(**BigGAN**)
Large Scale GAN Training for High Fidelity Natural Image Synthesis

Andrew Brock, Jeff Donahue, Karen Simonyan

DeepMind

https://arxiv.org/abs/1809.11096

VIKRAM VOLETI | PhD, Mila

Presented at Mila, University of Montreal

October 30th, 2018

Contents

1. Key points

Figure 1: Class-conditional samples generated by our model.

- **Inception score** (128x128) **166.3** from 52.52, **FID 9.6** from 18.65

  previously held by Self-Attention GAN (SAGAN) ([Zhang et al., 2018](#))

1. Key points

Main contributions:

- We demonstrate that GANs benefit dramatically from scaling, and train models with two to four times as many parameters and eight times the batch size compared to prior art.
- We introduce two simple, general architectural changes that improve scalability (**shared conditional embeddings with linear projection**, **hierarchical latent space**), and modify a regularization scheme (**Orthogonal Regularization**) to improve conditioning, demonstrably boosting performance
- As a side effect of our modifications, our models become amenable to the “**truncation trick**,” a simple sampling technique that allows explicit, fine-grained control of the tradeoff between sample variety and fidelity.
- We discover instabilities specific to large scale GANs, and characterize them empirically. Leveraging insights from this analysis, we demonstrate that a combination of novel and existing techniques can reduce these instabilities, but complete training stability can only be achieved at a dramatic cost to performance.

1. Key points

	Prev. 128x128	128x128	256x256	512x512
Inception Score (higher is better)	52.52	166.3	233	241.4
Frechet Inception Distance (FID) (lower is better)	18.5	9.6	9.3	10.9

Inception Score: "Improved techniques for training gans" - Salimans et al.; NIPS 2016 (Salimans et al., 2016)

FID: "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium"; NIPS 2017 (Heusel et al., 2017)

Good summary of IS and FID: Medium

1. Key points

My takeaways:

- Possibly next SOTA
- Brief review of current best practices in GANs/adversarial learning
- Brief review of (latest?) key concepts in GAN training

Not covered in this presentation:

(Section 4 in the paper) ANALYSIS:

- Characterizing instability: the Generator
- Characterizing instability: the Discriminator

2. Scaling up GANs

a) Incremental changes
b) Innovations

2. Scaling up GANs

- **Incremental changes**
- Innovations

2. Scaling Up GANs - Incremental changes

1) Use **Self-Attention GAN (SAGAN)** as a baseline (Zhang et al., 2018)

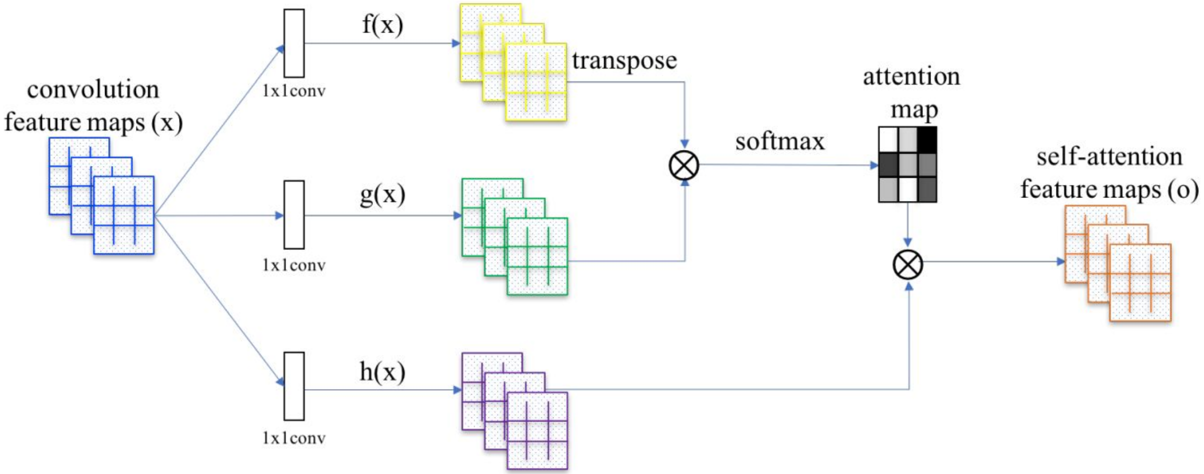


Figure 2: The proposed self-attention mechanism. The ⊗ denotes matrix multiplication. The softmax operation is performed on each row.

2. Scaling Up GANs - Incremental changes

1) Use **Self-Attention GAN (SAGAN)** as a baseline (Zhang et al., 2018)

Techniques to stabilize GAN training: (in the SAGAN paper)

a. **Spectral Normalization** (Miyato et al., 2018) for **both** Generator and Discriminator

$\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W)$, where σ(W) is the largest singular value of W

b. Imbalanced learning rate for generator and discriminator updates (**TTUR**) (Heusel et al., 2018)

https://github.com/bioinf-jku/TTUR

* In the paper, TTUR is not specifically mentioned, but they used different learning rates for G and D.

2. Scaling Up GANs - Incremental changes

2) Use **hinge loss** GAN objective (Geometric GAN: Lim & Ye, 2017; Tran et al., 2017)

Original GAN objective:

$$\min_G \max_D \mathbb{E}_{x \sim q_{\mathrm{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Hinge loss GAN objective: *(from the SAGAN paper)*

"the proposed attention module has been applied to both generator and discriminator, which are trained in an **alternating** fashion by **minimizing** the hinge version of the adversarial loss":

$$\begin{aligned} L_D = & -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0, -1 - D(G(z),y))], \\ L_G = & -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z),y), \end{aligned}$$

2. Scaling Up GANs - Incremental changes

3) Provide class information to **G** with **class-Conditional BatchNorm** (Dumoulin et al., 2017; de Vries et al., 2017)

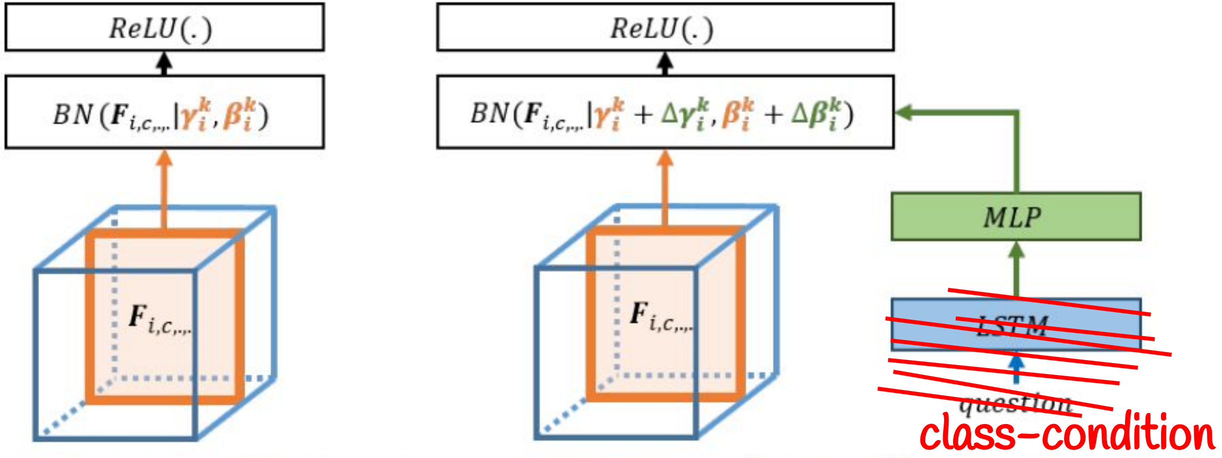


Figure 2: An overview of the computation graph of batch normalization (left) and conditional batch normalization (right). Best viewed in color.

2. Scaling Up GANs - Incremental changes

4) Provide class information to **D** with **projection** (Miyato & Koyama, 2018)

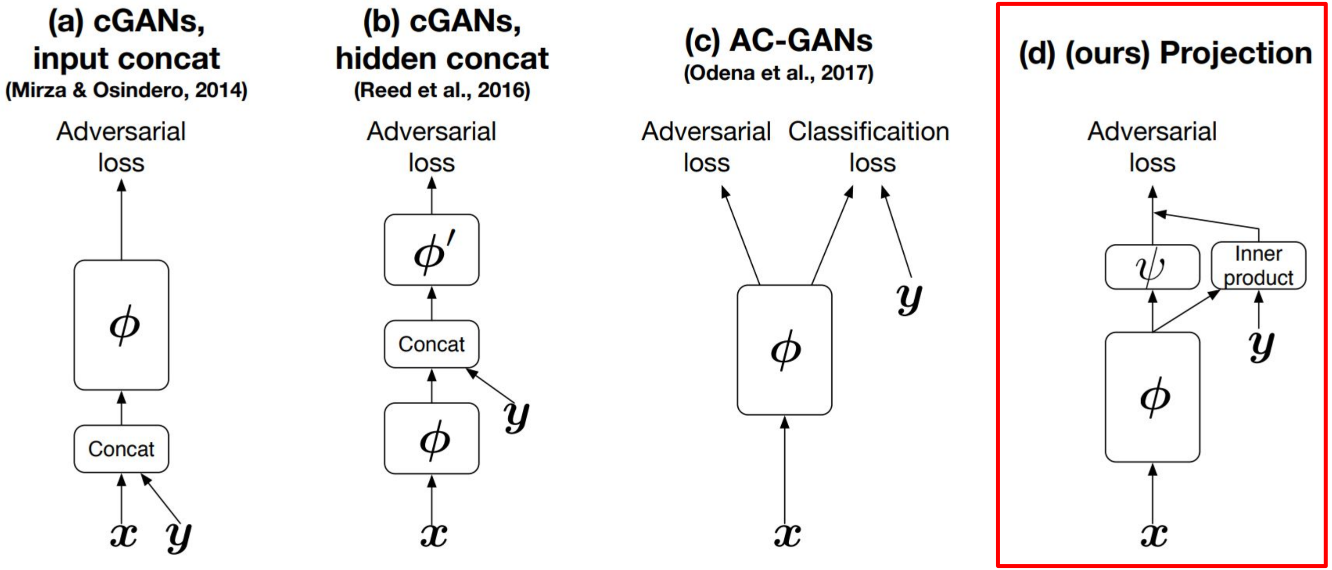


Figure 1: Discriminator models for conditional GANs

2. Scaling Up GANs - Incremental changes

5) Optimization: The optimization settings follow Zhang et al. (2018):

- Adam optimizer (Kingma & Ba, 2014), with a constant learning rate of $2 \cdot 10^{-4}$ in D and $5 \cdot 10^{-5}$ in G (whereas in SAGAN: 4.10^{-4} in D, 1.10^{-4} in G); in both networks, $\beta_1 = 0$ and $\beta_2 = 0.999$
- 2 D steps per G step (experimented with 1 to 6, found 2 to give best results)
- **Spectral Norm** (Miyato et al., 2018) in G and D: $\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W)$, where $\sigma(W)$ is the largest singular value of W

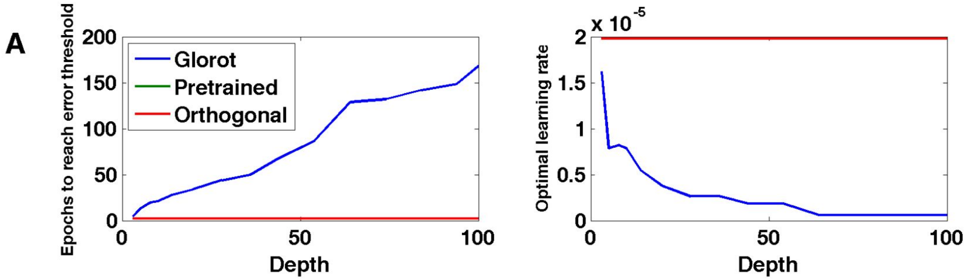
2. Scaling Up GANs - Incremental changes

6) Evaluation: exponential moving averages of G's weights following (ProgressiveGANS) Karras et al. (2018); Mescheder et al. (2018), with a decay of 0.9999.

- Karras et al. (2018): "...for visualizing generator output at any given point during the training, we use an exponential running average for the weights of the generator with decay 0.999."
- Mescheder et al: "...Similarly to prior work (Karras et al., 2017; Yazici et al., 2018; Gidel et al., 2018), we use an exponential moving average with decay 0.999 over the weights to produce the final model."

7) Initialization: Orthogonal Initialization (Saxe et al., 2014)

- “We empirically show that if we choose the **initial weights** in each layer to be a random **orthogonal matrix** (satisfying $W^TW = I$), instead of a scaled random Gaussian matrix, then this yields **depth independent learning times** just like greedy layerwise pre-training (indeed the red and green curves are indistinguishable).”



2. Scaling Up GANs - Incremental changes

8) “Each model is trained on 128 to 512 cores of a Google **TPU** v3 Pod (Google, 2018), and computes BatchNorm statistics in G across all devices, rather than per-device as in standard implementations”

2. Scaling Up GANs - Incremental changes

SUMMARY:

1) Baseline architecture: Use Self-Attention GAN (SAGAN) as a baseline (Zhang et al., 2018)
   a) Spectral Norm for both G and D
   b) TTUR
2) Loss: Use hinge loss GAN objective (Geometric GAN: Lim & Ye, 2017; Tran et al., 2017)
3) Provide class information to G with class-Conditional BatchNorm (de Vries et al., 2017)
4) Provide class information to D with projection (Miyato & Koyama, 2018)
5) Optimization: half the LRs than SAGAN, 2 D steps per G step, Spectral Norm in G and D
6) Evaluation: exponential moving averages of G's weights following Karras et al. (2018)
7) Initialization: Orthogonal Initialization (Saxe et al., 2014)
8) TPU, BatchNorm across all devices

2. Scaling up GANs

- Incremental changes
- **Innovations**

2. Scaling Up GANs - Innovations

Batch	Ch.	Param (M)	Shared	Hier.	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)

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Hier.* is using a hierarchical latent space, *Ortho.* is Orthogonal Regularization, and *Itr* either indicates that the setting is stable to 10^6 iterations, or that it collapses at the given iteration. Other than rows 1-4, results are computed across 8 different random initializations.

2. Scaling Up GANs - Innovations

1) Increase **batch size** to **8x**……………………………………..and nothing else! => **46%↑** in IS

Batch size
8x

46%↑

Batch	Ch.	Param (M)	Shared	Hier.	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)

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Hier.* is using a hierarchical latent space, *Ortho.* is Orthogonal Regularization, and *Itr* either indicates that the setting is stable to 10^6 iterations, or that it collapses at the given iteration. Other than rows 1-4, results are computed across 8 different random initializations.

2. Scaling Up GANs - Innovations

1) Increase **batch size** to **8x**..and nothing else! => **46%↑** in IS
2) Increase **width** (# of channels) in every layer by **50%**.......and nothing else! => further **21%↑** in IS
   (increasing depth degraded performance)

Batch size 8x

Width↑ 50%

46%↑

21%↑

Batch	Ch.	Param (M)	Shared	Hier.	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(±1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03(±1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85(±3.83)
2048	96	173.5	✗	✗	✗	295(±18)	9.54(±0.62)	92.98(±4.27)
2048	96	160.6	✓	✗	✗	185(±11)	9.18(±0.13)	94.94(±1.32)
2048	96	158.3	✓	✓	✗	152(±7)	8.73(±0.45)	98.76(±2.84)
2048	96	158.3	✓	✓	✓	165(±13)	8.51(±0.32)	99.31(±2.10)
2048	64	71.3	✓	✓	✓	371(±7)	10.48(±0.10)	86.90(±0.61)

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Hier.* is using a hierarchical latent space, *Ortho.* is Orthogonal Regularization, and *Itr* either indicates that the setting is stable to 10^6 iterations, or that it collapses at the given iteration. Other than rows 1-4, results are computed across 8 different random initializations.

2. Scaling Up GANs - Innovations

3) "...we opt to use a **shared embedding**, which is **linearly projected** to each layer's gains and biases."

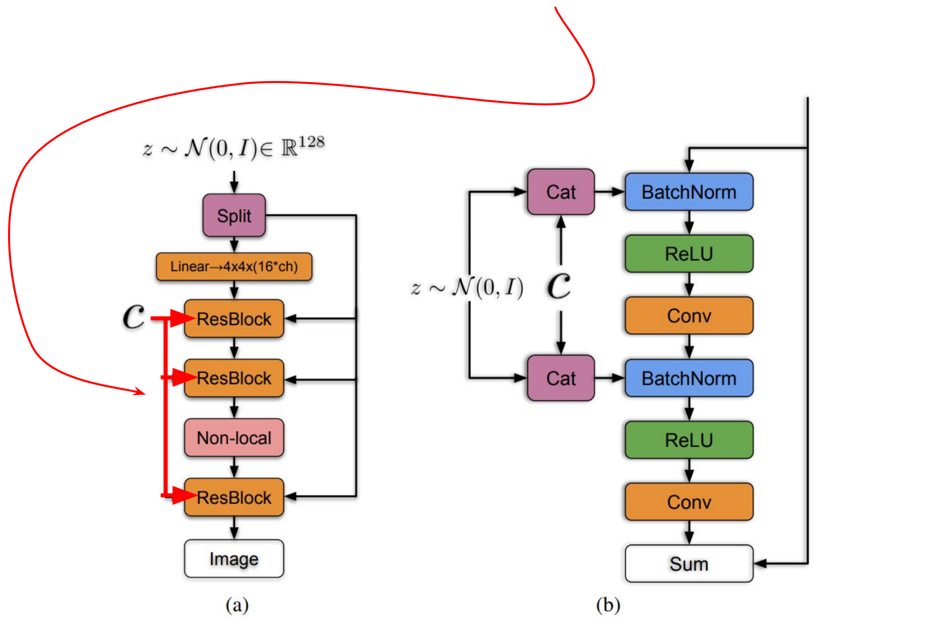
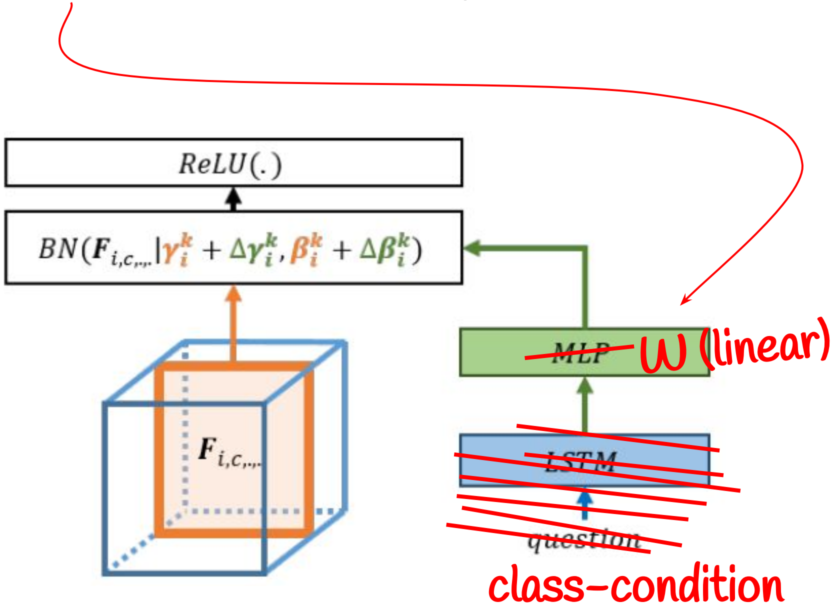


Figure 15: (a) A typical architectural layout for **G**; details are in the following tables. (b) A Residual Block in **G**. c is concatenated with a chunk of z and projected to the BatchNorm gains and biases.



2. Scaling Up GANs - Innovations

3) "...we opt to use a **shared embedding**, which is **linearly projected** to each layer's gains and biases."

"This reduces computation and memory costs, and improves training speed (in number of iterations required to reach a given performance) by 37%."

Shared embeddings with linear projection

Batch	Ch.	Param (M)	Shared	Hier.	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)

37%↑

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Hier.* is using a hierarchical latent space, *Ortho.* is Orthogonal Regularization, and *Itr* either indicates that the setting is stable to 10^6 iterations, or that it collapses at the given iteration. Other than rows 1-4, results are computed across 8 different random initializations.

2. Scaling Up GANs - Innovations

4) "...we employ a variant of **hierarchical** latent spaces, where the noise vector **z** is fed into **multiple layers of G** rather than just the initial layer."

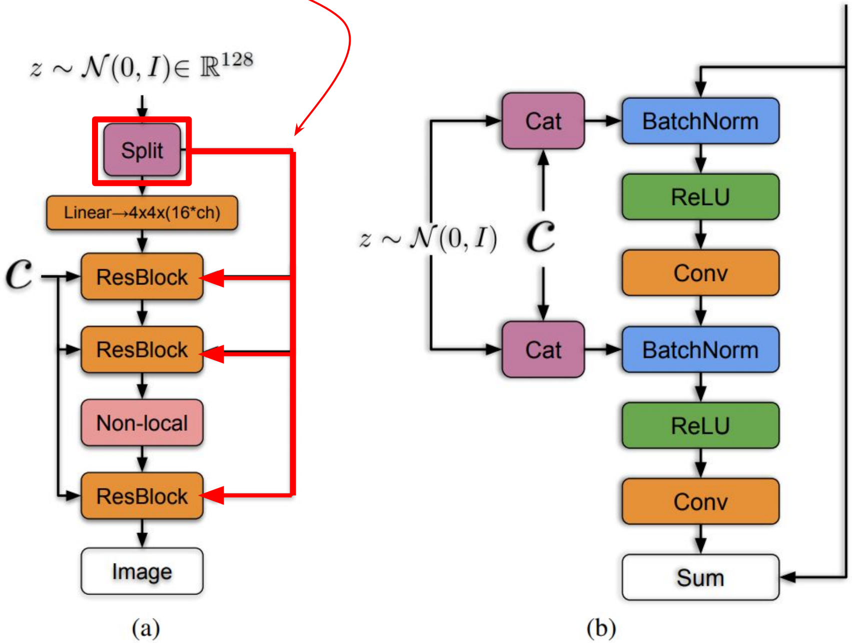


Figure 15: (a) A typical architectural layout for **G**; details are in the following tables. (b) A Residual Block in **G**. c is concatenated with a chunk of z and projected to the BatchNorm gains and biases.

2. Scaling Up GANs - Innovations

4) "...we employ a variant of **hierarchical** latent spaces"

"Hierarchical latents improve memory and compute costs (primarily by reducing the parametric budget of the first linear layer), provide a modest performance improvement of around 4%, and improve training speed by a further 18%."

Batch	Ch.	Param (M)	Shared	Hier.	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)

Hierarchical latent space

18%↑

4%↑

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Hier.* is using a hierarchical latent space, *Ortho.* is Orthogonal Regularization, and *Itr* either indicates that the setting is stable to 10^6 iterations, or that it collapses at the given iteration. Other than rows 1-4, results are computed across 8 different random initializations.

2. Scaling Up GANs - Innovations

5) “...**Truncation Trick**: truncating a z vector by resampling the values with magnitude above a chosen threshold”

- “...our best results come from using a different latent distribution for sampling than was used in training. Taking a model trained with z ~ N (0, I) and sampling z from a truncated normal immediately provides a boost to IS and FID.”
- “…leads to improvement in individual sample quality at the cost of reduction in overall sample variety.”

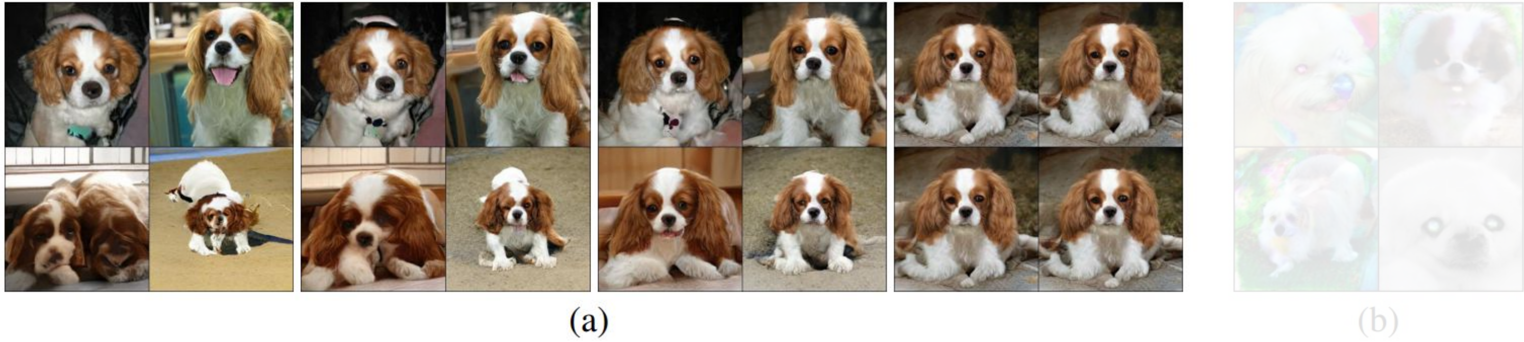


Figure 2: (a) The effects of increasing truncation. From left to right, threshold=2, 1.5, 1, 0.5, 0.04. (b) Saturation artifacts from applying truncation to a poorly conditioned model.

2. Scaling Up GANs - Innovations

5) "...**Truncation Trick**: truncating a z vector by resampling the values with magnitude above a chosen threshold"

- " As IS does not penalize lack of variety in class-conditional models, **reducing the truncation threshold** leads to a direct **increase in IS** (analogous to precision)"

- "FID penalizes lack of variety (analogous to recall) but also rewards precision, so we initially see a moderate improvement in FID, but as **truncation approaches zero** and variety diminishes, the **FID sharply drops**"

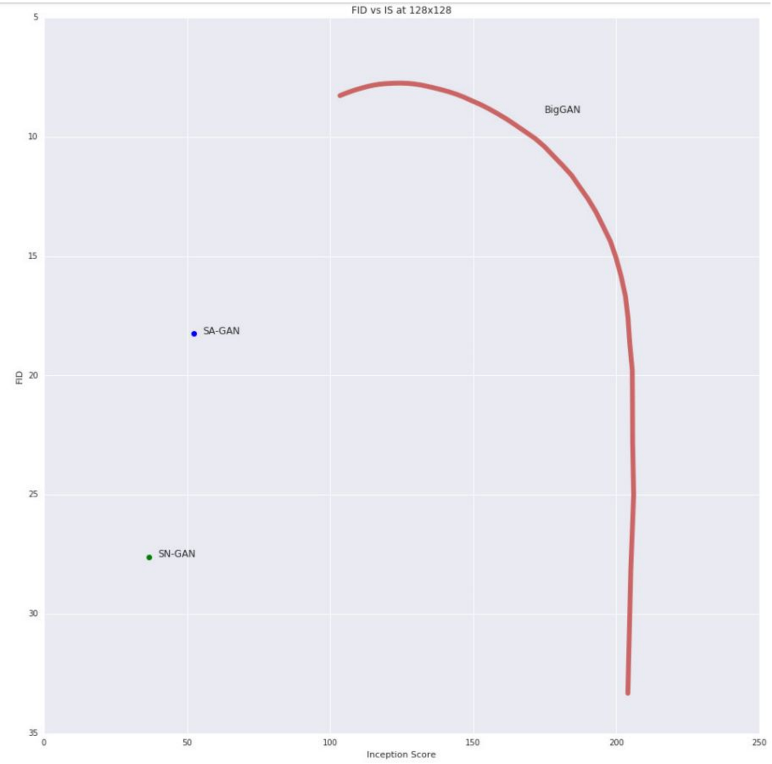
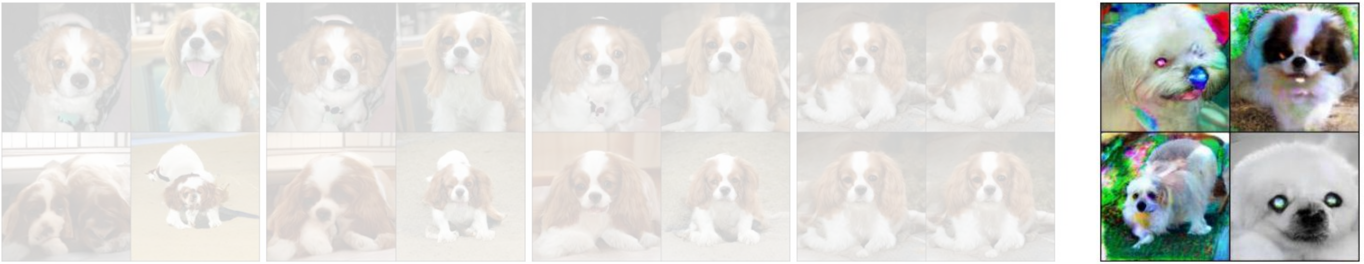


Figure 16: IS vs. FID at 128×128. Scores are averaged across three random seeds.

5) "...**Truncation Trick**: truncating a z vector by resampling the values with magnitude above a chosen threshold"

Problem!

- "... Some of our larger models are not amenable to truncation, producing saturation artifacts (Figure 2(b)) when fed truncated noise."



(a) (b)

Figure 2: (a) The effects of increasing truncation. From left to right, threshold=2, 1.5, 1, 0.5, 0.04. (b) Saturation artifacts from applying truncation to a poorly conditioned model.

Solution: Orthogonal Regularization

2. Scaling Up GANs - Innovations

6) **Orthogonal Regularization** (Brock et al., 2017)

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

Variant used in the paper:

$$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_\mathrm{F}^2, \tag{3}$$

where $\mathbf{1}$ denotes a matrix with all elements set to 1. We sweep β values and select 10^{-4}, find-

2. Scaling Up GANs - Innovations

6) **Orthogonal Regularization** (Brock et al., 2017)

- "...we observe that **without** Orthogonal Regularization, only **16%** of models are amenable to truncation, compared to **60%** when trained **with** Orthogonal Regularization."

Orthogonal Regularization

Batch	Ch.	Param (M)	Shared	Hier.	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)

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Hier.* is using a hierarchical latent space, *Ortho.* is Orthogonal Regularization, and *Itr* either indicates that the setting is stable to 10^6 iterations, or that it collapses at the given iteration. Other than rows 1-4, results are computed across 8 different random initializations.

2. Scaling Up GANs - Innovations

SUMMARY:

1) Increase batch size to 8x
2) Increase width (# of channels) by 50%
3) Shared embedding, linearly projected
4) Hierarchical latent space
5) Truncation trick
6) Orthogonal Regularization

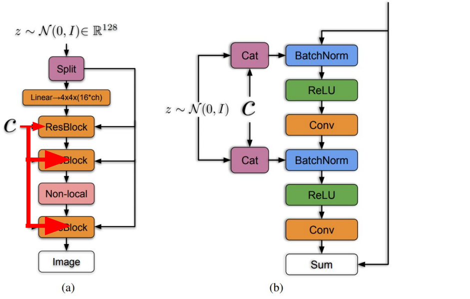
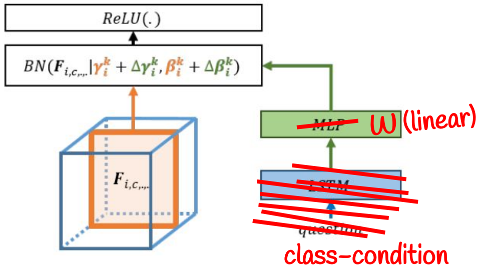
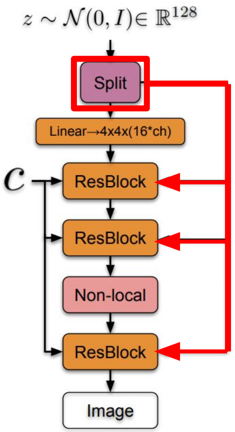


Figure 15: (a) A typical architectural layout for G; details are in the following tables. (b) A Residual Block in G. c is concatenated with a chunk of z and projected to the BatchNorm gains and biases.





$$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_F^2$$

2. Scaling Up GANs - Innovations

SUMMARY:

Batch size 8x

Width↑ 50%

Shared embeddings w/ linear projection

Hierarchical latent space

Orthogonal Regularization

Batch	Ch.	Param (M)	Shared	Hier.	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)

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Hier.* is using a hierarchical latent space, *Ortho.* is Orthogonal Regularization, and *Itr* either indicates that the setting is stable to 10^6 iterations, or that it collapses at the given iteration. Other than rows 1-4, results are computed across 8 different random initializations.

3. Cool examples

- 512x512
- Interpolations b/w c,z pairs
- Interpolations b/w c with z constant
- Weird examples from @memotv

3. Cool examples - 512x512

3. Cool examples - Interpolations b/w c,z pairs

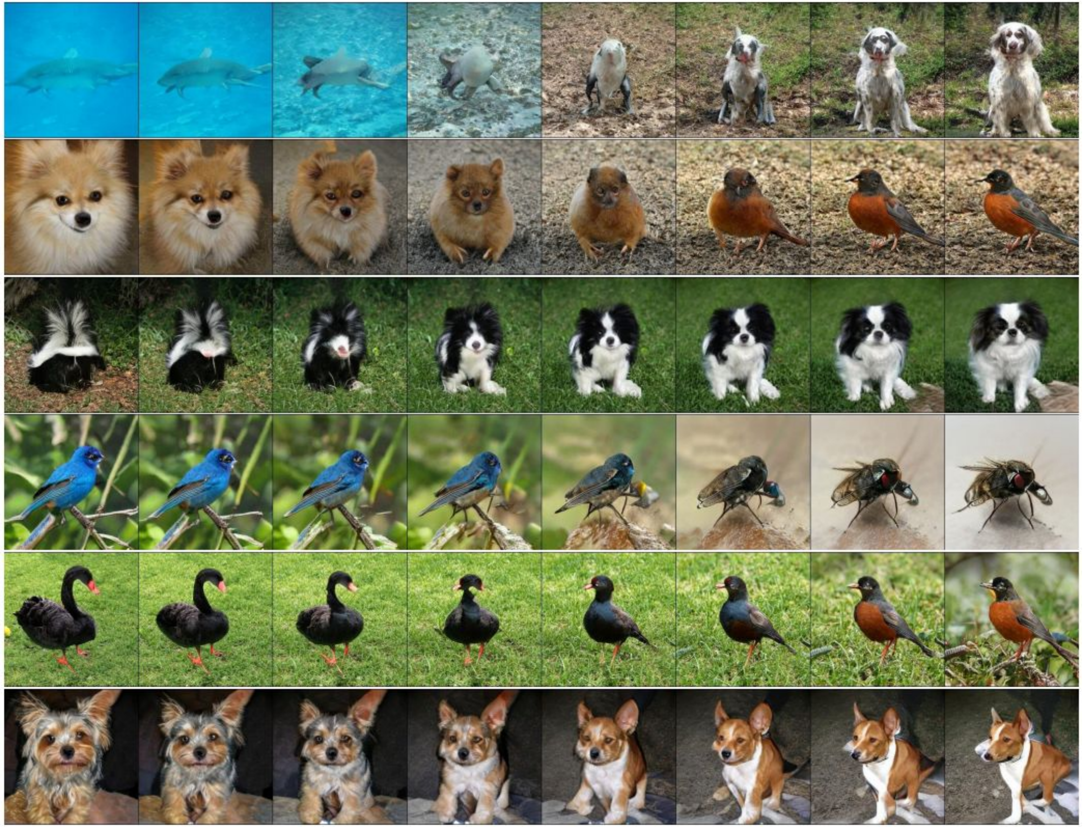


Figure 8: Interpolations between z, c pairs.

3. Cool examples - Interpolations b/w c with constant z

Figure 9: Interpolations between c with z held constant. Pose semantics are frequently maintained between endpoints (particularly in the final row). Row 2 demonstrates that grayscale is encoded in the joint z, c space, rather than in z.

3. Cool examples - Weird examples from @memotv








Thank you!