TISE: Bag of Metrics for Text-to-Image Synthesis Evaluation — Supplementary Material —

Tan M. Dinh*, Rang Nguyen, and Binh-Son Hua

VinAI Research, Hanoi, Vietnam
{tan.m.dinh.vn,rangnhm,binhson.hua}@gmail.com

Abstract. This supplemental document provides more details of our bag of metrics. We first present the details of our improved IS* metric including the impact of the calibration step on the IS* score (Section 1.1) and the implementation details for the counter model for reproducing the inconsistency problem of IS (Section 1.2). We then detailed our improved versions of RP and SOA and show how our metrics can mitigate the overfitting issues in the original versions in the multi-object text-to-image synthesis (Section 2). Next, we detail our benchmark including the statistics of our test data (Section 3.1), a complete benchmark of multi-object text-to-image models based on each assessment criterion (Section 3.2), the architecture and network configurations of our AttnGAN++ baseline (Section 4) as well as more visual examples (Section 5) and t-SNE visualization (Section 6). Finally, we provide the caption ids we used in our user study (Section 7).

1 Details for our IS* metric

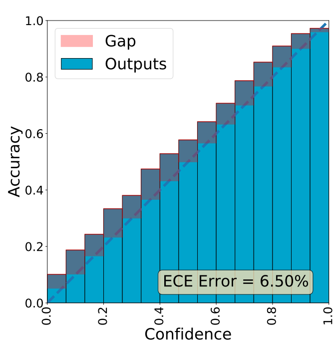
1.1 Impact of calibration on the classifier and IS*

In the main paper, we showed that severe miscalibration of the classifier used to compute IS on the CUB dataset in previous methods led to inconsistent IS scores of a counter model, and we proposed IS* to fix this issue. In this section, we further conduct another experiment to verify the impact of the calibration step causing on computing IS*. In detail, this experiment is performed on the vanilla GAN task, which is the Tiny ImageNet [11] image generation. The classifier used to measure IS in these works is the Inception-v3 network pre-trained on ImageNet [21]. It is worth noting that this classifier is used popularly for measuring IS in the traditional GAN image generation task. The IS and IS* results are shown in Table 1.1. We also plot the reliability diagrams and ECE errors of this classifier before and after calibration in Figure 1. As can be seen, even before calibration, this classifier is noticeably well calibrated. Hence, the effect of the calibration process on this classifier is negligible demonstrated through the temperature T after calibration is 0.909 ($T = 1$ means calibration does not have any effects on classifier). Therefore, we would only see the local ranking differs in IS and IS*.

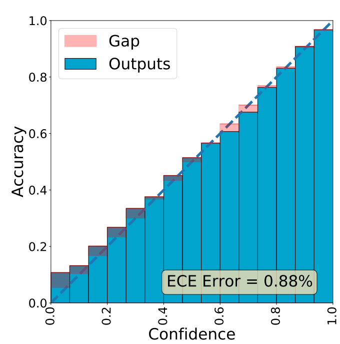
* Corresponding author

Table 1. Comparing the ranking of IS and IS* on Tiny ImageNet dataset with various GAN models. The cases, which are ranked inconsistently between IS and IS*, are marked in **bold**. As we expect, only local ranking differs between IS and IS* appear due to the well-calibrated of the classifier even before calibration.

Method	IS (↑)	IS* (↑)
WGAN-GP [4]	1.64	1.79
GGAN [13]	5.22	7.00
DCGAN [19]	5.70	7.79
ACGAN [17]	6.51	9.30
BigGAN-LO [24]	7.83	11.29
SNGAN [16]	**8.38**	**12.36**
SAGAN [27]	**8.48**	**12.34**
WGAN-DRA [10]	9.35	14.00
BigGAN [1]	12.43	18.80
ContraGAN [6]	13.76	21.64



(a) Before calibration



(b) After calibration

Fig. 1. Reliability diagrams of the Inception-v3 network pre-trained on the ImageNet dataset before and after calibration.

1.2 Counter model implementation

This section details the development of our counter model, which was utilized to demonstrate the inconsistency problem of IS. Our counter model is built on AttnGAN++ and MSG-GAN [7]. Table 8 shows the network details of the counter model. The training and evaluation configurations can be found in Table 9. More random (not curated) visual samples synthesized by the counter model are also provided in Figure 2 to demonstrate that these samples from the counter model are quite poor in comparison to those from AttnGAN++.

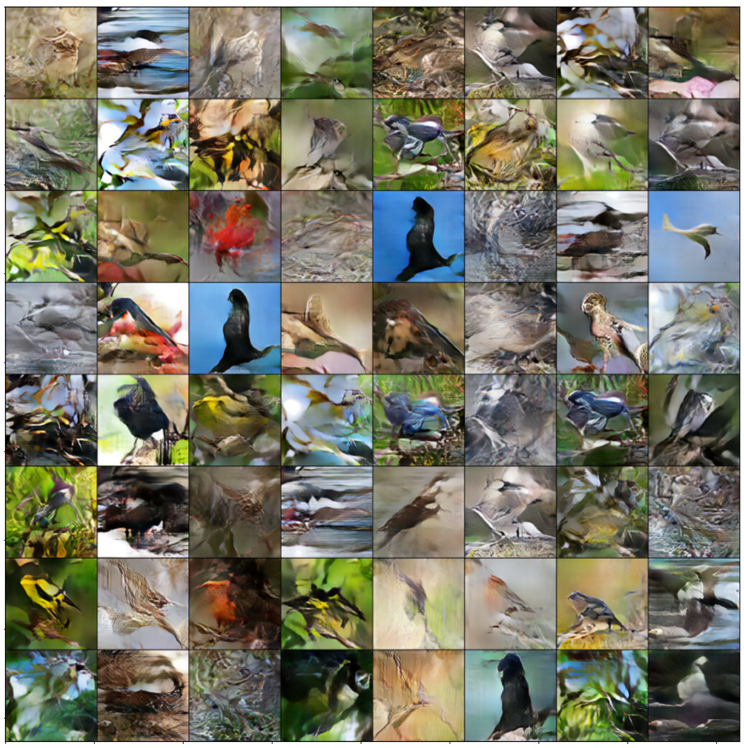


Fig. 2. More random (not curated) visual samples from our countermodel on the CUB dataset. As can be seen, the synthesized images are not realistic in most cases.

2 Details of our improved RP and SOA

In the main paper, we presented that the existing versions of RP and SOA overfit in the multi-object scenario of the MS-COCO dataset, as evidenced by the fact that the values from some methods on these metrics exceed the corresponding

Table 2. Comparison of the original version of RP and our improved version one on the MS-COCO [14] dataset. Inconsistent results are marked in **bold**. As can be observed, the value of RP on real photographs is the lowest, showing the original RP's heavily overfitting issue. Noticeably, our improved RP alleviated significantly by using CLIP [18]. Note that the values of RP in these experiments are calculated from $30,000$ captions as most of the previous works do. In the main paper, we only sample $5,000$ captions and calculate RP from these captions to save time but guarantee consistent scores.

Method	R-precision (original) (↑)	R-precision (ours) (↑)
StackGAN [28]	**72.03**	38.46
AttnGAN [25]	**83.76**	50.92
DM-GAN [29]	**92.23**	65.91
CPGAN [12]	**93.59**	70.36
AttnGAN++ (ours)	**96.39**	73.37
Real Images	67.35	83.65

values from real photos although these methods produce images with poorer quality than real photos. We demonstrate this by comparing the values of the original and our modified versions of these metrics in Table 2 and Table 3. As can be seen, the overfitting phenomena on RP and SOA is fully eliminated in our enhanced versions.

3 Details of our benchmark

3.1 Benchmark data

In the previous works, the inconsistency in the construction of testing data has caused many difficulties in benchmark models. A comprehensive survey [3] also pointed that there are some metrics are reported with inconsistent numbers between different research works. We find out that the non-unified input test data is one of the reasons leading to this issue. Therefore, we provide unified testing data in our TISE toolbox in order to compare techniques fairly.

The details of our test data is as follows. The number of captions used in each metrics are shown in Table 4 and Table 5 for CUB and MS-COCO, respectively. The distribution of per-class object count and positional words for counting alignment (CA) and positional alignment (PA) metric are visualized in Figure 3 and Figure 4, respectively.

Table 3. Comparison of the original version of SOA (including SOA-I and SOA-C) and our improved version of SOA on the MS-COCO [14] dataset. Inconsistent results are highlighted in **bold**, which shows that SOA-C and SOA-I of CPGAN are higher than real images and our SOA greatly migrated this phenomenon. Note that the values of SOA in this experiment are calculated on full captions provided by the authors of this metric, while the ones we report by our TISE toolbox are computed on the sample from them (about 16k captions) to save time but output the consistent scores.

Method	SOA-C (↑) (original)	SOA-I (↑) (original)	SOA-C (↑) (ours)	SOA-I (↑) (ours)
StackGAN [28]	21.09	30.35	31.34	49.97
AttnGAN [25]	25.88	39.01	47.26	62.02
DM-GAN [29]	33.44	48.03	55.40	68.76
CPGAN [12]	**77.02**	**84.55**	82.25	88.97
AttnGAN++(ours)	48.33	67.19	67.52	76.33
Real Images	74.97	80.84	89.98	92.92

Table 4. The number of test captions used in evaluation on the CUB dataset.

Metric	#Captions
Image Realism (IS, FID)	30,000
Text Relevance (RP)	30,000

Table 5. The number of test captions used in evaluating each evaluation aspect on the MSCOCO dataset.

Metric	#Captions
Image Realism (IS, FID)	10,000
Object Fidelity (O-IS, O-FID)	10,000
Text Relevance (RP)	5,000
Object Accuracy (SOA-C, SOA-I)	15,223
Positional Alignment (PA)	1,046
Counting Alignment (CA)	1,000

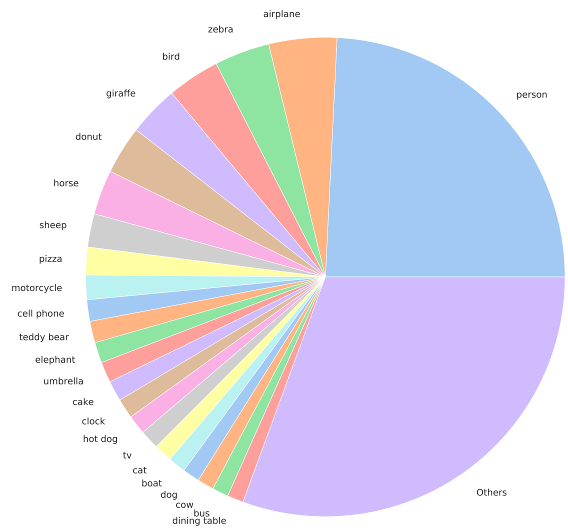


Fig. 3. Distribution of the number of object classes in our provided testing data for counting alignment factor in multi-object case. Best viewed in zoom.

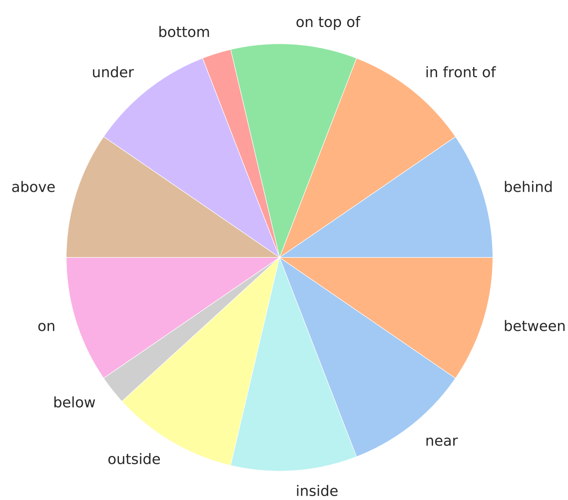


Fig. 4. Distribution of the number of test captions having corresponding positional words considered in our PA metric.

Table 6. The details of the ranking scores for each evaluation aspect of each method on the MS-COCO [14] dataset. **Best** and <u>runner-up</u> values are marked in bold and underline, respectively.

Method	Image Realism	Text Relevance	Object Accuracy	Object Fidelity	Counting Alignment	Positional Alignment
GAN-CLS [20]	1.0	2.0	1.0	1.0	1.0	1.0
StackGAN [28]	2.5	1.0	2.0	2.0	2.0	2.0
AttnGAN [25]	5.0	5.0	5.5	4.5	6.0	3.0
DM-GAN [29]	6.5	7.0	7.0	7.5	8.0	5.0
CPGAN [12]	7.5	8.0	**10.0**	7.5	4.0	6.0
DF-GAN [22]	7.0	3.0	4.0	<u>8.5</u>	5.0	4.0
AttnGAN + CL [26]	6.5	6.0	5.5	5.0	7.0	7.0
DM-GAN + CL [26]	<u>8.5</u>	<u>9.0</u>	8.0	7.0	<u>9.0</u>	**10.0**
DALLE-mini (zero-shot) [2]	2.5	4.0	3.0	3.0	3.0	8.0
AttnGAN++ (Ours)	**9.0**	**10.0**	<u>9.0</u>	**9.0**	**10.0**	<u>9.0</u>
Real Images	*10.0*	*11.0*	*11.0*	*11.0*	*11.0*	*11.0*

3.2 Benchmark on MS-COCO for each evaluation aspect

Table 6 provides the details of aspect's ranking scores for each method on MS-COCO dataset. Here we show the performance of each model on six evaluation criteria, including Image Realism, Object Accuracy, Text Relevance, Object Accuracy, Object Fidelity, Counting Alignment, Positional Alignment.

4 AttnGAN++ Architecture

Along with the assessment toolkit, we also offered our AttnGAN++, a new baseline based on AttnGAN [25]. The main difference between AttnGAN++ and AttnGAN is that we apply spectral normalization [16] to discriminators to stabilize the training process of GAN. With this simple technique, the performance of the model is boosted significantly comparing with the original version. The architecture of AttnGAN++ is shown in Figure 5. The network details and training settings of AttnGAN++ are demonstrated in Table 7 and Table 9 respectively.

5 More visual results

Additionally, we show more visual examples of our AttnGAN++ comparing with the current state-of-the-art text-to-image models on CUB [23] dataset in Figure 6 and MS-COCO [14] dataset in Figure 7 for qualitative measuring.

6 t-SNE Visualizations

To visualize the statistics of synthesized images, we utilize t-SNE [15]. Firstly, we extract feature vectors from these synthesized images using a pre-trained image encoder [25]. Then, we use t-SNE to convert these high dimensional feature vectors to 2-dimensional positions at which we display the images. The t-SNE visualization of generated images by AttnGAN++ and counter model on CUB can be found in Figure 8 and Figure 9, respectively. Additionally, we also show the t-SNE of all real images of the CUB test set in Figure 10 for reference. Note that the t-SNE image has a very high resolution so it is best viewed with an offline pdf viewer.

7 User Study

To facilitate reproducibility, we provide the IDs of captions which we used in our human evaluation: *503647, 302716, 817708, 72017, 563987, 434439, 375212, 478341, 737362, 323692, 177535, 338067, 810717, 416305, 680452, 439866, 558122, 545601, 196294, 380857, 782291, 324845, 767124, 63597, 648878, 73383, 327849, 799148, 829090, 107333, 805428, 371195, 443142, 394904, 754057, 421896, 361352, 517666, 75305, 625131, 202787, 723526, 569736, 442834, 183253, 642468, 277787, 150568, 502193, 643215.*

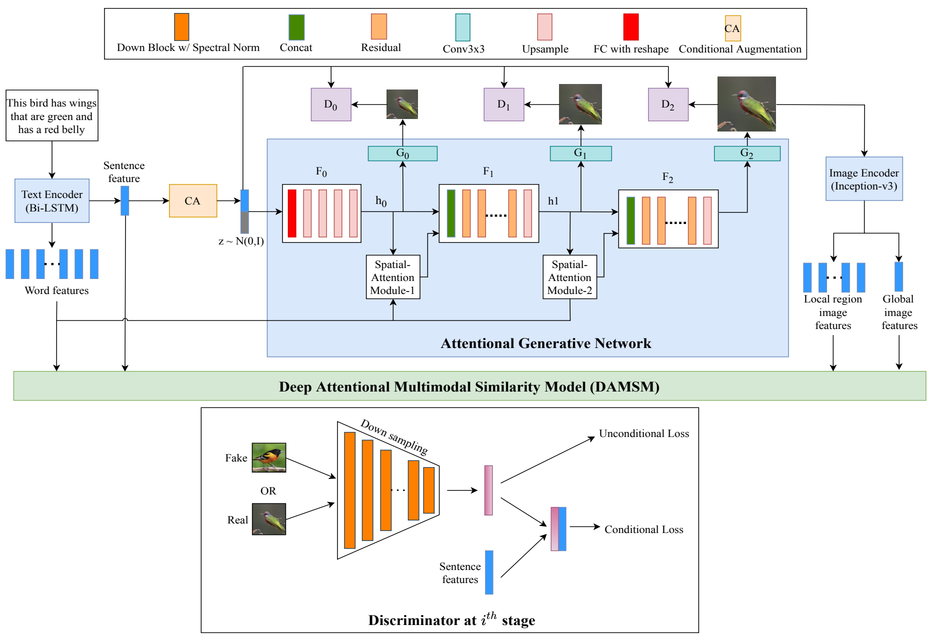


Fig. 5. The architecture of our AttnGAN++. In each discriminator, we employ spectral normalization [16] to each convolution layer instead of using batch normalization [5] as in the original AttnGAN [25]. Implementation details for each layer can be found in Table 7.

Fig. 6. Qualitative examples of the single object text-to-image generation models on the CUB [23] dataset. Best viewed in zoom.

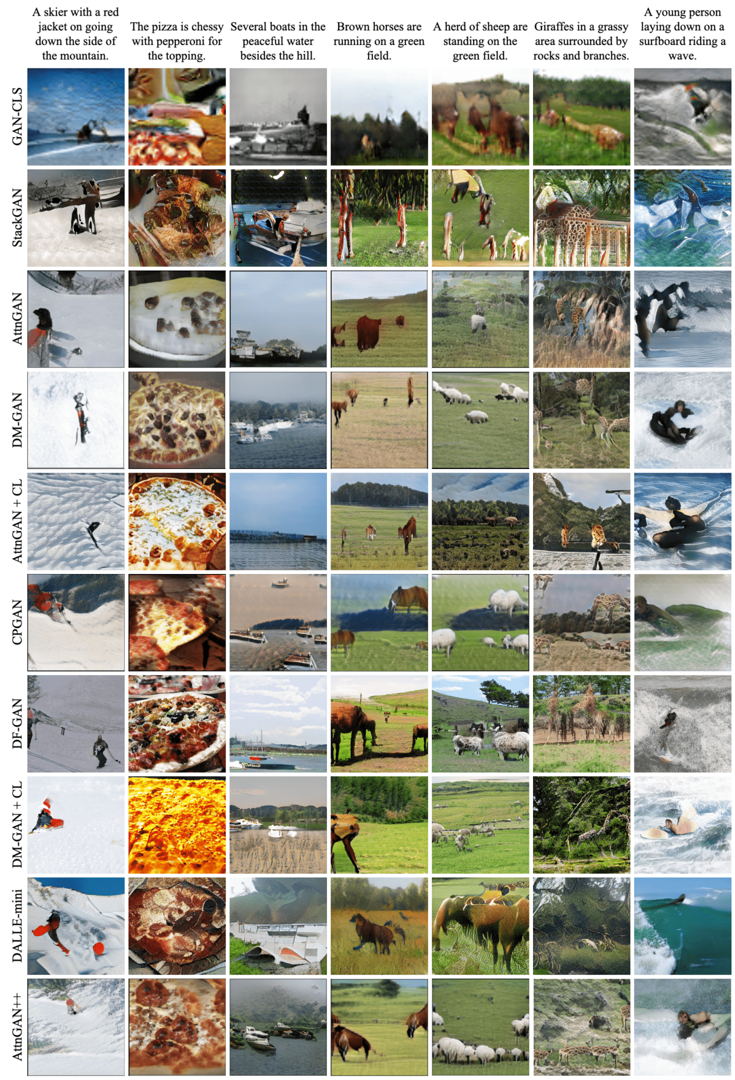


Fig. 7. Qualitative examples of the multi-object text-to-image synthesis models on the MS-COCO [14] dataset. Best viewed in zoom.

Table 7. Network details of our AttnGAN++. Some components which are not mentioned here such as text encoder, image encoder, DAMSM, its settings can be found in AttnGAN [25]. In the tables, k = kernel size, s = stride, p = padding, b = bias.

(a) Generator

Module	Output shape / Details
Up Block	
Params: (in_planes, out_planes)	
Input shape	in_planes × h × w
Upsampling	Nearest Neighbor, scale factor = 2
Conv(k=3, s=1, p=1, b=False)	2 ∗ out_planes × $h * 2$ × $w * 2$
BatchNorm2D	No change shape
Gated Linear Unit (GLU)	out_planes × $h * 2$ × $w * 2$
Residual Block	
Input X	
Conv(k=3, s=1, p=1, b=False)	Up channel size by 2,
BatchNorm2D	No change shape
Gated Linear Unit (GLU)	Down channel size by 2
Conv(k=3, s=1, p=1, b=False)	No change shape
BatchNorm2D	No change shape
Add output w/ X (skip connection)	No change shape
Spatial Attention layer	see AttnGAN
Conditional Augumentation (CA)	see AttnGAN
Generator 64×64	
Input	
Input noise	nzf
Caption Embedding	nef
Computation	
CA on caption embedding to get c	ncf
Concat c w/ noise	$ncf + nzf$
Linear(b=False)	$ngf * 16 * 4 * 4 * 2$
BatchNorm1D	No change shape
Gated Linear Unit (GLU)	$ngf * 16 * 4 * 4 * 1$
Reshape	$16 * ngf \times 4 \times 4$
Up Block 1	$8 * ngf \times 8 \times 8$
Up Block 2	$4 * ngf \times 16 \times 16$
Up Block 3	$2 * ngf \times 32 \times 32$
Up Block 4	$ngf \times 64 \times 64$
Conv(k=3, s=1, p=1, b=False)	$3 \times 64 \times 64$
Tanh	No change shape
Generator 128×128	
Input	
Previous hidden features	$ngf \times 64 \times 64$
Word Mask	$word\_num$
Word features	$nef \times word\_num$
Computation	
Spatial Attention Layer	$ngf \times 64 \times 64$
Residual Block × residual_num	$ngf \times 64 \times 64$
Concat w/ previous hidden features	$2 * ngf \times 64 \times 64$
Up Block	$ngf \times 128 \times 128$
Conv(k=3, s=1, p=1, b=False)	$3 \times 128 \times 128$
Tanh	No change shape
Generator 256×256	
Input	
Previous hidden features	$ngf \times 128 \times 128$
Word Mask	$word\_num$
Word features	$nef \times word\_num$
Computation	
Spatial Attention Layer	$ngf \times 128 \times 128$
Residual Block × residual_num	$ngf \times 128 \times 128$
Concat w/ previous hidden features	$2 * ngf \times 128 \times 128$
Up Block	$ngf \times 256 \times 256$
Conv(k=3, s=1, p=1, b=False)	$3 \times 256 \times 256$
Tanh	No change shape

(b) Discriminator

Module	Output shape / Details
Down Block	
Params: (in_planes, out_planes)	
Input shape	in_planes × h × w
SpectralNorm(Conv(k=4, s=2, p=1, b=True)	out_planes × h/2 × w/2
LeakyReLU(alpha=0.2)	No change shape
Block3x3_leakyReLU	
Params: (in_planes, out_planes)	
Input shape	in_planes × h × w
SpectralNorm(Conv(k=3, s=1, p=1, b=True)	out_planes × h × w
LeakyReLU(alpha=0.2)	No change shape
Discriminator 256×256	
Input tensor	$3 \times 256 \times 256$
Down Block	$ndf \times 128 \times 128$
Down Block	$2 * ndf \times 64 \times 64$
Down Block	$4 * ndf \times 32 \times 32$
Down Block	$8 * ndf \times 16 \times 16$
Down Block	$16 * ndf \times 8 \times 8$
Down Block	$32 * ndf \times 4 \times 4$
Block3x3_leakyReLU	$16 * ndf \times 4 \times 4$
Block3x3_leakyReLU	$8 * ndf \times 4 \times 4$
Unconditional logits	
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1
Conditional logits	
Caption Embedding	nef
Concat w/ replicated caption embedding	$8 * ndf + nef \times 4 \times 4$
Block3x3_leakyReLU	$8 * ndf \times 4 \times 4$
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1
Discriminator 128×128	
Input tensor	$3 \times 128 \times 128$
Down Block	$ndf \times 64 \times 64$
Down Block	$2 * ndf \times 32 \times 32$
Down Block	$4 * ndf \times 16 \times 16$
Down Block	$8 * ndf \times 8 \times 8$
Down Block	$16 * ndf \times 4 \times 4$
Block3x3_leakyReLU	$8 * ndf \times 4 \times 4$
Unconditional logits	
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1
Conditional logits	
Caption Embedding	nef
Concat w/ replicated caption embedding	$8 * ndf + nef \times 4 \times 4$
Block3x3_leakyReLU	$8 * ndf \times 4 \times 4$
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1
Discriminator 64×64	
Input tensor	$3 \times 64 \times 64$
Down Block	$ndf \times 32 \times 32$
Down Block	$2 * ndf \times 16 \times 16$
Down Block	$4 * ndf \times 8 \times 8$
Down Block	$8 * ndf \times 4 \times 4$
Unconditional logits	
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1
Conditional logits	
Caption Embedding	nef
Concat w/ replicated caption embedding	$8 * ndf + nef \times 4 \times 4$
Block3x3_leakyReLU	$8 * ndf \times 4 \times 4$
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1

Table 8. Network details of our countermodel. Some components which are not mentioned here such as text encoder, image encoder, DAMSM, its settings can be found in AttnGAN [25]. In the tables, k = kernel size, s = stride, p = padding, b = bias.

(a) Generator

Module	Output shape / Details
Up Block	see Table 7
Residual Block	see Table 7
Spatial Attention Layer	see AttnGAN
Conditional Augumentation	see AttnGAN
Generator 4×4	
Input	
Input noise	nzf
Caption Embedding	nef
Computation	
Conditional Augumentation on caption embedding	ncf
Concat w/ noise	$ncf + nef$
Linear(b=False)	$ngf * 16 * 4 * 4 * 2$
BatchNorm1D	No change shape
Gated Linear Unit (GLU)	$ngf * 16 * 4 * 4 * 1$
Reshape	$16 * ngf \times 4 \times 4$
Conv(k=3, s=1, p=1, b=False)	$3 \times 4 \times 4$
Tanh	No change shape
Generator 8×8	
Up Block	$8 * ngf \times 8 \times 8$
Conv(k=3, s=1, p=1, b=False)	$3 \times 8 \times 8$
Tanh	No change shape
Generator 16×16	
Up Block	$4 * ngf \times 16 \times 16$
Conv(k=3, s=1, p=1, b=False)	$3 \times 16 \times 16$
Tanh	No change shape
Generator 32×32	
Up Block	$2 * ngf \times 32 \times 32$
Conv(k=3, s=1, p=1, b=False)	$3 \times 32 \times 32$
Tanh	No change shape
Generator 64×64	
Up Block	$ngf \times 64 \times 64$
Conv(k=3, s=1, p=1, b=False)	$3 \times 64 \times 64$
Tanh	No change shape
Generator 128×128	
Input	
Previous hidden features	$ngf \times 64 \times 64$
Word Mask	$word\_num$
Word features	$nef \times word\_num$
Computation	
Spatial Attention Layer	$ngf \times 64 \times 64$
Residual Block $\times$ residual_num	$ngf \times 64 \times 64$
Concat w/ previous hidden features	$2 * ngf \times 64 \times 64$
Up Block $ngf \times 128 \times 128$	
Conv(k=3, s=1, p=1, b=False)	$3 \times 128 \times 128$
Tanh	No change shape
Generator 256×256	
Input	
Previous hidden features	$ngf \times 128 \times 128$
Word Mask	$word\_num$
Word features	$nef \times word\_num$
Computation	
Spatial Attention Layer	$ngf \times 128 \times 128$
Residual Block $\times$ residual_num	$ngf \times 128 \times 128$
Concat w/ previous hidden features	$2 * ngf \times 128 \times 128$
Up Block	$ngf \times 256 \times 256$
Conv(k=3, s=1, p=1, b=False)	$3 \times 256 \times 256$
Tanh	No change shape

(b) Discriminator

Module	Output shape / Details
Block3x3_leakyReLU	see Table 7
DisGeneralConvBlock	
Params: in_planes, concat_planes, out_planes	
MinibatchStdDev (see [8,9])	in_planes + concat_planes $\times$ h $\times$ w
Block3x3_leakyRelu	in_planes $\times$ h $\times$ w
Block3x3_leakyRelu	out_planes $\times$ h $\times$ w
AvgPool2d(k=2)	out_planes $\times$ h/2 $\times$ w/2
Discriminator	
Input	
Caption Embedding	nef
Image scale 4×4	$3 \times 4 \times 4$
Image scale 8×8	$3 \times 8 \times 8$
Image scale 16×16	$3 \times 16 \times 16$
Image scale 32×32	$3 \times 32 \times 32$
Image scale 64×64	$3 \times 64 \times 64$
Image scale 128×128	$3 \times 128 \times 128$
Image scale 256×256	$3 \times 256 \times 256$
Computation	
Image scale 256×256	$3 \times 256 \times 256$
Conv(k=1, s=1, p=0, b=True)	$ndf \times 256 \times 256$
DisGeneralConvBlock(ndf, 1, $2 * ndf$)	$ndf * 2 \times 128 \times 128$
Concat w/ Image scale 128×128	$ndf * 2 + 3 \times 128 \times 128$
DisGeneralConvBlock($2 * ndf$, 4, $4 * ndf$)	$4 * ndf \times 64 \times 64$
Concat w/ Image scale 64×64	$4 * ndf + 3 \times 64 \times 64$
DisGeneralConvBlock($4 * ndf$, 4, $8 * ndf$)	$8 * ndf \times 32 \times 32$
Concat w/ Image scale 32×32	$8 * ndf + 3 \times 32 \times 32$
DisGeneralConvBlock($8 * ndf$, 4, $8 * ndf$)	$8 * ndf \times 16 \times 16$
Concat w/ Image scale 16×16	$8 * ndf + 3 \times 16 \times 16$
DisGeneralConvBlock($8 * ndf$, 4, $8 * ndf$)	$8 * ndf \times 8 \times 8$
Concat w/ Image scale 8×8	$8 * ndf + 3 \times 8 \times 8$
DisGeneralConvBlock($8 * ndf$, 4, $8 * ndf$)	$8 * ndf \times 4 \times 4$
Unconditional logits	
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1
Conditional logits	
Caption Embedding	nef
Concat w/ replicated caption embedding	$8 * ndf + nef \times 4 \times 4$
Block3x3_leakyReLU	$8 * ndf \times 4 \times 4$
Conv(k=4, s=4, p=0, b=True)	1
Sigmoid	1

Table 9. Training settings of both AttnGAN++ and counter model. Most of settings in evaluation process is the same with training process except word_num. In the evaluation process, word_num=25 for the CUB [23] dataset and word_num=20 for the MS-COCO [14] dataset.

Dataset	CUB [23]	MS-COCO [14]
Optimizer	Adam($\beta_1 = 0.5$, $\beta_2 = 0.999$)	Adam($\beta_1 = 0.5$, $\beta_2 = 0.999$)
Generator (G) Learning Rate	0.0002	0.0002
Discriminator (D) Learning Rate	0.0002	0.0002
G/D Update	1 : 1	1 : 1
γ_1	4.0	4.0
γ_2	5.0	5.0
γ_3	10.0	10.0
λ	5.0	50.0
residual_num	2	3
ngf	64	64
ndf	32	32
nef	256	256
nzf	100	100
ncf	100	100
max_epochs	800	200
word_num	18	12

Fig. 8. Visualization of generated images from captions on the test set of CUB [23] dataset by **our AttnGAN++** using t-SNE [15]. The number of clusters in the visualization shows that the photos generated by our model span a wide range of bird species. As a result of the similar appearances of various bird species, some clusters are near together and overlap slightly. We also found no intra-class mode dropping, indicating that the model does not create the same sample in each bird class over and over. As can be seen in each cluster, the samples are belonging to one bird class with a variety of poses, and backgrounds. Best viewed in zoom.

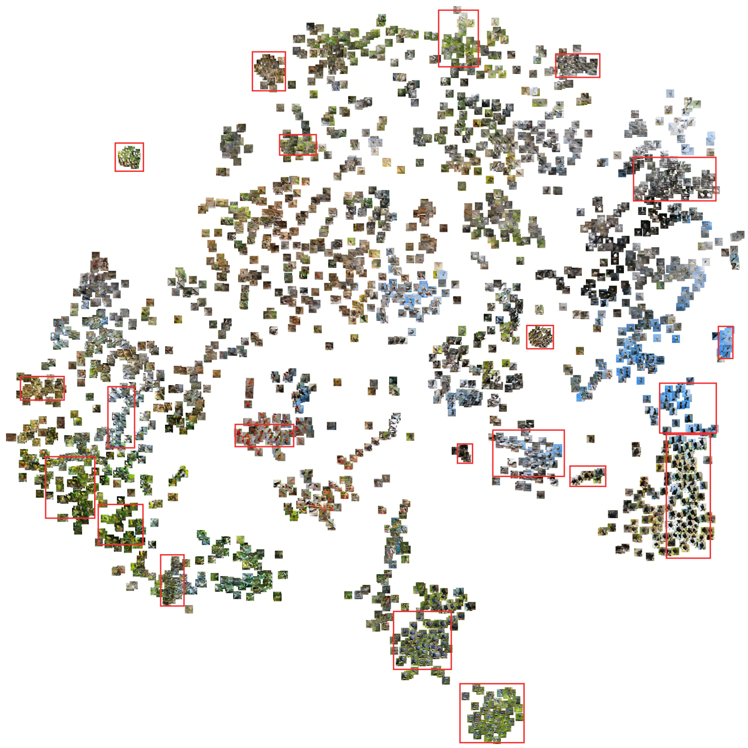


Fig. 9. Visualization of generated images from captions on the test set of CUB [23] dataset by **counter model** using t-SNE [15]. As can be seen, the images of counterexample are not realistic. The counter model tends to generate only one sample again and again per class that are surrounded by red squares in the visualization. Best viewed in zoom.

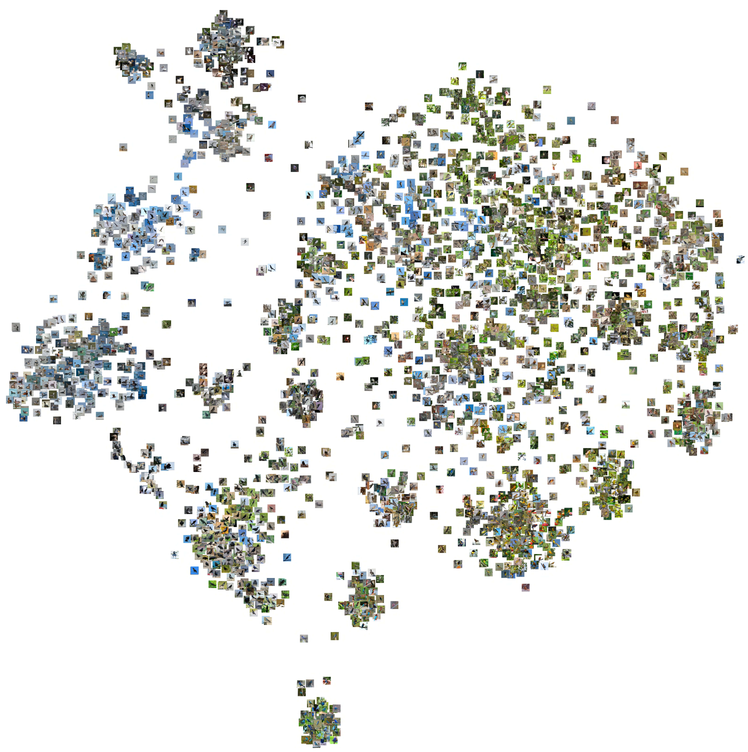


Fig. 10. Visualization of **real images** from CUB [23] test set by using t-SNE [15]. Best viewed in zoom.

References

1. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018) 2
2. Dayma, B., Patil, S., Cuenca, P., Saifullah, K., Abraham, T., Le Khac, P., Melas, L., Ghosh, R.: Dall·e mini (2021), https://github.com/borisdayma/dalle-mini 7
3. Frolov, S., Hinz, T., Raue, F., Hees, J., Dengel, A.: Adversarial text-to-image synthesis: A review. Neural Networks (2021) 4
4. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028 (2017) 2
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015) 9
6. Kang, M., Park, J.: Contragan: Contrastive learning for conditional image generation (2020) 2
7. Karnewar, A., Wang, O.: Msg-gan: Multi-scale gradients for generative adversarial networks. In: CVPR (2020) 3
8. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017) 13
9. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019) 13
10. Kodali, N., Abernethy, J., Hays, J., Kira, Z.: On convergence and stability of gans. arXiv preprint arXiv:1705.07215 (2017) 2
11. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N (2015) 1
12. Liang, J., Pei, W., Lu, F.: Cpgan: Full-spectrum content-parsing generative adversarial networks for text-to-image synthesis. arXiv preprint arXiv:1912.08562 (2019) 4, 5, 7
13. Lim, J.H., Ye, J.C.: Geometric gan. arXiv preprint arXiv:1705.02894 (2017) 2
14. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014) 4, 5, 7, 11, 14
15. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(Nov) (2008) 8, 15, 16, 17
16. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018) 2, 7, 9
17. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. ICML (2017) 2
18. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020 (2021) 4
19. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015) 2
20. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text-to-image synthesis. In: ICML (2016) 7
21. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3) (2015) 1

22. Tao, M., Tang, H., Wu, F., Jing, X.Y., Bao, B.K., Xu, C.: Df-gan: A simple and effective baseline for text-to-image synthesis. In: CVPR (2022) 7
23. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-ucsd birds 200. Tech. Rep. CNS-TR-2010-001, California Institute of Technology (2010) 7, 10, 14, 15, 16, 17
24. Wu, Y., Donahue, J., Balduzzi, D., Simonyan, K., Lillicrap, T.: Logan: Latent optimisation for generative adversarial networks. arXiv preprint arXiv:1912.00953 (2019) 2
25. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., He, X.: Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In: CVPR (2018) 4, 5, 7, 8, 9, 12, 13
26. Ye, H., Yang, X., Takac, M., Sunderraman, R., Ji, S.: Improving text-to-image synthesis using contrastive learning. arXiv preprint arXiv:2107.02423 (2021) 7
27. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: ICML (2019) 2
28. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: ICCV (2017) 4, 5, 7
29. Zhu, M., Pan, P., Chen, W., Yang, Y.: Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In: CVPR (2019) 4, 5, 7