Run-Sort-ReRun

Escaping Batch Size Limitation in Sliced Wasserstein Generative Models

Authors: Jose Lezama, Wei Chen, Qiang Qiu

Reporter: Fengjiao Gong

Date: 2021/09/30

Outline

Run-Sort-ReRun

- **Background**
- **Method**
- **Experiments**

Background

Generative Adversarial Networks(GANs)

A. Generative Model

given a dataset $D = \{(x)\}$ with m samples $x \sim P_r$ where P_r is an unknown **real distribution**

task - the model learn a parametric transformation from a tractable base distribution to a **learned distribution** P_θ

goal - make P_θ as similar as possible to P_r

B. Generative Adversarial Networks(GANs)

- task - a distance minimization problems
- **generator** with parameter θ — generate synthetic data $y \sim P_\theta$

  discriminator — distinguish synthetic data from real data

both are parameterized by **deep neural networks** + trained via **stochastic gradient descent**

- problems - **standard** Jensen-Shannon **divergence** between P_r and P_θ

  A. gradient vanishing - **Wasserstein Distance**

  B. mode collapse

Background

Sliced Wasserstein Distance

C. $p-$Wasserstein Distance

Wasserstein Distance considers the minimal transformation between two distributions of points

$$W_p(P_r, P_\theta) = \inf_{\gamma \in \Pi(P_r, P_\theta)} \left(\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma}[\|\boldsymbol{x}-\boldsymbol{y}\|^p]\right)^{\frac{1}{p}}$$

since P_r is unknown, the infimum is **hard to compute**

D. Sliced $p-$Wasserstein Distance

$$\tilde{W}(P_r, P_\theta) = \left[\int_{\boldsymbol{\omega}\in\Omega} W_p^p(P_r^{\boldsymbol{\omega}}, P_\theta^{\boldsymbol{\omega}}) d\boldsymbol{\omega}\right]^{\frac{1}{p}}$$

where

$P_r^\omega, P_\theta^\omega$ denote the **projection** of P_r and P_θ onto the direction ω

Ω is the set of all possible directions on the unit sphere

Background

Sliced Wasserstein Distance

E. Sliced 2−Wasserstein Distance approximation

samples $D = \{(x)\}$ with $x \sim P_r$, **samples** $F = \{(y)\}$ with $y \sim P_\theta$

$$\min_{\boldsymbol{\theta}} \frac{1}{|\hat{\Omega}|} \sum_{\boldsymbol{\omega} \in \hat{\Omega}} W_2^2(D^{\boldsymbol{\omega}}, F^{\boldsymbol{\omega}})$$

$\hat{\Omega}$ is a randomly chosen set of unit vectors

F. for $1 - D$ distributions

$$W_2^2(D^{\boldsymbol{\omega}}, F^{\boldsymbol{\omega}}) = \frac{1}{|D|} \sum_i \|F^{\boldsymbol{\omega}}_{\pi_F(i)} - D^{\boldsymbol{\omega}}_{\pi_D(i)}\|_2^2$$

where $\pi_D(i)$ and $\pi_F(i)$ are permutations that sort the projected sample sets D^ω and F^ωrespectively

$$\text{i.e. } D^\omega_{\pi_D(1)} \leq D^\omega_{\pi_D(2)} \leq \ldots D^\omega_{\pi_D(|D|)}$$

when the number of samples increase, the empirical distributions D and F arbitrarily approximate their unknown continuous counterparts

Method

Motivating Example:

Suppose we learn **a distribution with four modes**(leftmost subfigure), a restrictive training batch size - **four samples per training batch**

only compares two distributions using relatively few samples, not enough to capture the complexity of the target distribution

Adversarial — **mode collapse** problem

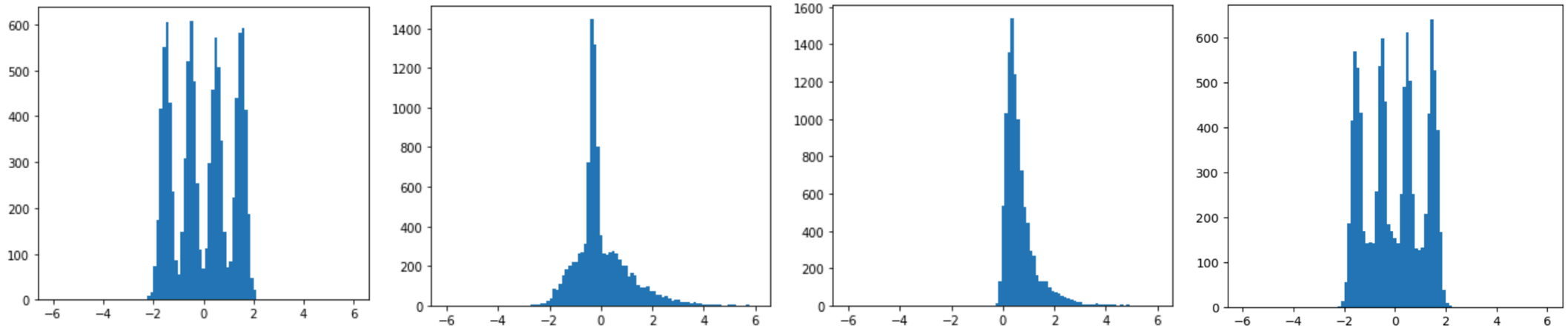


Figure 1. Comparison of empirical distributions obtained by different methods. In this example we show the shortcomings of training generative models with a batch size that is too small compared to the complexity of the target distribution, a common scenario under memory limitations. We train simple 1-D generative models to replicate a mixture of 4 Gaussians (leftmost), using batches of 4 samples. The standard GAN suffers from "mode collapse". Optimizing the empirical 1-D Wasserstein distance also fails to capture the 4 modes with such few samples. Our method, Run-Sort-ReRun is able to process one small batch (4 samples here) at a time, yet virtually support arbitrarily large batch sizes (in this case 1024), overcoming the memory limitations and successfully capturing all the modes of the target distribution.

Method

Batch Size Limitation:

memory constraints: the capacity of the GPU to store all the activations and gradients of the model for backpropagation

Run-Sort-ReRun:

the sliced Wasserstein distance is a regression problem:

$$W_2^2(D^{\boldsymbol{\omega}}, F^{\boldsymbol{\omega}}) = \frac{1}{|D|} \sum_i \|F^{\boldsymbol{\omega}}_{\pi_F(i)} - D^{\boldsymbol{\omega}}_{\pi_D(i)}\|_2^2$$

regression target of each generated sample => corresponding sample from the real distribution

$$i \to \pi_D^{-1} \circ \pi_F(i)$$

the central idea is to compute the regression target as they would be and when the number of samples is very large, the empirical distance arbitrarily approximates the actual distance between the continuous distributions

Method

Generative model:

$G(z;\theta)$ — generative model with parameter θ, and latent vector $z \in R^h$ is the input, output is a generated synthetic sample

Run-Sort-ReRun Algorithm:

① Run:

one small batch of size m at a time ($|D^{\omega}| \gg m$)

compute and store both real and synthetic empirical distributions Dω and Fω are computed and stored.

store the latent vectors z of the synthetic data so that they can be used to regenerate the synthetic samples in the ReRun step.

② Sort:

sort both stored distributions to obtain $\pi_D(i)$ and $\pi_F(i)$, so the regression targets are known

③ ReRun:

regenerate small batches of samples from the stored latent vectors and compare to the targets obtained in the previous step

backpropagate the Wasserstein loss fuction — automatic differentiation on

only store intermediate network activations and gradients for a small number of samples

Algorithm 1 Run-Sort-ReRun

Input: Real data $X \in \mathbb{R}^{d \times n}$, large batch size b, small batch size m, generator G with parameters $\boldsymbol{\theta}$, number of projections p, step size α, latent dimension d_z

repeat

(auto-diff OFF)

Initialize $Z \in \mathbb{R}^{d_z \times b}, D \in \mathbb{R}^{d \times b}, F \in \mathbb{R}^{d \times b}$: storage for latent vectors, real data and model output resp.

Initialize $\Omega \in \mathbb{R}^{d \times p}$: projection matrix

RUN

for $i = 1$ **to** b/m **do**

Z_i = m samples from $\mathcal{N}(0, I)$

$Z[:, (i-1) \times m : i \times m] \leftarrow Z_i$ *// column indexing*

$F[:, (i-1) \times m : i \times m] \leftarrow G(Z_i; \boldsymbol{\theta})$

$D[:, (i-1) \times m : i \times m] \leftarrow m$ samples from X

end for

$D = \Omega D$, $F = \Omega F$ *// project*

SORT

D = sort(D) *// row-wise*

Π_F^{-1} = argsort(argsort(F)) *// row-wise*

$D = D[:, \Pi_F^{-1}]$ *// aligns rows of F and D as per (6)*

(auto-diff ON)

RERUN

Initialize $\boldsymbol{g} = \mathbf{0}$

for $i = 1$ **to** b/m **do**

$Z_i = Z[:, (i-1) \times m : i \times m]$

$F_i = G(Z_i; \boldsymbol{\theta})$

$D_i = D[:, (i-1) \times m : i \times m]$

$c = ||\Omega F_i - D_i||_F^2 / m$

$\boldsymbol{g} = \boldsymbol{g} + \nabla_{\boldsymbol{\theta}} c$

end for

$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \boldsymbol{g}$

until convergence

Method

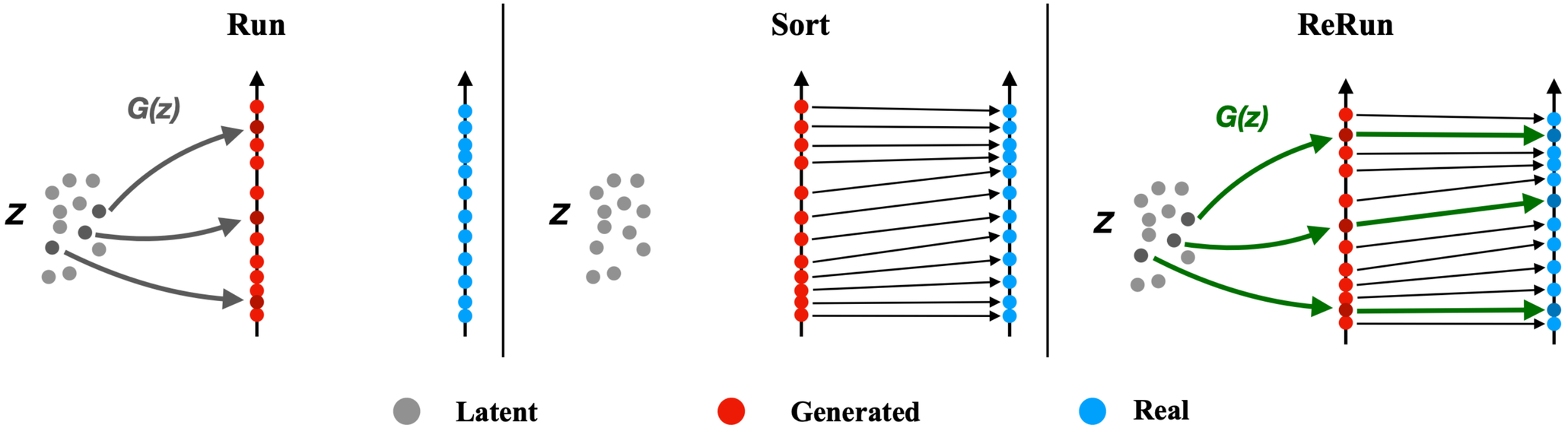


Figure 2. Schematic representation of the Run-Sort-ReRun algorithm. In this example, we assume that the generator G can only process 3 samples at a time, and that the desired full sample size is 12. We consider a single projection for illustration. In the **Run** step, the latent vectors $\boldsymbol{z}$ (gray) are used to generate synthetic samples (red), 3 at a time (dark gray/dark red). A group of 12 real samples (blue) is also collected. In the **Sort** step, the 12 synthetic and the 12 real samples are sorted, thus establishing the correspondences (straight arrows) for computing (5). In the **ReRun** step, synthetic samples are generated again, 3 at a time, from the same latent vectors as in the Run step. The distances to their correspondences (dark blue) are computed, so the gradient of the 1-D Wasserstein on the 12 samples can be computed, even though no more than 3 samples were ran through the generator at any time.

Algorithm 1 Run-Sort-ReRun

Input: Real data $X \in \mathbb{R}^{d \times n}$, large batch size b, small batch size m, generator G with parameters $\boldsymbol{\theta}$, number of projections p, step size α, latent dimension d_z

repeat

(auto-diff OFF)

Initialize $Z \in \mathbb{R}^{d_z \times b}, D \in \mathbb{R}^{d \times b}, F \in \mathbb{R}^{d \times b}$: storage for latent vectors, real data and model output resp.

Initialize $\Omega \in \mathbb{R}^{d \times p}$: projection matrix

RUN

for $i = 1$ **to** b/m **do**

Z_i = m samples from $\mathcal{N}(0, I)$

$Z[:, (i-1) \times m : i \times m] \leftarrow Z_i$ *// column indexing*

$F[:, (i-1) \times m : i \times m] \leftarrow G(Z_i; \boldsymbol{\theta})$

$D[:, (i-1) \times m : i \times m] \leftarrow m$ samples from X

end for

$D = \Omega D$, $F = \Omega F$ *// project*

SORT

D = sort(D) *// row-wise*

Π_F^{-1} = argsort(argsort(F)) *// row-wise*

$D = D[:, \Pi_F^{-1}]$ *// aligns rows of F and D as per (6)*

(auto-diff ON)

RERUN

Initialize $\boldsymbol{g} = \mathbf{0}$

for $i = 1$ **to** b/m **do**

$Z_i = Z[:, (i-1) \times m : i \times m]$

$F_i = G(Z_i; \boldsymbol{\theta})$

$D_i = D[:, (i-1) \times m : i \times m]$

$c = ||\Omega F_i - D_i||_F^2 / m$

$\boldsymbol{g} = \boldsymbol{g} + \nabla_{\boldsymbol{\theta}} c$

end for

$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \boldsymbol{g}$

until convergence

Method

Projection Directions:

the contribution to the sliced Wasserstein distance of a generated sample y_i under projection ω is given by

$$c_i = ((\boldsymbol{y}_i - \boldsymbol{x}_{[i]})^\top \boldsymbol{\omega})^2$$

where $x_{[i]}$is the corresponding real sample after the sorting.

Thus, if we wanted to steer ω to a direction of greater separation between the samples, the gradient $\nabla_\omega c_i$ would point towards $(y_i - x_{[i]})$

so construct the distribution of projection directions by sampling random pairs of generated and real samples

$$\hat{W}_p^p(P_r, P_\theta) = \mathbb{E}_{\boldsymbol{x}\sim P_r, \boldsymbol{y}\sim P_\theta}\left[W_p^p(P_r^{\omega(\boldsymbol{x},\boldsymbol{y})}, P_\theta^{\omega(\boldsymbol{x},\boldsymbol{y})})\right]$$

where $\omega(x, y) = (x - y)/ \parallel x - y \parallel$

Finally, to maintain a balance between exploiting informative directions and exploring new directions, in each new iteration we re-utilize a subset of the random directions that produced the largest separation according to the 1-D Wasserstein distance

Algorithm 2 Choice of Projection Directions

Input: Number of projections p, previous projection directions $\Omega_t \in \mathbb{R}^{p\times d}$, number of re-utilized directions r, real data $X \in \mathbb{R}^{d\times n}$, synthetic data $F \in \mathbb{R}^{d\times n}$

Compute $\boldsymbol{d}_{\Omega_t} \in \mathbb{R}^p$, vector of sliced distances for each row of Ω_t using *Run-Sort-ReRun*
Initialize $\Omega_{t+1} \in \mathbb{R}^{d\times p}$
$\boldsymbol{z}_t$ = argsort($\boldsymbol{d}_{\Omega_t}$) *// descending*
$\Omega_{t+1}[0{:}r, :] = \Omega_t[\boldsymbol{z}_t[0{:}r], :]$ *// keep rows with largest distances*
for $i = 1$ **to** $p - r$ **do**
  Sample $\boldsymbol{x}_i$ from X, $\boldsymbol{y}_i$ from F
  $\Omega_{t+1}[r + i, :] = \{(\boldsymbol{x}_i - \boldsymbol{y}_i)/\|\boldsymbol{x}_i - \boldsymbol{y}_i\|\}$
end for
return Ω_{t+1}

Experiments

FID score:

common metric for evaluating the performance of a generative model is the Frechet Inception Distance or FID score

by fitting Gaussian distributions to the real and generated image features extracted by an Inception network , and then computing the 2-Wasserstein distance (in closed form) between the two Gaussians

baseline — E2GAN

CIFAR-10 50,000 training images of size 32 × 32

STL-10 105,000 training images of size 96 × 96

large batch size b = 16000, p = 4000 projection directions and r = 1333

baseline — StyleGAN-v2

LSUN Church datasets images of size 256x256, using large batch size b = 8192 and p = 4000, r = 1333

Notably, the FID decreases significantly with respect to the baseline, and the scores outperform the most advanced GAN models

CIFAR-10

Method	FID ↓
E2GAN (Tian et al., 2020)	11.26
DiffAugment-CR-BigGAN (Zhao et al., 2020)	8.49
Mix-BigGAN (Tang, 2020)	8.17
StyleGAN-v2 + ADA (Karras et al., 2020a)	7.1
E2GAN + RSR (Ours)	**4.9**

STL-10

Method	FID ↓
Auto GAN (Gong et al., 2019)	31.01
DEGAS (Doveh & Giryes, 2019)	28.76
E2GAN (Tian et al., 2020)	25.35
E2GAN + RSR (Ours)	**8.6**

LSUN Church

Method	FID ↓
StyleGAN-v2 (Karras et al., 2020b)	3.86
StyleGAN-v2 + converted weights	5.93
StyleGAN-v2 + converted weights + RSR (Ours)	**3.14**

Table 1. FID scores on CIFAR-10, STL-10 and LSUN Church. For CIFAR-10 and STL-10, we finetune a baseline method (Tian et al., 2020) using the Run-Sort-ReRun (RSR) algorithm with large batch sizes of 16,000 samples. For LSUN Church, we finetune a StyleGAN-v2 model (Karras et al., 2020b) with 8192 samples per batch.

Experiments

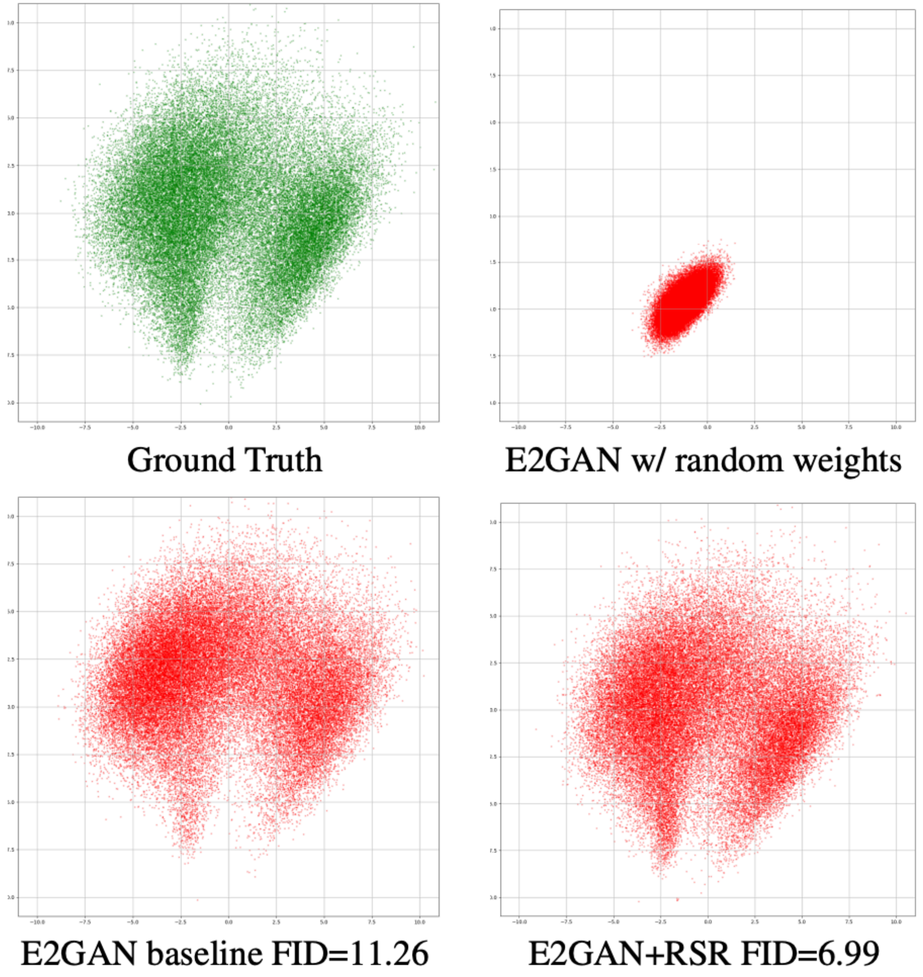


Figure 3. PCA projections of Inception features for CIFAR-10 dataset. Run-Sort-ReRun refines the distribution of samples generated by the baseline E2GAN (Tian et al., 2020) to more accurately match the ground truth distribution.

Experiments

Ablation Study on Effective Batch Size:

For different values of b, we adapt the number of iterations such that in all experiments the total number of samples seen during training is the same (in this case ~ 5M).

We observe a notorious decrease in FID score at b = 1024, a batch size that would be completely impractical under traditional methods

CIFAR-10	
Large batch size b	FID ↓
Baseline	11.26
RSR, $b = 256$	10.02
RSR, $b = 1024$	7.39
RSR, $b = 2048$	7.23
RSR, $b = 8192$	6.99

Table 2. Ablation study on the large batch size parameter b. We finetune the baseline method (Tian et al., 2020) using Run-Sort-ReRun (RSR) for different values of b. The number of iterations is adjusted so that all models see the same total number of samples.

Experiments

On Generated Image Quality:

there is not a notorious increase in image quality (see Figure 4).

Baseline: StyleGAN2 (Karras et al., 2020b), (FID 5.93)

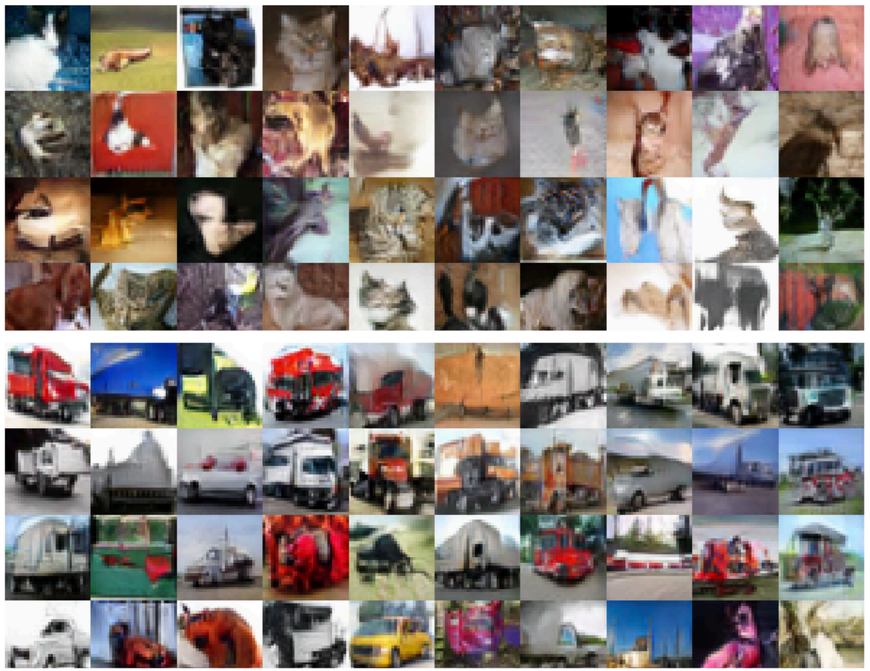
StyleGAN2 finetuned with RSR, (FID 3.14)

Figure 4. Results of StyleGAN2 on LSUN Churches[4]. We observe that directly optimizing the distribution of Inception features improves the FID score over the state-of-the-art methods.

Experiments

On Generated Image Quality:

improvements in generated image quality can be obtained with Run-Sort-ReRun by using **four** rather than just one layer of **the Inception network**



Baseline (Tian et al., 2020) (FID 25.35)

Baseline + RSR (FID 14.39)

Figure 5. Qualitative comparison between baseline E2GAN (Tian et al., 2020) and the same baseline finetuned with Run-Sort-ReRun on the features produced by four different layers of the Inception network, using an effective batch size of 16,000. An external pre-trained classifier was used on the generated samples to separate samples of different classes (top: cat, bottom: truck). Besides decreasing the FID score, the samples of the model trained with the proposed method exhibit better defined shapes. Best appreciated in electronic format.

Q&A

Thanks !