Large Scale GAN Training for High Fidelity Natural Image Synthesis [1]

February 19, 2020

[1] "Large Scale GAN Training for High Fidelity Natural Image Synthesis" by Brock, Donahue, and Simonyan

Main ideas

Main ideas

Main ideas

- Authors trained the largest GAN model (to the moment of the project)

Main ideas

- Authors trained the largest GAN model (to the moment of the project)
- Explored a lot of techniques of training GANs

Main ideas

- Authors trained the largest GAN model (to the moment of the project)
- Explored a lot of techniques of training GANs
- Proposed a “truncation trick” for better sampling at test-time

Main ideas

- Authors trained the largest GAN model (to the moment of the project)
- Explored a lot of techniques of training GANs
- Proposed a “truncation trick” for better sampling at test-time
- Dramatically improved SotA on ImageNet

Inception Score (IS)

Inception Score (IS)

1. Take Inception classifier

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples
3. For each sample x compute class probabilities $p(y|x)$

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples
3. For each sample x compute class probabilities $p(y|x)$
4. Average them to approximately compute $p(y) \approx \mathop{\mathbb{E}}_{p_g(x)} [p(y|x)]$

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples
3. For each sample x compute class probabilities $p(y|x)$
4. Average them to approximately compute $p(y) \approx \mathop{\mathbb{E}}_{p_g(x)}[p(y|x)]$
5. Compute *inception score*:

$$\mathrm{IS} = \exp\left(\mathop{\mathbb{E}}_{p_g(x)}\left[D_{KL}(p(y|x)||p(y))\right]\right) \tag{1}$$

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples
3. For each sample x compute class probabilities $p(y|x)$
4. Average them to approximately compute $p(y) \approx \mathop{\mathbb{E}}_{p_g(x)} [p(y|x)]$
5. Compute *inception score*:

$$\mathrm{IS} = \exp\left(\mathop{\mathbb{E}}_{p_g(x)} [D_{KL}(p(y|x)||p(y))] \right) \tag{1}$$

Intuition:

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples
3. For each sample x compute class probabilities $p(y|x)$
4. Average them to approximately compute $p(y) \approx \mathop{\mathbb{E}}_{p_g(x)} [p(y|x)]$
5. Compute *inception score*:

$$\mathrm{IS} = \exp\left(\mathop{\mathbb{E}}_{p_g(x)} [D_{KL}(p(y|x)||p(y))]\right) \tag{1}$$

Intuition:

- If samples represent meaningful objects then $p(y|x)$ is one-hot

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples
3. For each sample x compute class probabilities $p(y|x)$
4. Average them to approximately compute $p(y) \approx \mathop{\mathbb{E}}_{p_g(x)} [p(y|x)]$
5. Compute *inception score*:

$$\mathrm{IS} = \exp\left(\mathop{\mathbb{E}}_{p_g(x)} [D_{KL}(p(y|x) || p(y))] \right) \tag{1}$$

Intuition:

- If samples represent meaningful objects then $p(y|x)$ is one-hot
- If samples are diverse then $p(y)$ is uniform

Inception Score (IS)

1. Take Inception classifier
2. Generate a lot of samples
3. For each sample x compute class probabilities $p(y|x)$
4. Average them to approximately compute $p(y) \approx \mathop{\mathbb{E}}_{p_g(x)} [p(y|x)]$
5. Compute *inception score*:

$$\mathrm{IS} = \exp\left(\mathop{\mathbb{E}}_{p_g(x)} [D_{KL}(p(y|x)||p(y))] \right) \tag{1}$$

Intuition:

- If samples represent meaningful objects then $p(y|x)$ is one-hot
- If samples are diverse then $p(y)$ is uniform
- This gives very high values for KL between $p(y|x)$ and $p(y)$.

Frechet Inception Distance (FID)

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
    2.1 Compute embeddings for them

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings
3. Take a lot of real images and:

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings
3. Take a lot of real images and:
   3.1 Compute embeddings for them

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings
3. Take a lot of real images and:
   3.1 Compute embeddings for them
   3.2 Compute statistics μ_r, Σ_r for embeddings

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings
3. Take a lot of real images and:
   3.1 Compute embeddings for them
   3.2 Compute statistics μ_r, Σ_r for embeddings
4. Compute Frechet distance between two Guassians:

$$\mathrm{FID} = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right) \tag{2}$$

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings
3. Take a lot of real images and:
   3.1 Compute embeddings for them
   3.2 Compute statistics μ_r, Σ_r for embeddings
4. Compute Frechet distance between two Guassians:

$$\mathrm{FID} = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right) \tag{2}$$

Intuition:

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings
3. Take a lot of real images and:
   3.1 Compute embeddings for them
   3.2 Compute statistics μ_r, Σ_r for embeddings
4. Compute Frechet distance between two Guassians:

$$\mathrm{FID} = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right) \tag{2}$$

Intuition:

- We directly measure the distribution

Frechet Inception Distance (FID)

1. Take feature extractor from Inception classifier
2. Generate a lot of samples and:
   2.1 Compute embeddings for them
   2.2 Compute statistics μ_g, Σ_g for embeddings
3. Take a lot of real images and:
   3.1 Compute embeddings for them
   3.2 Compute statistics μ_r, Σ_r for embeddings
4. Compute Frechet distance between two Guassians:

$$\mathrm{FID} = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right) \tag{2}$$

Intuition:

- We directly measure the distribution
- It is not obvious why we do that with FD instead of KL/JS/etc

Used tricks

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.
- Double the width in each layer: +21% IS.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.
- Double the width in each layer: +21% IS.
- Pass class information via Conditional Batch Norm. Use shared embeddings and train a linear projection for each level.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.
- Double the width in each layer: +21% IS.
- Pass class information via Conditional Batch Norm. Use shared embeddings and train a linear projection for each level.
- Pass noise on each level (by concatenating to a class embedding): +18% faster. Use either independent noise per each level or shared.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.
- Double the width in each layer: +21% IS.
- Pass class information via Conditional Batch Norm. Use shared embeddings and train a linear projection for each level.
- Pass noise on each level (by concatenating to a class embedding): +18% faster. Use either independent noise per each level or shared.
- Use moving averages of Generator with decay of 0.9999.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.
- Double the width in each layer: +21% IS.
- Pass class information via Conditional Batch Norm. Use shared embeddings and train a linear projection for each level.
- Pass noise on each level (by concatenating to a class embedding): +18% faster. Use either independent noise per each level or shared.
- Use moving averages of Generator with decay of 0.9999.
- They used orthogonal initialization instead of He/Xavier: sample from standard normal, compute svd, take U matrix.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.
- Double the width in each layer: +21% IS.
- Pass class information via Conditional Batch Norm. Use shared embeddings and train a linear projection for each level.
- Pass noise on each level (by concatenating to a class embedding): +18% faster. Use either independent noise per each level or shared.
- Use moving averages of Generator with decay of 0.9999.
- They used orthogonal initialization instead of He/Xavier: sample from standard normal, compute svd, take U matrix.
- Progressive growing was unnecessary.

Used tricks

Baseline: SA-GAN (self-attention GAN) which uses spectral normalization for both D and G.

- Just increasing a batch size helps a lot (+46% IS). But this makes training unstable for some reason.
- Double the width in each layer: +21% IS.
- Pass class information via Conditional Batch Norm. Use shared embeddings and train a linear projection for each level.
- Pass noise on each level (by concatenating to a class embedding): +18% faster. Use either independent noise per each level or shared.
- Use moving averages of Generator with decay of 0.9999.
- They used orthogonal initialization instead of He/Xavier: sample from standard normal, compute svd, take U matrix.
- Progressive growing was unnecessary.
- A model is trained on 128 to 512 TPU v3 cores

Truncation trick

Truncation trick

Truncation trick

- Idea: at test-time sample from truncated normal instead of $\mathcal{N}(0, 1)$, i.e. resample large coordinates.

Truncation trick

- Idea: at test-time sample from truncated normal instead of $\mathcal{N}(0, 1)$, i.e. resample large coordinates.
- This improves quality but decreases diversity.

Truncation trick

- Idea: at test-time sample from truncated normal instead of $\mathcal{N}(0, 1)$, i.e. resample large coordinates.
- This improves quality but decreases diversity.
- Some models are not amenable to truncation because of the distribution shift: during training we see z that comes from different distribution.

Truncation trick

- Idea: at test-time sample from truncated normal instead of $\mathcal{N}(0, 1)$, i.e. resample large coordinates.
- This improves quality but decreases diversity.
- Some models are not amenable to truncation because of the distribution shift: during training we see z that comes from different distribution.
- Use orthogonal regularization to alleviate this:

$$R_\beta(W) = \beta \left\| W^\top W \odot (1 - I) \right\|_F^2$$

How to track instability in Generator[2]

[2] Training collapse is when IS/FID deteriorates very rapidly in a few iterations

How to track instability in Generator[2]

- Authors were tracking first 3 singular values.

[2] Training collapse is when IS/FID deteriorates very rapidly in a few iterations

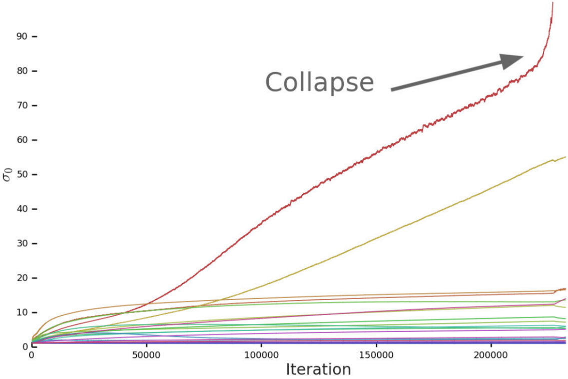
How to track instability in Generator[2]

- Authors were tracking first 3 singular values.
- They tried a lot of tricky ways to regularize them, but it worked only a little and couldn't guarantee stability.

[2] Training collapse is when IS/FID deteriorates very rapidly in a few iterations

How to track instability in Generator[2]

- Authors were tracking first 3 singular values.
- They tried a lot of tricky ways to regularize them, but it worked only a little and couldn't guarantee stability.
- This means that singular values explosion is a symptom and not the cause.



[2] Training collapse is when IS/FID deteriorates very rapidly in a few iterations

How to track instability in Discriminator

How to track instability in Discriminator

- Singular values are much more well-behaved, they only jump at collapse and do not explored

How to track instability in Discriminator

- Singular values are much more well-behaved, they only jump at collapse and do not explored
- Singluar values are more noisy and have spikes

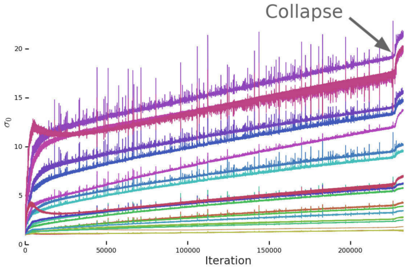
How to track instability in Discriminator

- Singular values are much more well-behaved, they only jump at collapse and do not explored
- Singluar values are more noisy and have spikes
- Authors hypothesize that spikes occur because D receives large gradients sometimes

How to track instability in Discriminator

- Singular values are much more well-behaved, they only jump at collapse and do not explored
- Singluar values are more noisy and have spikes
- Authors hypothesize that spikes occur because D receives large gradients sometimes
- To alleviate this they use R_1 regularization: it helps it decreases overall performance

$$R_1 := \frac{\gamma}{2}\mathbb{E}_{p_{\mathcal{D}}(x)}\left[\|\nabla D(x)\|_F^2\right] \tag{3}$$



Results

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	$7.5/152.8$	$7.7/181.4$	$11.5/241.5$	$39.7/298$