CENG 796
Deep Generative Models

Generative Adversarial Networks - Part III
(Important developments in GANs, cont'd)

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

GANs - so far

- Fundamentals
- DC-GAN: importance of architecture
- Techniques to improve GAN training.
- WGAN
  - More achitecture agnostic
  - 1-Lipschitz-ness problem in critique (discriminator) training
- WGAN-GP: a better way to ensure 1-Lipschitz D().
- Today: even a better way to ensure 1-Lipschitz D(), class conditional GANs, self-attention, scaling-up GANs, and StyleGAN.

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

SPECTRAL NORMALIZATION FOR GENERATIVE ADVERSARIAL NETWORKS

Takeru Miyato[1], Toshiki Kataoka[1], Masanori Koyama[2], Yuichi Yoshida[3]
{miyato, kataoka}@preferred.jp
koyama.masanori@gmail.com
yyoshida@nii.ac.jp
[1]Preferred Networks, Inc. [2]Ritsumeikan University [3]National Institute of Informatics

ABSTRACT

One of the challenges in the study of generative adversarial networks is the instability of its training. In this paper, we propose a novel weight normalization technique called spectral normalization to stabilize the training of the discriminator. Our new normalization technique is computationally light and easy to incorporate into existing implementations. We tested the efficacy of spectral normalization on CIFAR10, STL-10, and ILSVRC2012 dataset, and we experimentally confirmed that spectrally normalized GANs (SN-GANs) is capable of generating images of better or equal quality relative to the previous training stabilization techniques. The code with Chainer (Tokui et al., 2015), generated images and pretrained models are available at https://github.com/pfnet-research/sngan_projection.

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

$$f(\boldsymbol{x}, \theta) = W^{L+1} a_L(W^L(a_{L-1}(W^{L-1}(\dots a_1(W^1 \boldsymbol{x}) \dots))))$$

$$D(\boldsymbol{x}, \theta) = \mathcal{A}(f(\boldsymbol{x}, \theta))$$

$$\min_G \max_D V(G, D)$$

$$\arg\max_{\|f\|_{\text{Lip}} \leq K} V(G, D)$$

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

- Key idea: Connecting Lipschitzness of discriminator to Lipschitzness & spectral norm of each layer.

$$g : \boldsymbol{h}_{in} \mapsto \boldsymbol{h}_{out}$$

$$\|g\|_{\text{Lip}} = \sup_{\boldsymbol{h}} \sigma(\nabla g(\boldsymbol{h}))$$

$$\sigma(A) := \max_{\boldsymbol{h}:\boldsymbol{h}\neq \mathbf{0}} \frac{\|A\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = \max_{\|\boldsymbol{h}\|_2 \leq 1} \|A\boldsymbol{h}\|_2$$

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

For a linear layer g

$$\|g\|_{\text{Lip}} = \sup_{\boldsymbol{h}} \sigma(\nabla g(\boldsymbol{h})) = \sup_{\boldsymbol{h}} \sigma(W) = \sigma(W)$$

In general, with non-linear activations, if activation function is 1-Lipschitz: (eg. ReLU, LeakyReLU and many others)

$$\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$$

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

$$\|g\|_{\text{Lip}} = \sup_{\boldsymbol{h}} \sigma(\nabla g(\boldsymbol{h})) = \sup_{\boldsymbol{h}} \sigma(W) = \sigma(W)$$

$$\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$$

Finally, the Lipschitz norm of the full discriminator (with 1-Lipschitz activations):

$$\|f\|_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^{L}\boldsymbol{h}_{L-1})\|_{\text{Lip}}$$

$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^{1}\boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^{l}\boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l)$$

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

$$\|g\|_{\text{Lip}} = \sup_{\boldsymbol{h}} \sigma(\nabla g(\boldsymbol{h})) = \sup_{\boldsymbol{h}} \sigma(W) = \sigma(W)$$

$$\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$$

Finally, the Lipschitz norm of the full discriminator (with 1-Lipschitz activations):

$$\|f\|_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^{L}\boldsymbol{h}_{L-1})\|_{\text{Lip}}$$

$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^1\boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^l\boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l)$$

$$\bar{W}_{\text{SN}}(W) := W/\sigma(W)$$

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

$$\|g\|_{\text{Lip}} = \sup_{\boldsymbol{h}} \sigma(\nabla g(\boldsymbol{h})) = \sup_{\boldsymbol{h}} \sigma(W) = \sigma(W)$$

$$\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$$

Finally, the Lipschitz norm of the full discriminator (with 1-Lipschitz activations):

$$\|f\|_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^{L}\boldsymbol{h}_{L-1})\|_{\text{Lip}}$$

$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^{1}\boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^{l}\boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l)$$

$$\bar{W}_{\text{SN}}(W) := W/\sigma(W)$$

Miyato et al 2017

Use power-iteration for fast approximation of SN.

Spectral Normalization GAN (SNGAN)

$$V(G, D) := \mathop{\mathrm{E}}_{x \sim q_{\mathrm{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathop{\mathrm{E}}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

For G updates: $-\mathrm{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))]$

$$V_D(\hat{G}, D) = \mathop{\mathrm{E}}_{\boldsymbol{x} \sim q_{\mathrm{data}}(\boldsymbol{x})}[\min(0, -1 + D(\boldsymbol{x}))] + \mathop{\mathrm{E}}_{\boldsymbol{z} \sim p(\boldsymbol{z})}\left[\min\left(0, -1 - D\left(\hat{G}(\boldsymbol{z})\right)\right)\right]$$

$$V_G(G, \hat{D}) = -\mathop{\mathrm{E}}_{\boldsymbol{z} \sim p(\boldsymbol{z})}\left[\hat{D}\left(G(\boldsymbol{z})\right)\right],$$

Geometric GAN

Jae Hyun Lim[1], **Jong Chul Ye**[2,3]
[1] ETRI, South Korea
jaehyun.lim@etri.re.kr
[2] Dept. of Bio and Brain engineering, KAIST, South Korea

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
dense, $4 \times 4 \times 1024$
ResBlock up 1024
ResBlock up 512
ResBlock up 256
ResBlock up 128
ResBlock up 64
BN, ReLU, 3×3 conv 3
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down 64
ResBlock down 128
ResBlock down 256
ResBlock down 512
ResBlock down 1024
ResBlock 1024
ReLU
Global sum pooling
dense $\rightarrow$ 1

(b) Discriminator for unconditional GANs.

Miyato et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Making GANs class-conditional

Here, as a good example, we'll focus on the way it is done in Spectral GAN (mainly adaptation from other works):

- In G: *conditional batch normalization* (Dumoulin et al., 2017; de Vries et al., 2017)
- In D: *projection discriminator* (Miyato & Koyama, 2018)

Regular batch norm

```
# Batch Norm for convolution layers
def batch_normalization(x, *, eps=1e-8): # x in NHWC
  with tf.variable_scope('conv_bn'):
        mean, var = tf.nn.moments(x, [0, 1, 2], keep_dims=True)
        u = (x - mean)*tf.rsqrt(var + epsilon)
        scale, offset = tf.split(tf.get_variable('scale_offset',
                                                 [1, 1, 1, x.shape[-1].value] * 2))
              return u * scale + offset
```

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Conditional Batch Normalization

```
# Conditional Batch Norm for convolution layers
def conditional_batch_normalization(x, y, *, eps=1e-8): # x in NHWC
  # x: 4D [b, h, w, c]; y: 2D [b, num_classes] (one-hot encoded)
  with tf.variable_scope('cond_conv_bn'):
        mean, var = tf.nn.moments(x, [0, 1, 2], keep_dims=True)
        u = (x - mean)*tf.rsqrt(var + epsilon)
        scale, offset = tf.split(
                                    tf.layers.dense(y, 2 * x.shape[-1].value), 2, -1)
            return u * scale + offset
```

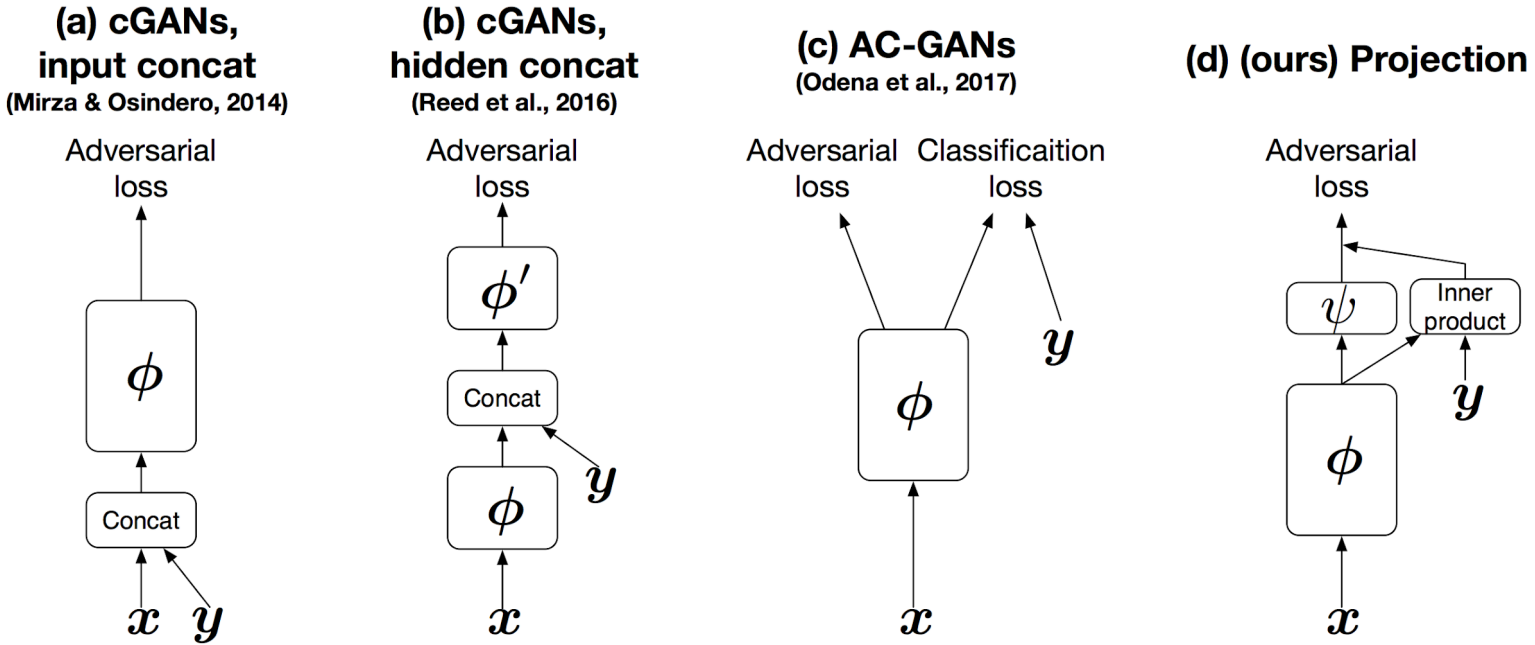
Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Conditional Batch Normalization with spec-norm

```
# Conditional Batch Norm (with SN) for convolution layers
def conditional_batch_normalization(x, y, *, eps=1e-8): # x in NHWC
  # x: 4D [b, h, w, c]; y: 2D [b, num_classes] (one-hot encoded)
  with tf.variable_scope('cond_conv_bn'):
        mean, var = tf.nn.moments(x, [0, 1, 2], keep_dims=True)
        u = (x - mean)*tf.rsqrt(var + epsilon)
        scale, offset = tf.split(
                                    dense(y, 2 * x.shape[-1].value, use_sn=True), 2, -1)
            return u * scale + offset
```

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Projection Discriminator



Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Spectral Normalization GAN (SNGAN)

Miyato et al 2017

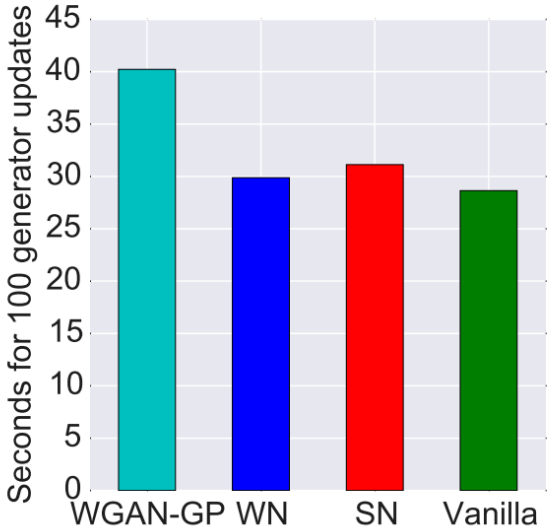
Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

SNGAN: Summary

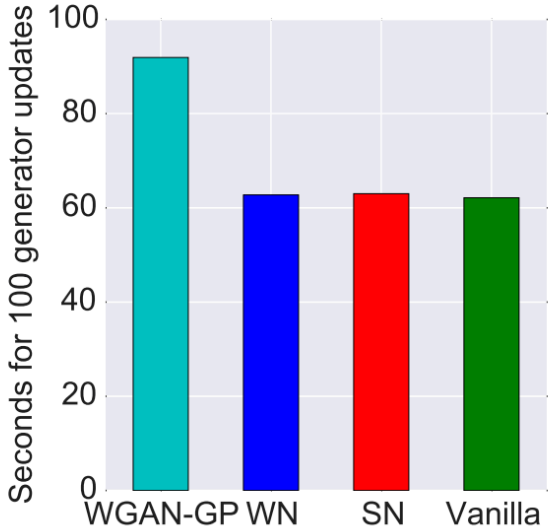
- High quality class conditional samples at Imagenet scale
- First GAN to work on full Imagenet (million image dataset)
- Computational benefits over WGAN-GP (single power iteration and no need of a backward pass)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

SNGAN: Computational Benefits



(a) CIFAR-10 (image size:32 × 32 × 3)



(b) STL-10 (images size:48 × 48 × 3)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Self Attention GAN (SAGAN)

Self-Attention Generative Adversarial Networks

Han Zhang* Rutgers University

Ian Goodfellow Google Brain

Dimitris Metaxas Rutgers University

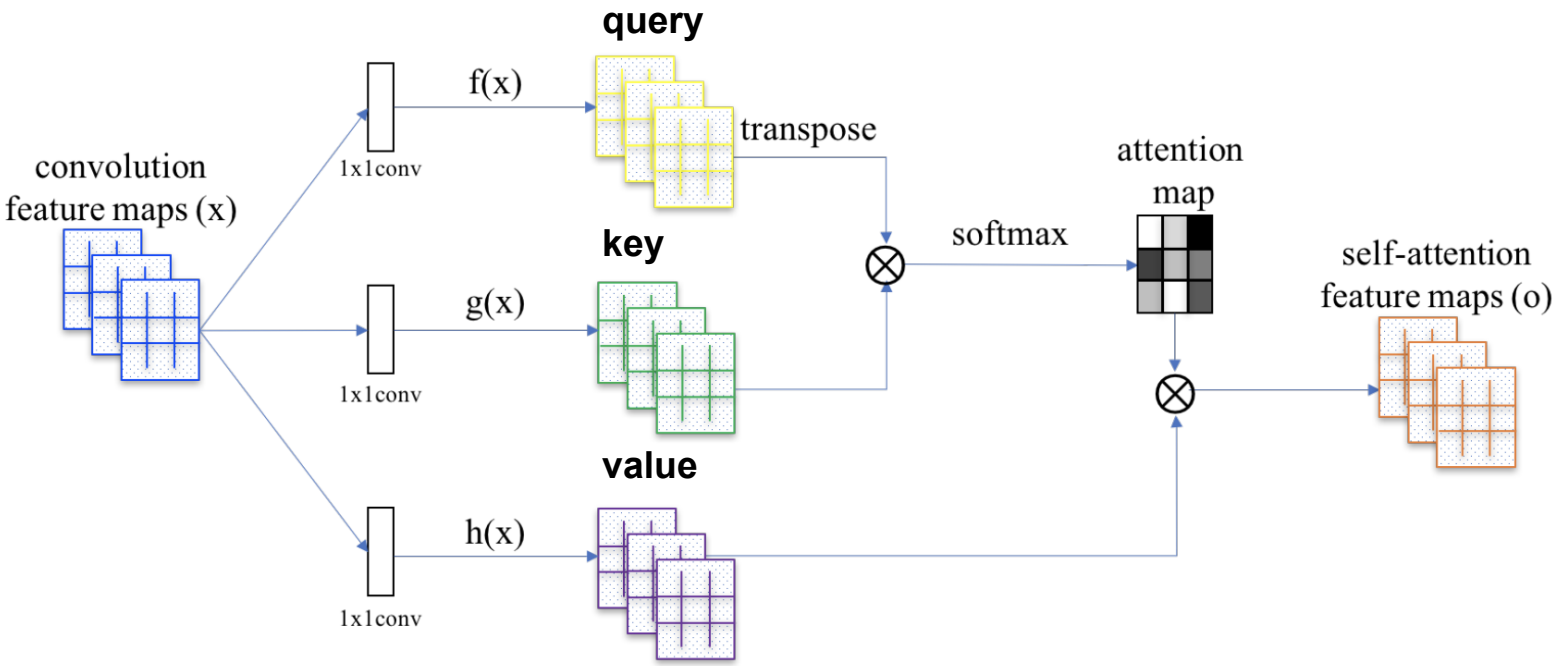
Augustus Odena Google Brain

Abstract

In this paper, we propose the Self-Attention Generative Adversarial Network (SAGAN) which allows attention-driven, long-range dependency modeling for image generation tasks. Traditional convolutional GANs generate high-resolution details as a function of only spatially local points in lower-resolution feature maps. In SAGAN, details can be generated using cues from all feature locations. Moreover, the discriminator can check that highly detailed features in distant portions of the image are consistent with each other. Furthermore, recent work has shown that generator conditioning affects GAN performance. Leveraging this insight, we apply spectral normalization to the GAN generator and find that this improves training dynamics. The proposed SAGAN achieves the state-of-the-art results, boosting the best published Inception score from 36.8 to 52.52 and reducing Fréchet Inception distance from 27.62 to 18.65 on the challenging ImageNet dataset. Visualization of the attention layers shows that the generator leverages neighborhoods that correspond to object shapes rather than local regions of fixed shape.

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Self Attention GAN (SAGAN)



Zhang et al 2018

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Self Attention GAN (SAGAN)

$$\boldsymbol{f(x) = W_f x},\ \boldsymbol{g(x) = W_g x}$$

$$s_{ij} = \boldsymbol{f(x_i)}^T \boldsymbol{g(x_j)}$$

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})}$$

$$\boldsymbol{y_i} = \gamma \boldsymbol{o_i} + \boldsymbol{x_i}$$

Zhang et al 2018

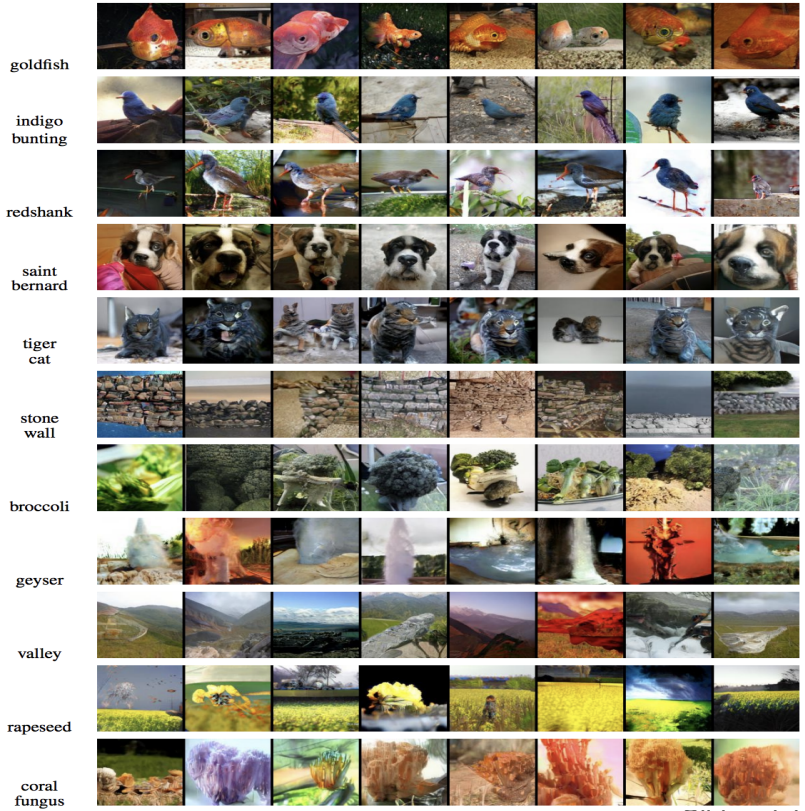
Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Self Attention GAN (SAGAN)

- Salient bits:
  - Applies spectral normalization to both the generator and discriminator weight matrices
    - This is counter-intuitive to popular belief that you only have to mathematically condition the discriminator
  - Uses self-attention in both the generator and discriminator
  - Hinge Loss
  - First GAN to produce “good” unconditional full Imagenet samples
  - Conditional models
    - Conditional BN for G, Projection Discriminator for D

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Self Attention GAN (SAGAN)



Zhang et al 2018

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Self Attention GAN (SAGAN)

Model	Inception Score	FID
AC-GAN [31]	28.5	/
SNGAN-projection [17]	36.8	27.62*
SAGAN	**52.52**	**18.65**

Table 2: Comparison of the proposed SAGAN with state-of-the-art GAN models [19, 17] for class conditional image generation on ImageNet. FID of SNGAN-projection is calculated from officially released weights.

Zhang et al 2018

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN

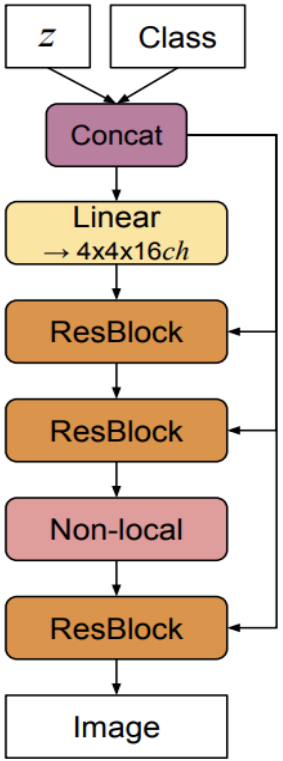
$$R_\beta(W) = \beta \|W^\top W - I\|_{\mathrm{F}}^2$$

$$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_{\mathrm{F}}^2,$$

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN

- Salient bits
  - Increase your batch size (as much as you c
  - Use Cross-Replica (Sync) Batch Norm
  - Increase your model size
  - Wider helps as much as deeper
  - Fuse class information at all levels
  - Hinge Loss
  - Self-attention
  - Orthonormal regularization & Truncation Tri



Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN

Batch	Ch.	Param (M)	Shared	Skip-z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	✗	✗	✗	1000	15.30	58.77($\pm$1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03($\pm$1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85($\pm$3.83)
2048	96	173.5	✗	✗	✗	295($\pm$18)	9.54($\pm$0.62)	92.98($\pm$4.27)
2048	96	160.6	✓	✗	✗	185($\pm$11)	9.18($\pm$0.13)	94.94($\pm$1.32)
2048	96	158.3	✓	✓	✗	152($\pm$7)	8.73($\pm$0.45)	98.76($\pm$2.84)
2048	96	158.3	✓	✓	✓	165($\pm$13)	8.51($\pm$0.32)	99.31($\pm$2.10)
2048	64	71.3	✓	✓	✓	371($\pm$7)	10.48($\pm$0.10)	86.90($\pm$0.61)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN

Model	Res.	FID/IS	(min FID) / IS	FID / (valid IS)	FID / (max IS)
SN-GAN	128	27.62/36.80	N/A	N/A	N/A
SA-GAN	128	18.65/52.52	N/A	N/A	N/A
BigGAN	128	$8.7 \pm .6/98.8 \pm 3$	$7.7 \pm .2/126.5 \pm 0$	$9.6 \pm .4/166.3 \pm 1$	$25 \pm 2/206 \pm 2$
BigGAN	256	$8.7 \pm .1/142.3 \pm 2$	$7.7 \pm .1/178.0 \pm 5$	$9.3 \pm .3/233.1 \pm 1$	$25 \pm 5/291 \pm 4$
BigGAN	512	8.1/144.2	7.6/170.3	11.8/241.4	27.0/275
BigGAN-deep	128	$5.7 \pm .3/124.5 \pm 2$	$6.3 \pm .3/148.1 \pm 4$	$7.4 \pm .6/166.5 \pm 1$	$25 \pm 2/253 \pm 11$
BigGAN-deep	256	$6.9 \pm .2/171.4 \pm 2$	$7.0 \pm .1/202.6 \pm 2$	$8.1 \pm .1/232.5 \pm 2$	$27 \pm 8/317 \pm 6$
BigGAN-deep	512	7.5/152.8	7.7/181.4	11.5/241.5	39.7/298

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN - Truncation Trick

(a)

(b)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN - Sampling

The default behavior with batch normalized classifier networks is to use a running average of the activation moments at test time. Previous works (Radford et al., 2016) have instead used batch statistics when sampling images. While this is not technically an invalid way to sample, it means that results are dependent on the test batch size (and how many devices it is split across), and further complicates reproducibility.

We find that this detail is extremely important, with changes in test batch size producing drastic changes in performance. This is further exacerbated when one uses exponential moving averages of **G**'s weights for sampling, as the BatchNorm running averages are computed with non-averaged weights and are poor estimates of the activation statistics for the averaged weights.

To counteract both these issues, we employ "standing statistics," where we compute activation statistics at sampling time by running the **G** through multiple forward passes (typically 100) each with different batches of random noise, and storing means and variances aggregated across all forward passes. Analogous to using running statistics, this results in **G**'s outputs becoming invariant to batch size and the number of devices, even when producing a single sample.

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN

(a) 128×128

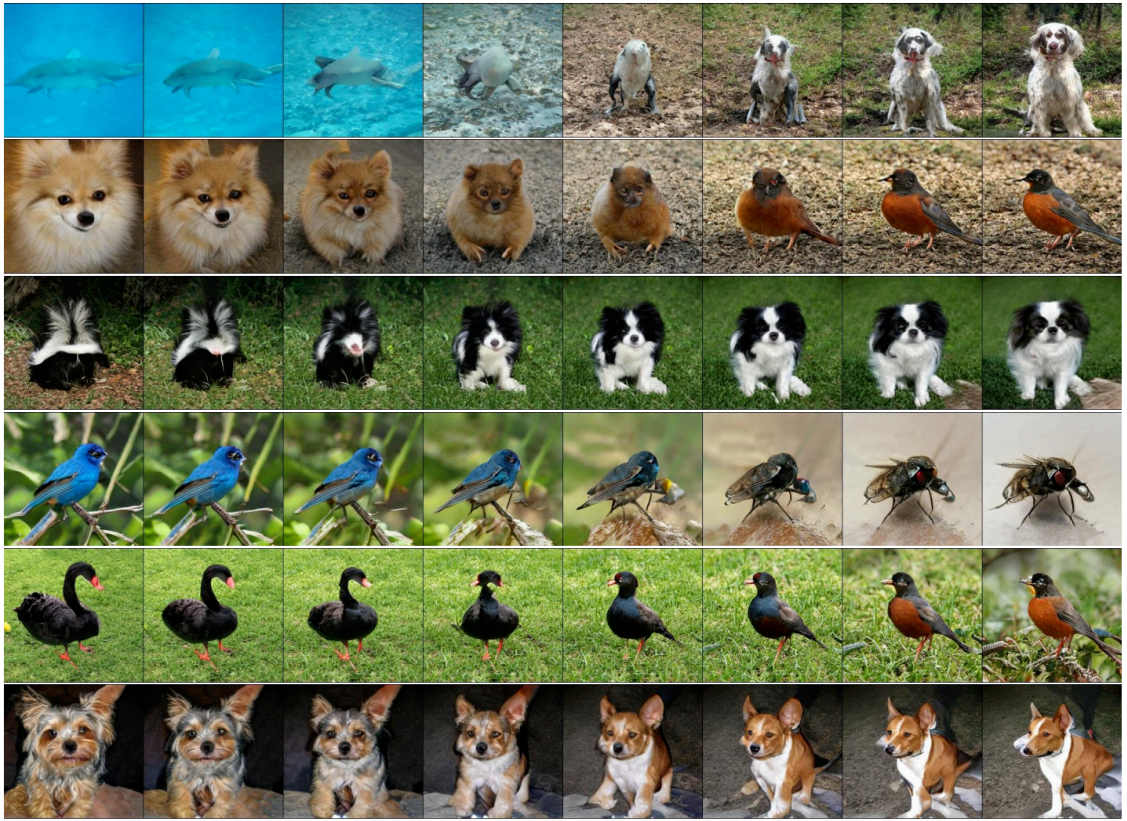
(b) 256×256

(c) 512×512

(d)

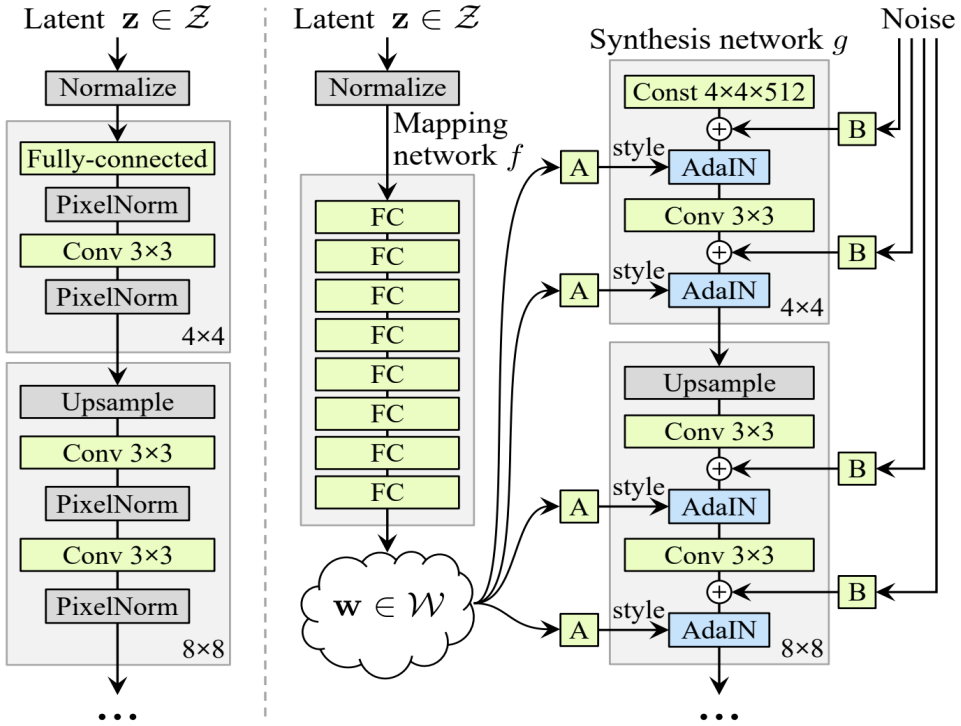
Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

BigGAN



Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN



(a) Traditional (b) Style-based generator

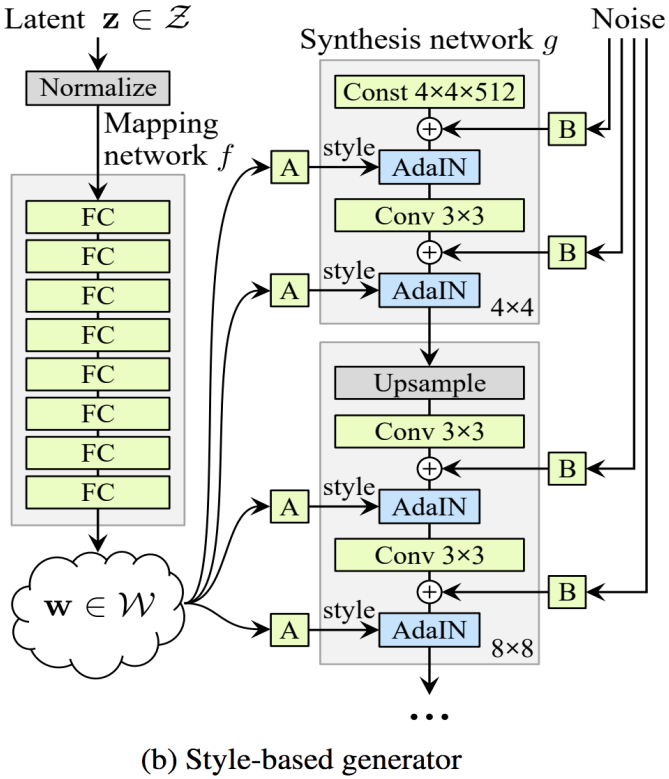
Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN - Adaptive Instance Norm

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN - Style Transfer



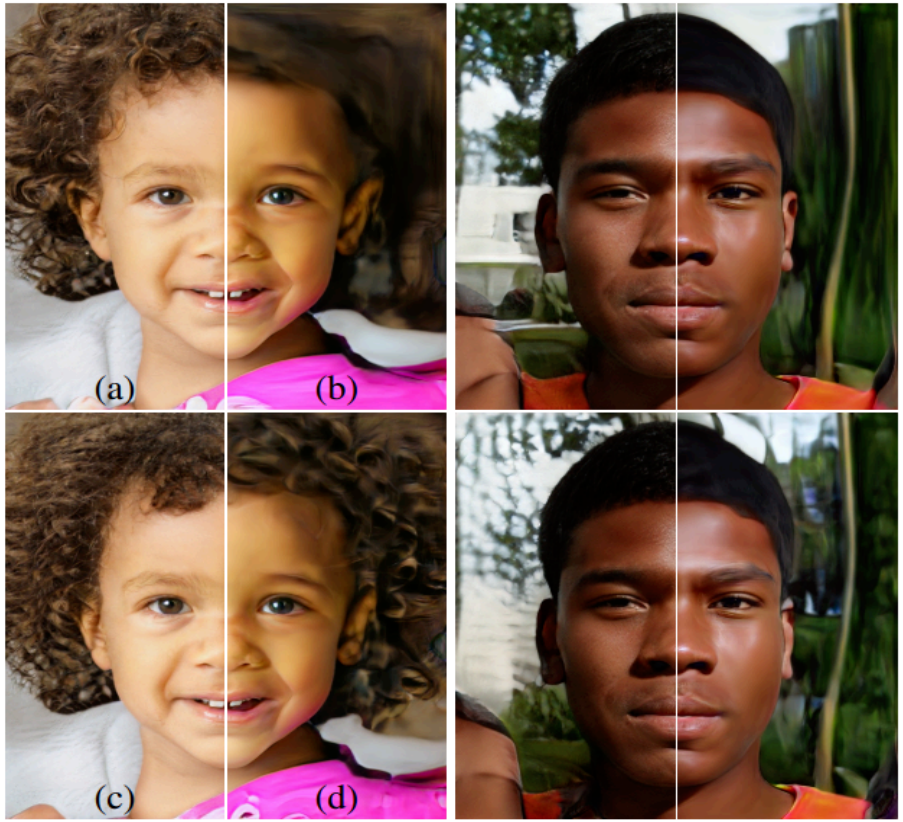
(b) Style-based generator

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN - Effect of adding noise



Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN-v2

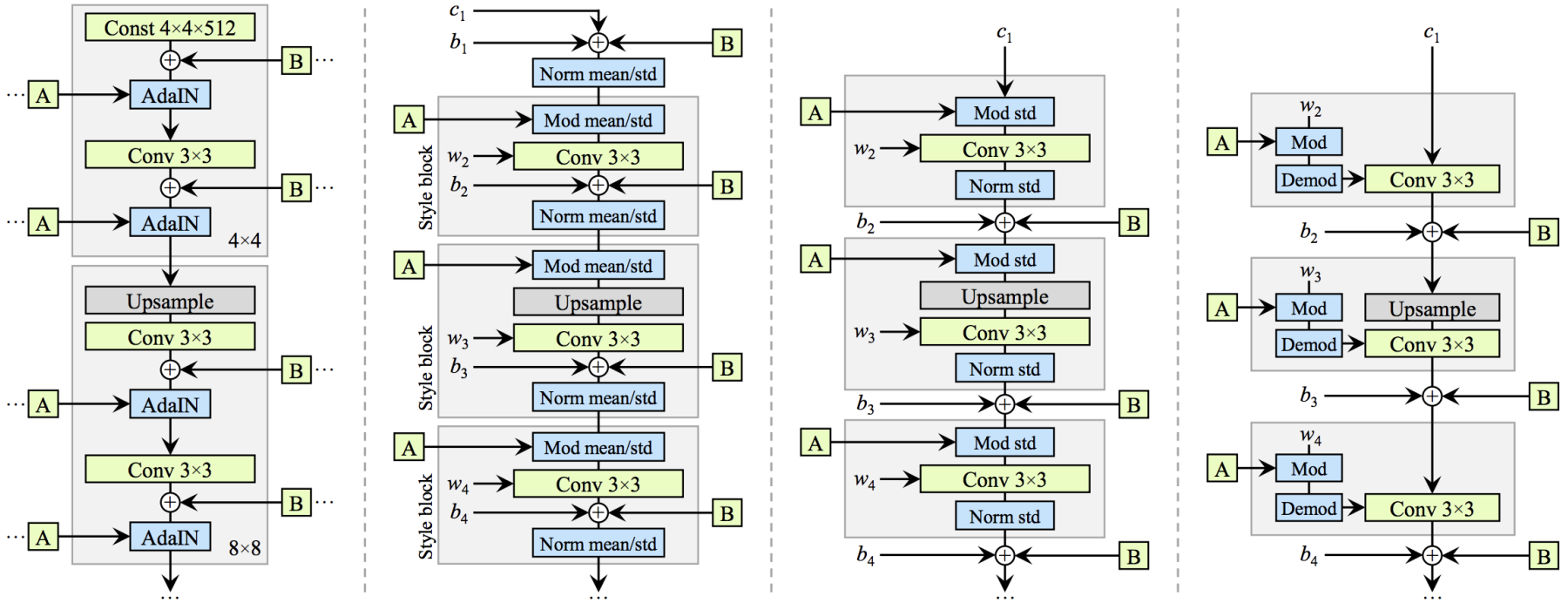
Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN-v2

We hypothesize that the droplet artifact is a result of the generator intentionally sneaking signal strength information past instance normalization: by creating a strong, localized spike that dominates the statistics, the generator can effectively scale the signal as it likes elsewhere.

StyleGAN-v2



(a) StyleGAN (b) StyleGAN (detailed) (c) Revised architecture (d) Weight demodulation

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

StyleGAN-v2

https://www.youtube.com/watch?time_continue=1&v=c-NJtV9Jvp0&feature=emb_logo

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Summary

To sum up, we have covered:

- Spectral normalization for learning 1-Lipschitz D()
- How to construct conditional GANs
- Self-attention for GANs
- BigGAN -- scaling-up GANs
- StyleGAN v1 and v2.

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan