Improved Consistency Regularization for GANs[1]

August 26, 2020

[1] *Improved Consistency Regularization for GANs* by Zhao et al., 2020

Overview

- CR-GAN (consistency regularized) proposes to add self-supervision loss to D via augmentations
- Authors improve upon CR-GAN model in two ways:
  - Add augmentations on the generated images as well
  - Add self-supervision by augmenting z space as well
- They compare with other regularization strategies (WGAN-GP, JSR, CR-GAN, etc) and show SotA results on CIFAR-10, CelebA and ImageNet

Constistency Regularized GAN (CR-GAN)[2]

- Let $T(x)$ be an augmentation function: it takes image x and produces its augmented version
  - For example, randomly flips it, rotates, shifts, cutouts, etc
- CR-GAN adds the following loss to D:

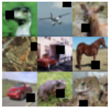
$$L_{\text{cr}} = \mathop{\mathbb{E}}_{p_{\text{real}}(x)} \left[\| D(x) - D(T(x)) \|_2^2 \right],$$

  where $D(x)$ is the output vector before the activation in D model.
- CR-GAN uses BigGAN as a base model, compares to other regularization strategies and shows SotA performance
- I.e. they improve FID for ImageNet 128x128 from 8.73 to 6.66
- The augmentation that worked the best is random flips + random shifts

[2] "Consistency Regularization for Generative Adversarial Networks" by Zhang et al., 2020

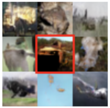
Generation artifacts in CR-GAN

Since CR-GAN applies $T(x)$ only for real data, D thinks that augmentations is a part of real distribution.


(a) 8×8 cutout.


(b) CR samples.


(d) 16×16 cutout.


(e) CR samples.


(g) 32×32 cutout.


(h) CR samples.

This limits the set of augmentations we can use.

Balanced Consistency Regularization (bCR)

Authors remove the artifacts by adding augmentations to G as well

Algorithm 1 Balanced Consistency Regularization (bCR)

Input: parameters of generator θ_G and discriminator θ_D, consistency regularization coefficient for real images λ_{real} and fake images λ_{fake}, number of discriminator iterations per generator iteration N_D, augmentation transform T (for images, e.g. shift, flip, cutout, etc).

for number of training iterations **do**
  for $t = 1$ **to** N_D **do**
    Sample batch $z \sim p(z)$, $x \sim p_{\text{real}}(x)$
    Augment both real $T(x)$ and fake $T(G(z))$ images
    $L_D \leftarrow D(G(z)) - D(x)$
    $L_{\text{real}} \leftarrow \|D(x) - D(T(x))\|^2$
    $L_{\text{fake}} \leftarrow \|D(G(z)) - D(T(G(z)))\|^2$
    $\theta_D \leftarrow \text{AdamOptimizer}(L_D + \lambda_{\text{real}} L_{\text{real}} + \lambda_{\text{fake}} L_{\text{fake}})$
  end for
  Sample batch $z \sim p(z)$
  $L_G \leftarrow -D(G(z))$
  $\theta_G \leftarrow \text{AdamOptimizer}(L_G)$
end for

They usually set $\lambda_{\text{real}} = \lambda_{\text{fake}} = 10$.

Latent Consistency Regularization (zCR)

- Motivation: $D(G(z))$ should not change much when we change z a little
- This leads to the following additional loss

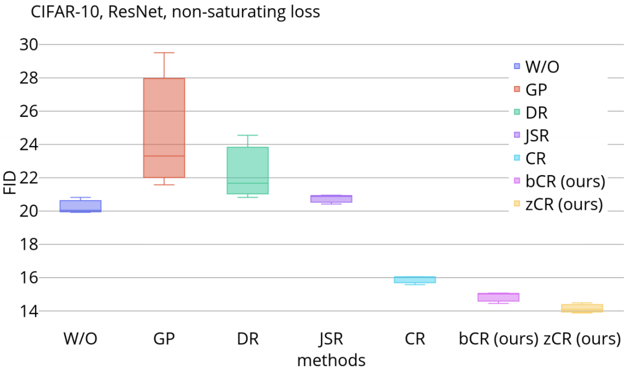
$$\|D(G(z)) - D(G(T(z)))\|_2^2$$

  where $T(z) = z + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)$.
- This loss on its own leads to mode collapse (since G tries to output same images for different z)
- This problem is alleviated by forcing G to output different images for close z

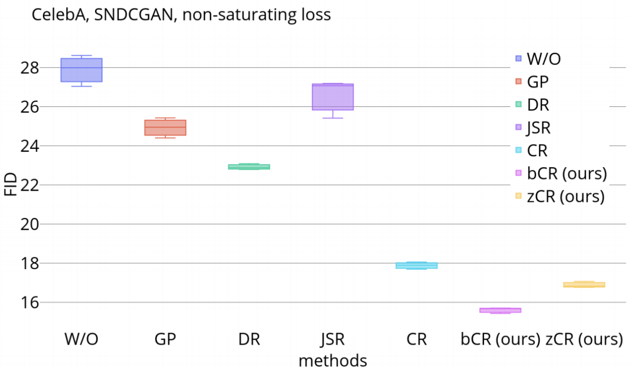
$$-\|G(z) - G(T(z))\|_2^2$$

- This loss on G is applied with 10-20 times lower weight

Results on CIFAR-10



Results on CelebA



Improved CR (ICR): use bCR + zCR simultaneously

Results for small models:

Methods	CIFAR-10 (DCGAN)	CIFAR-10 (ResNet)	CelebA (DCGAN)
W/O	24.73	19.00	25.95
GP	25.83	19.74	22.57
DR	25.08	18.94	21.91
JSR	25.17	19.59	22.17
CR	18.72	14.56	16.97
ICR (ours)	**15.87**	**13.36**	**15.43**

Results for BigGANs:

Models	CIFAR-10	ImageNet
SNGAN	17.50	27.62
BigGAN	14.73	8.73
CR-BigGAN	11.48	6.66
ICR-BigGAN (ours)	**9.21**	**5.38**

bCR is not very sensitive to loss weight

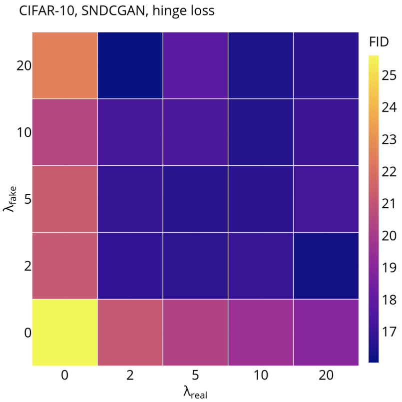


Figure: FID on CIFAR-10 for spectrally normalized DC-GAN

Final thoughts

- Number of additional forward passes for models:
  - CR-GAN: 1 for D
  - bCR-GAN: 2 for D
  - zCR-GAN: 1 for D, 1 for G, 1 for G inside D loop
- Maybe, it is possible not to apply it on each iter (aka *lazy regularization* from StyleGAN2)
- The overall direction of replacing "math-heavy" gradient penalties with self-supervision like losses is interesting, produces better results and trains faster