\rightarrow$ ResBlock down 128 (AN) $\rightarrow$ ResBlock down 256 $\rightarrow$ concat(Embed(y), h) $\rightarrow$ ResBlock down 512 $\rightarrow$ ResBlock down 1024 $\rightarrow$ ReLU $\rightarrow$ Global sum pooling $\rightarrow$ FC $\rightarrow$ 1

Generative Image Inpainting The two-stage inpainting framework is from [4]. It has two stacked encoder-decoder network structures. The second one is equipped with an extra branch, which has attentive normalization for context modeling. The generator design is illustrated in Figure 3. Suppose the input image is $\mathbf{X}$ and the prediction of the generator is $\hat{\mathbf{X}}$. The specification of the generator is given as

$\mathbf{X} \rightarrow$ **Encoder** $\rightarrow$ **Bottleneck** $\rightarrow$ **Decoder1** $\rightarrow \tilde{\mathbf{X}}$,
concat($\tilde{\mathbf{X}}$, $\mathbf{X}$) $\rightarrow$ **Encoder** $\rightarrow$ **Attentive branch** $\rightarrow \tilde{\mathbf{X}}_f^a$,
concat($\tilde{\mathbf{X}}$, $\mathbf{X}$) $\rightarrow$ **Encoder** $\rightarrow$ **Bottleneck** $\rightarrow \tilde{\mathbf{X}}_f^b$,
concat($\tilde{\mathbf{X}}_f^a$, $\tilde{\mathbf{X}}_f^b$) $\rightarrow$ **Decoder2** $\rightarrow \hat{\mathbf{X}}$,
where $\tilde{\mathbf{X}}$ is the coarse prediction of the first stage from the generator.

The design of the used components is given as
Encoder: Conv(5, 32, 1, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 64, 2, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 64, 1, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 128, 2, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 128, 1, 1) $\rightarrow$ ELU,
Bottleneck: Conv(3, 128, 1, 2) $\rightarrow$ ELU $\rightarrow$ Conv(3, 128, 1, 4) $\rightarrow$ ELU $\rightarrow$ Conv(3, 128, 1, 8) $\rightarrow$ ELU $\rightarrow$ Conv(3, 128, 1, 16) $\rightarrow$ ELU,
Decoder1: $\times 2 \rightarrow$ Conv(3, 64, 1, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 64, 1, 1) $\rightarrow$ ELU $\rightarrow \times 2 \rightarrow$ Conv(3, 32, 1, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 16, 1, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 3, 1, 1) $\rightarrow$ Clip to [-1, 1],
Decoder2: Conv(3, 128, 1, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 128, 1, 1) $\rightarrow$ ELU $\rightarrow$ **Decoder1**,
Attentive branch: Conv(3, 128, 1, 1) $\rightarrow$ AN $\rightarrow$ Conv(3, 128, 1, 1) $\rightarrow$ ELU $\rightarrow$ Conv(3, 128, 1, 1) $\rightarrow$ ELU,
where Conv(k, c, s, d) denotes a convolution layer, ELU denotes exponential linear unit [1] and $\times 2$ denotes a nearest-neighbor upsampling operator (with scaling factor 2). k, c, s, and d denote kernel size, filter number, stride size, and dilation rate, respectively.

The used discriminators are the same as those in [4] for global and local operations.

Algorithm 1 Pseudo code of regional normalization (Tensorflow style)

Input: Input feature maps $\mathbf{X} \in \mathcal{R}^{b\times h\times w\times c}$ and the learned soft semantic layout $\mathbf{S} \in \mathcal{R}^{b\times h\times w\times n}$, and learnable parameter vectors β and $\alpha \in \mathcal{R}^{1\times 1\times 1\times c\times 1}$.

Output: The normalized feature maps$\bar{\mathbf{X}} \in \mathcal{R}^{b\times h\times w\times c}$.

$\mathbf{S} = \text{tf.expand\_dims}(\mathbf{S}, -2)$ # shape: $b \times h \times w \times 1 \times n$

$\text{c} = \text{tf.reduce\_sum}(\mathbf{S}, \text{axis} = [1, 2], \text{keepdims} = \text{True}) + \epsilon$

$\mathbf{X} = \text{tf.expand\_dims}(\mathbf{X}, -1)$ # shape: $b \times h \times w \times c \times 1$

$\mathbf{X}_{\text{activated}} = \mathbf{X} * \mathbf{S}$ # shape: $b \times h \times w \times c \times n$

$\mathbf{X}_{\text{mean}} = \text{tf.reduce\_sum}(\mathbf{X}_{\text{activated}}, \text{axis} = [1, 2], \text{keepdims} = \text{True})/\text{c}$

$\mathbf{X}_{\text{std}} = \text{tf.sqrt}(\text{tf.reduce\_sum}((\mathbf{X}_{\text{activated}} - \mathbf{X}_{\text{mean}}) * *2, \text{axis} = [1, 2], \text{keepdims} = \text{True})/\text{c})$

$\mathbf{X} = (\mathbf{X} - \mathbf{X}_{\text{mean}})/(\mathbf{X}_{\text{std}} + \epsilon) * \beta + \alpha$

$\bar{\mathbf{X}} = \text{tf.reduce\_sum}(\mathbf{X} * \mathbf{S}, \text{axis} = -1)$

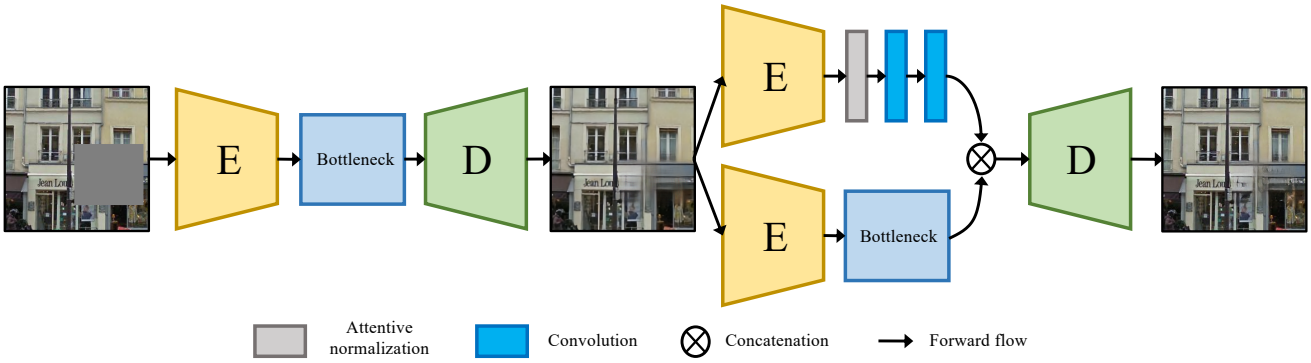


Figure 3. Employed generator equipped with the proposed attentive normalization for generative image inpainting.

3. More Experimental Results and Analysis

3.1. Ablation Studies

Attention Maps Activated By the Learned Semantic Entities The illustration is given in Figure 4. These entities correspond to different regions with their respective semantics. Using only 4-5 semantic entities in generation from Figure 4 is with the following reasons.

1) These figures are the binarized versions for visualization of the learned soft semantic layout (Eq. (4)). Only the maximal activation among all semantic entities is reserved as 1 and all others are clipped to 0.

2) Analogous to data clustering, there exist similar semantic entities in the feature space where its distribution density is high and n (like a clustering number) is relatively large. Increasing n from 8 to 16 may not produce semantic entities with more salient patterns. But it can better fit the feature distribution with larger model capacity, leading to performance improvement.

Effectiveness of Self-sampling Regularization (SSR) As analyzed in the paper, our proposed method fails to capture multiple semantics without SSR since most semantic entities tend to become useless in training. Besides quantitative evaluation in the ablation studies of the paper, more visualization results are provided in Figure 5 for reference. Note these binarized semantic layouts (Figure 5(b)) are in color. The feature points are activated by one semantic entity in all examples (Figure 5(h)).

3.2. Class-conditional Image Generation

More randomly generated images from our method on ImageNet are given in Figures 6, 7, and 8. And more categorical interpolation examples are shown in Figure 9.

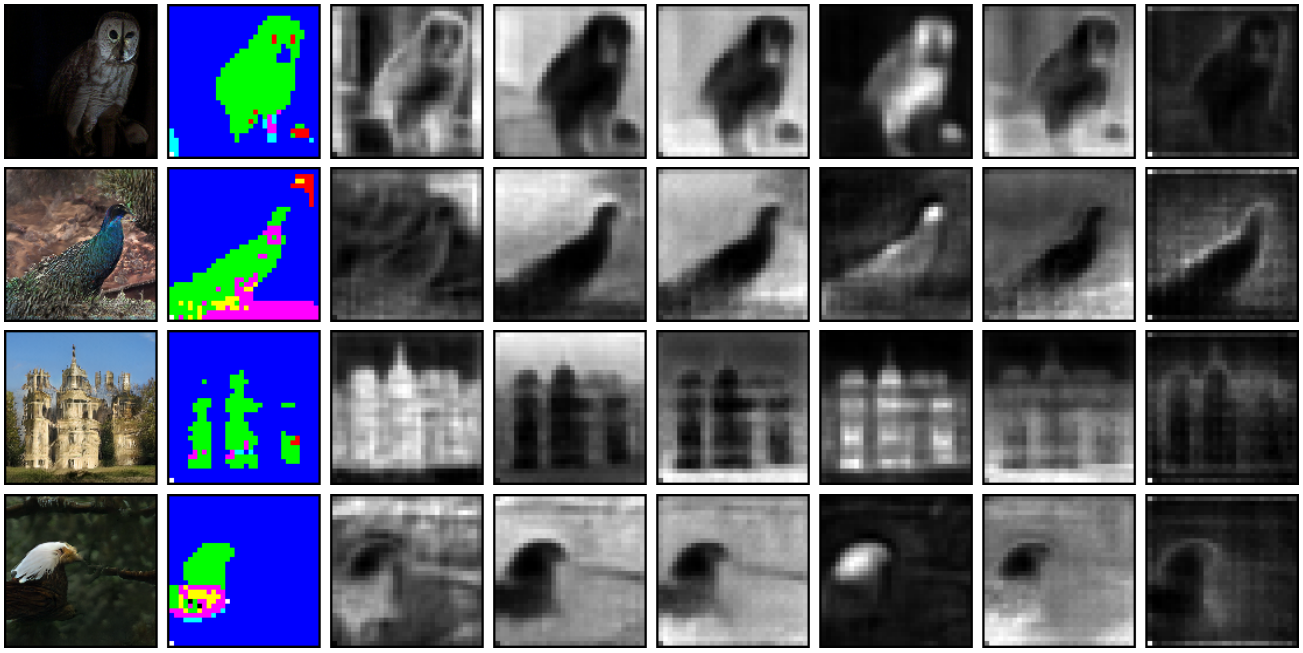
3.3. Generative Image Inpainting

More visual evaluation is presented in Figure 10.

3.4. Limitations

Since the relation between the feature points is computed by their similarities with the learned semantic entities instead of formed in a pair-wise fashion, it is possible that some features are not normalized when they are not similar to the given entities accordingly.

The only supervision for learning the semantic layout comes from the gradients of the discriminator for the class-conditional image generation, or the gradients of reconstructing images and the critics of the discriminator for the generative inpainting. Though these learned semantic entities are correlated with the high-level understanding, it is



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4. More visualization examples of the learned semantic layout on ImageNet. (a) The class-conditional generation results from our method. (b) The binarized version of the learned semantic layout. (c-h) The attention maps activated by the learned semantic entities. The brighter the activated regions are, the higher correlation they are with the used semantic entity. The resolution of the input feature maps is 32×32.

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5. Visualizations of the learned semantic layout without self-sampling regularization on ImageNet. (a) The class-conditional generation results from our method. (b) The binarized version of the learned semantic layout. (c-h) The attention maps activated by the learned semantic entities. The brighter the activated regions are, the higher correlation they are with the used semantic entity. The resolution of the input feature maps is 32×32.

Blenheim spaniel (156)

Persian cat (283)

Monarch butterfly (323)

Starfish (327)

Figure 6. Randomly generated images (128×128) by our model on ImageNet.

still hard to interpret their exact meaning.

References

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[3] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 1

[4] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018. 1, 8

[5] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 1

Aircraft carrier (403)

Drilling platform (540)

Schooner (780)

Alp (970)

Figure 7. Randomly generated images (128×128) by our model on ImageNet.

Promontory (976)

Agaric (992)

Espresso (967)

Daisy (985)

Figure 8. Randomly generated images (128×128) by our model on ImageNet.

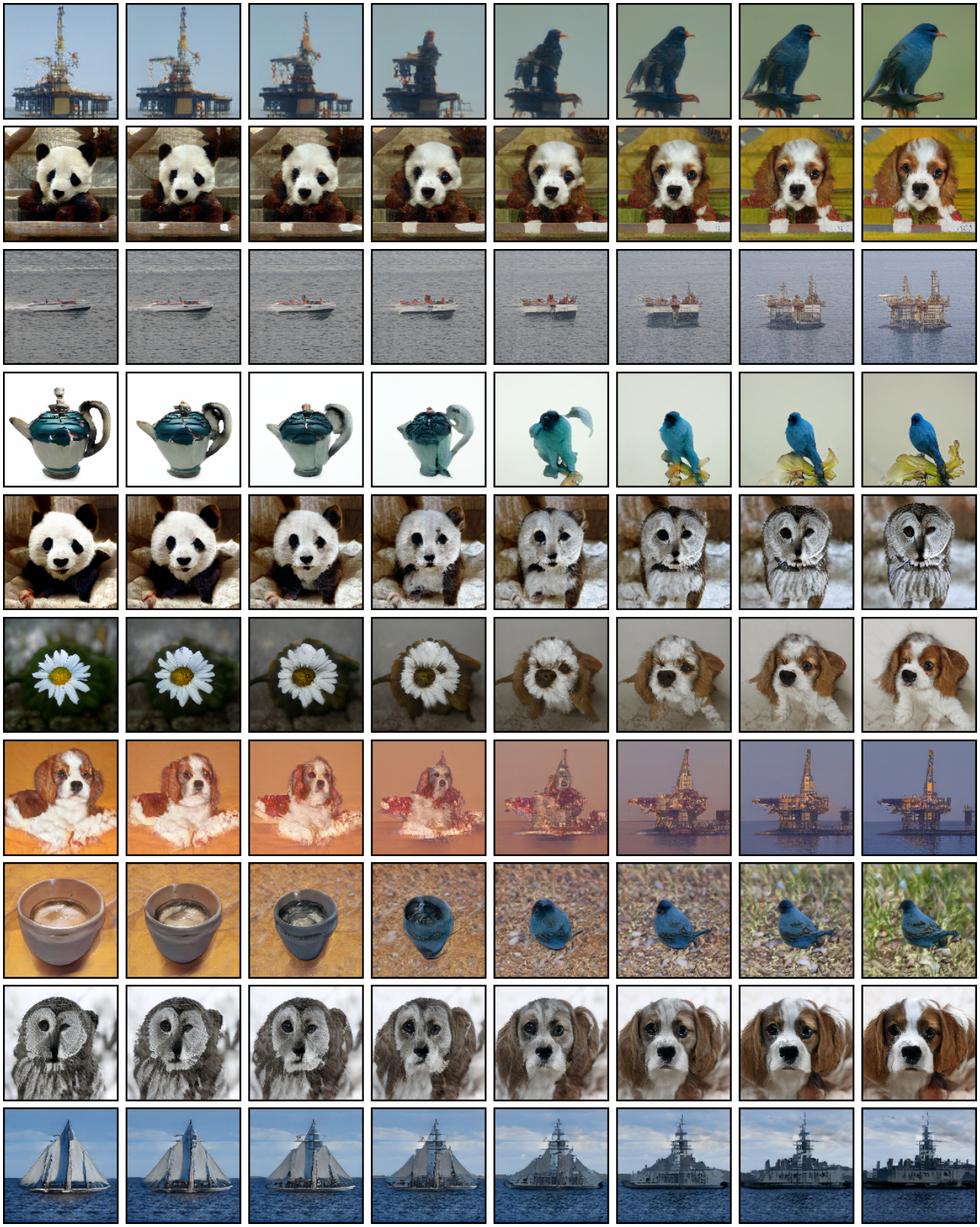


Figure 9. The categorical interpolation and intermediate results by our method. The results in each row are generated from a individual fixed noisy signal z.

(a) (b) (c) (a) (b) (c)

Figure 10. Visual comparisons on generative image inpainting on Paris street view. (a) Input image. (b) CA [4]. (c) Our results.