Large Scale GAN Training for High Fidelity Natural Image Synthesis

Andrew Brock, Jeff Donahue, Karen Simonyan
Heriot-Watt University and DeepMind

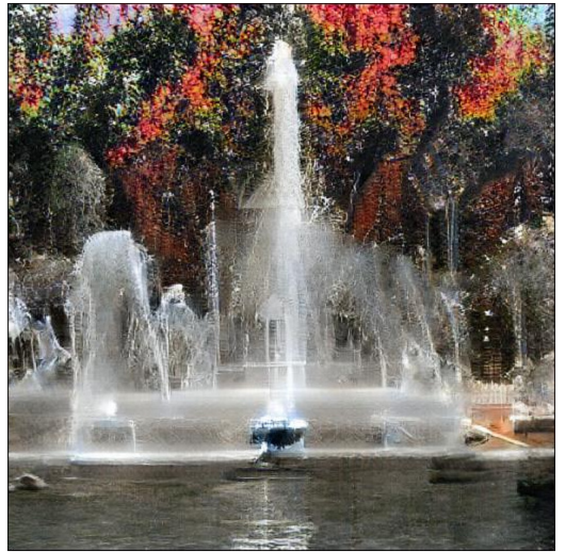
Presenter: Hongming Shan

Real v.s. Fake images

Fake or Real? (256x256)

Fake or Real? (512x512)





Contribution

- We demonstrate that GANs benefit dramatically from scaling, and train models **2~4x** as many parameters and **8x** the batch size compared to prior art. We introduce two simple, general architectural changes that improve scalability, and modify a regularization scheme 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.

Achievement

- When trained on ImageNet at 128x128 resolution, our models (BigGANs) improve the state-of-the-art Inception Score (IS) and Frechet Inception Distance (FID) from 52.52 and 18.65 to **166.3** and **9.6** respectively.

- We also successfully train BigGANs on ImageNet at 256x256 and 512x512 resolution, and achieve IS and FID of 233.0 and 9.3 at 256x256 and IS and FID of 241.4 and 10.9 at 512x512.

Generative Adversarial Network

- A GAN involves Generator (G) and Discriminator (D), whose purpose, respectively, is to map random noise to samples and discriminate real and generated samples

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

distribution $p(\boldsymbol{z})$ such as $\mathcal{N}(0, I)$ or $\mathcal{U}[-1, 1]$.

- Without auxiliary stabilization techniques, this training procedure is notoriously brittle, requiring finely-tuned hyperparameters and architectural choices to work at all.

Modification to vanilla GAN

- Counteract the use of unbounded loss functions and ensure D provides gradients everywhere to G .

  ✓ Changing the objective function to encourage convergence

  ✓ Constraining D through gradient penalties or normalization

- Choice of architecture,

  ✓ SA-GAN: adds the self-attention block to improve the ability of G and D to model global structure.

  ✓ ProGAN: Train high-resolution GANs in single-class setting by training a single model across a sequence of increasing resolutions.

Conditional GAN

- Class information can be fed into the model in various ways

✓ Class information is provided to G by concatenating a 1-hot class vector to the noise vector, an auxiliary classifier is to maximize class probability

✓ Class information is passed to G by supplying it with class-conditional gain and bias in BatchNorm (conditional BatchNorm)

✓ D is conditioned by using the cosine similarity between it features and a set of learned class embedding as additional evidence for distinguishing real and generated samples

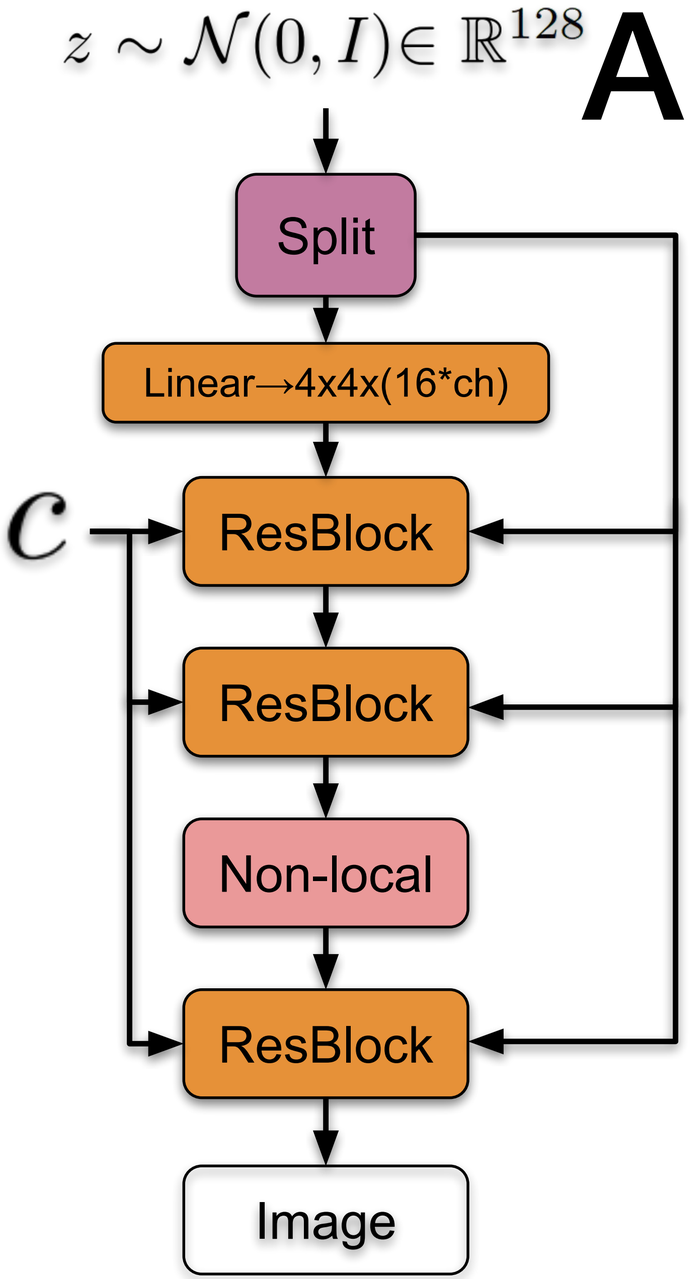
Evaluating implicit generative model

- Two popular scores based on Inception Network (pre-trained on Imagenet)
- **Inception score (IS) (higher is better)**
  - The quality of the generated images and diversity

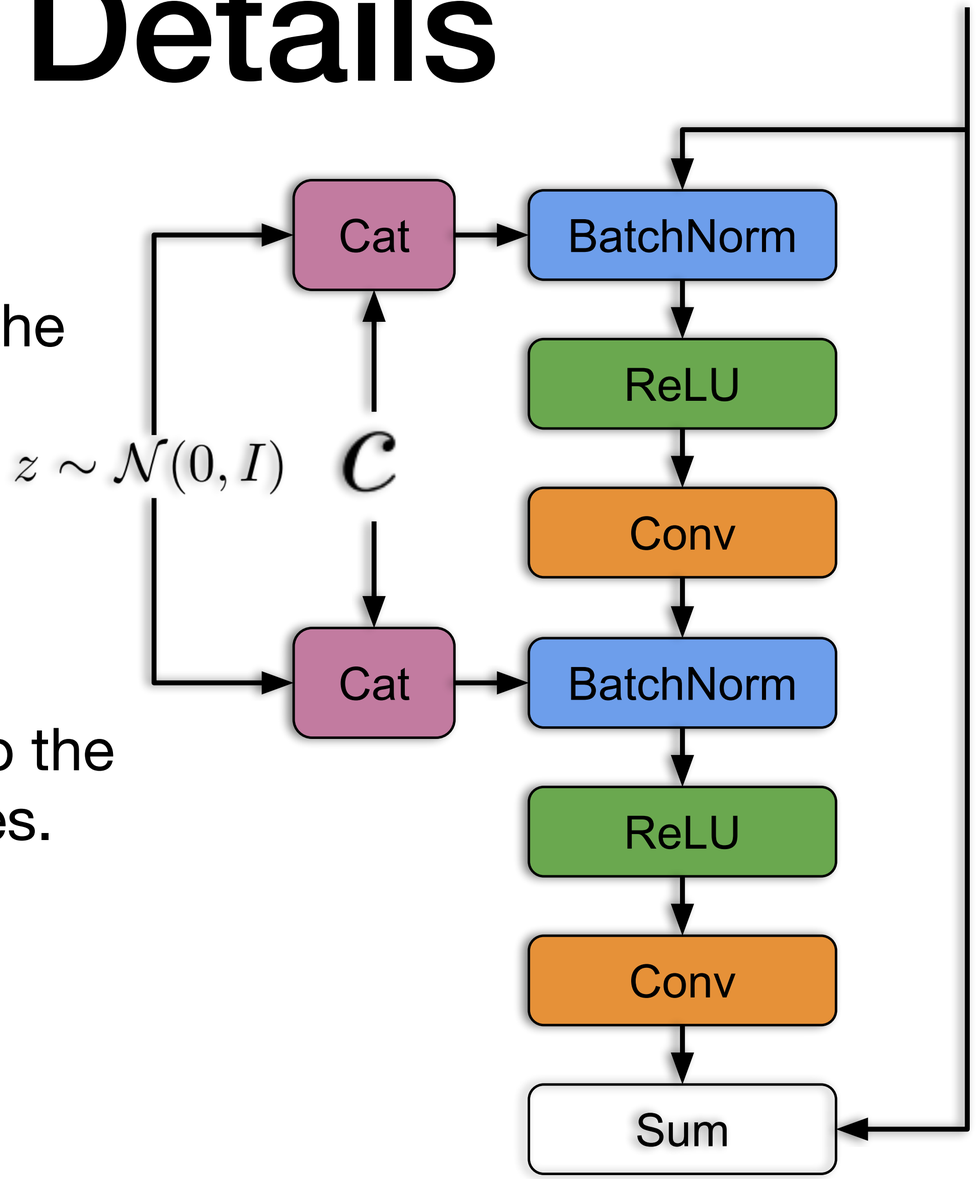
$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right),$$

- **Frechet inception distance (FID) (smaller is better)**
  - Model the data distribution for these features using a multivariate Gaussian distribution
  - $$\mathtt{FID}(x, g) = ||\mu_x - \mu_g||_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{\frac{1}{2}}),$$

Architectural Details



- (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.
- Different resolutions used different structures



128x128

Table 4: Architectures for ImageNet at 128×128 pixels. "ch" represents the channel width multiplier in each network from Table 1.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
dense, $4 \times 4 \times 16 \cdot ch$
ResBlock up $16 \cdot ch$
ResBlock up $8 \cdot ch$
ResBlock up $4 \cdot ch$
ResBlock up $2 \cdot ch$
Non-Local Block (64×64)
ResBlock up $1 \cdot ch$
BN, ReLU, 3×3 conv 3
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down $1 \cdot ch$
Non-Local Block (64×64)
ResBlock down $2 \cdot ch$
ResBlock down $4 \cdot ch$
ResBlock down $8 \cdot ch$
ResBlock down $16 \cdot ch$
ResBlock $16 \cdot ch$
ReLU
Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (dense $\rightarrow$ 1)

(b) Discriminator

Batch	Ch.	Param (M)	Shared	Hier.	Ortho.	Itr $\times 10^3$	FID	IS

- **Batch** is the batch size
- **Param** is the total number of parameters
- **Ch**. is the channel multiplier representing the number of unit in each layer
- **Shared** is using shared embeddings
- **Hier**. is using a hierarchical latent space
- **Ortho**. is Orthogonal regularization
- **Itr** either indicates the setting is stable to 10^6 or that it collapses at given iteration
- Other than rows 1-4, results are computed across 8 different random initializations.

Shared: Conditional BN

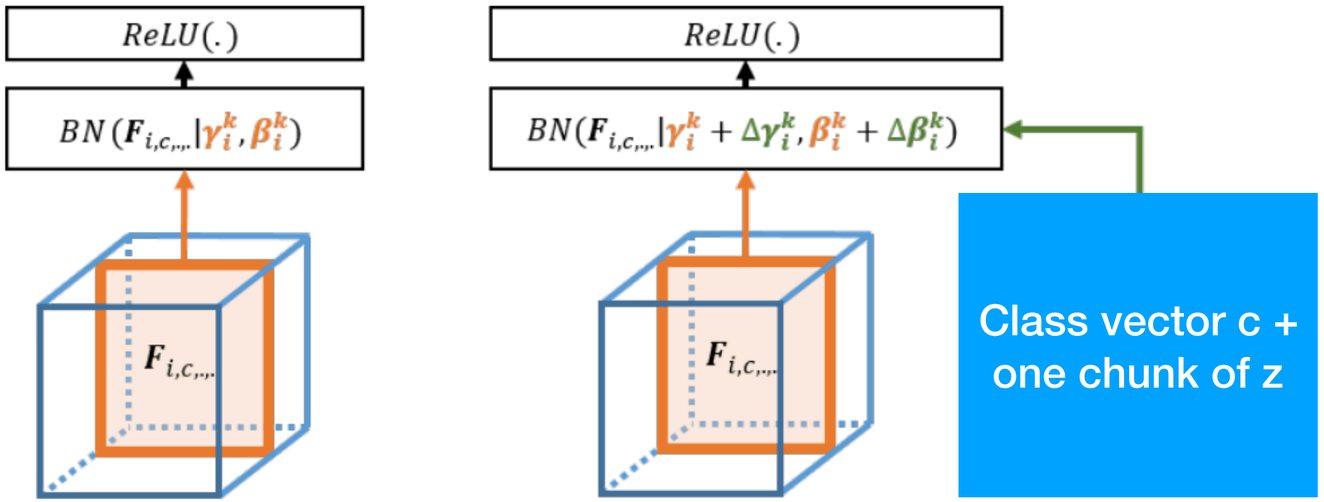
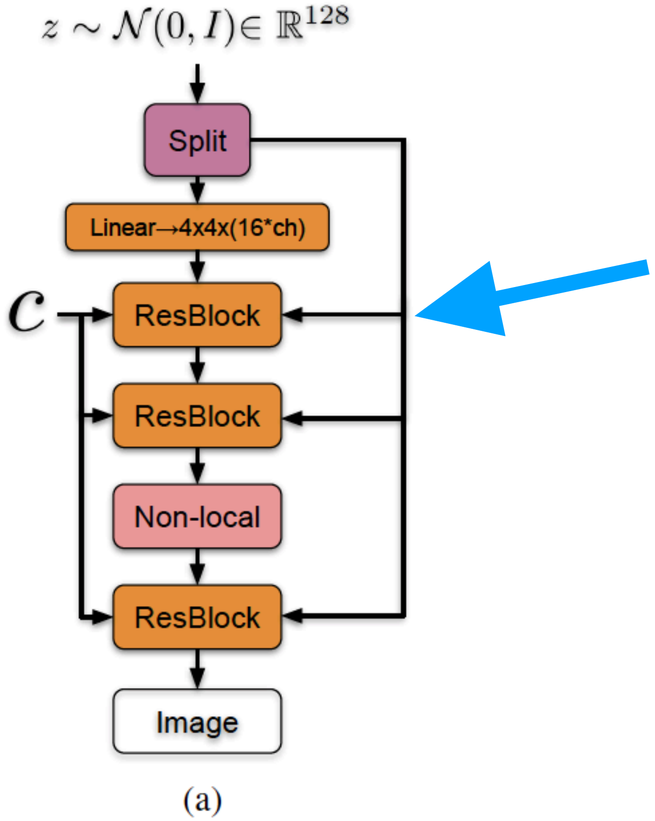


Figure 2: An overview of the computation graph of batch normalization (left) and conditional batch normalization (right). Best viewed in color.

- We use a single shared class embedding in G , which is linearly projected to produce per-sample gains and biases for the BatchNorm layers.

Hierarchical latent space

- When employing hierarchical latent spaces, the latent vector z is split along its channel dimension into equal sized chunks, and each chunk is separately concatenated to the copy of the class embedding passed into a given block



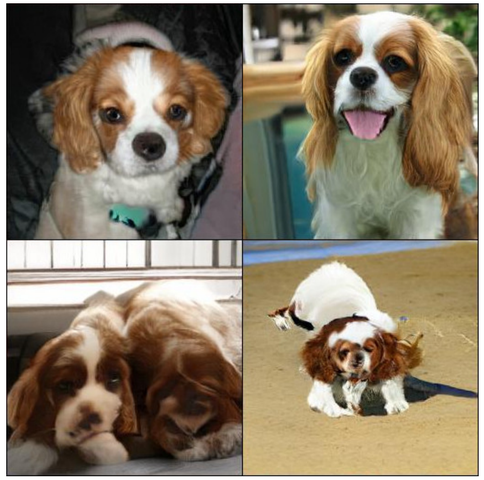
Scaling Up GANs

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)

Truncation trick: trading off variety and fidelity

- Remarkably, our best results come from using a different latent distribution for sampling than was used in training.

- Truncation trick: Taking a model trained with z~N(0; I) and sampling z from a truncated normal (where values which fall outside a range are resampled to fall inside that range) immediately provides a boost to IS and FID.

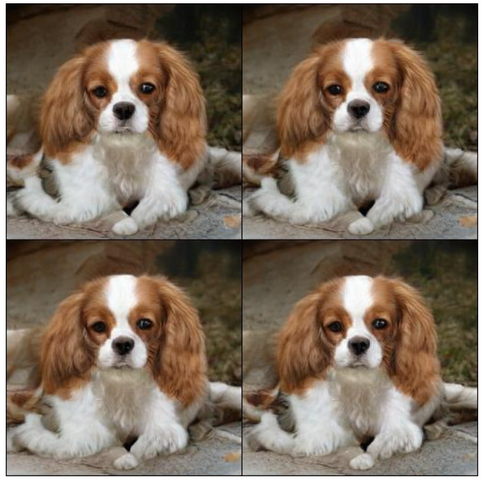


threshold=2

threshold=1.5

threshold=1

threshold=0.5



threshold=0.04

Orthogonal regularization

- Saturation artifacts from applying truncation to a poorly conditioned model

- Enforce amenability to truncation by conditioning G to be smooth, so that the full space of z will map to good output samples.

- Orthogonal Regularization.

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

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

Saturation artifacts

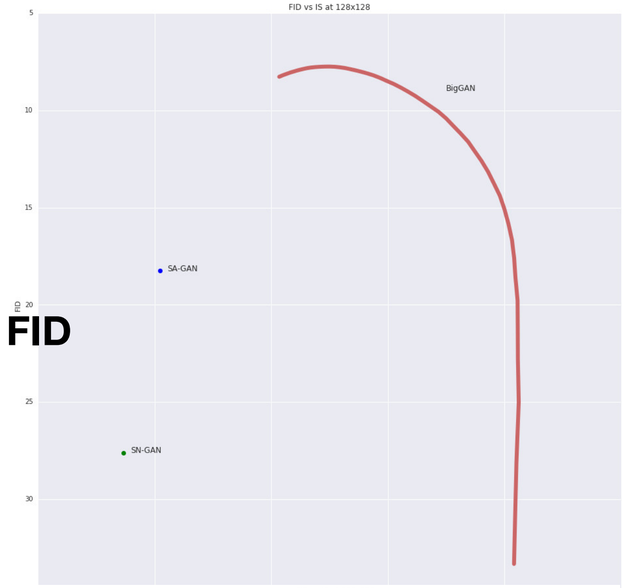
W is a weight matrix, cosine similarity between filters

Why truncation trick improves IS and FID?

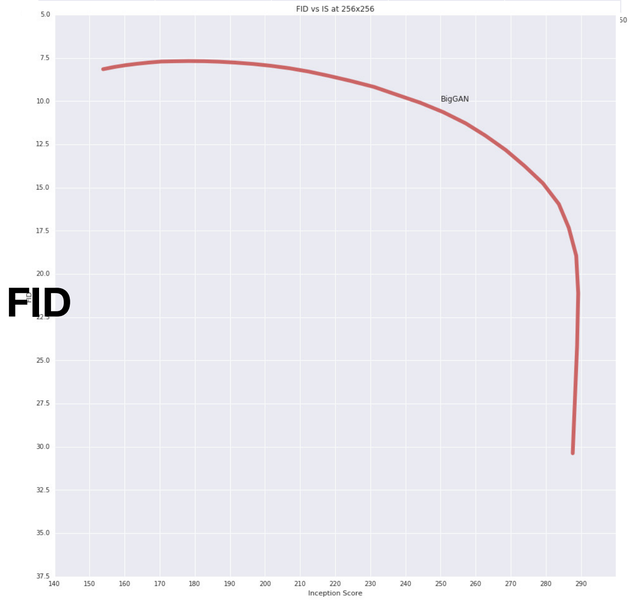
- This technique allows fine-grained, post-hoc selection of the trade-off between sample quality and variety for a given G

- **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.



128x128



256x256

IS

Method Summary

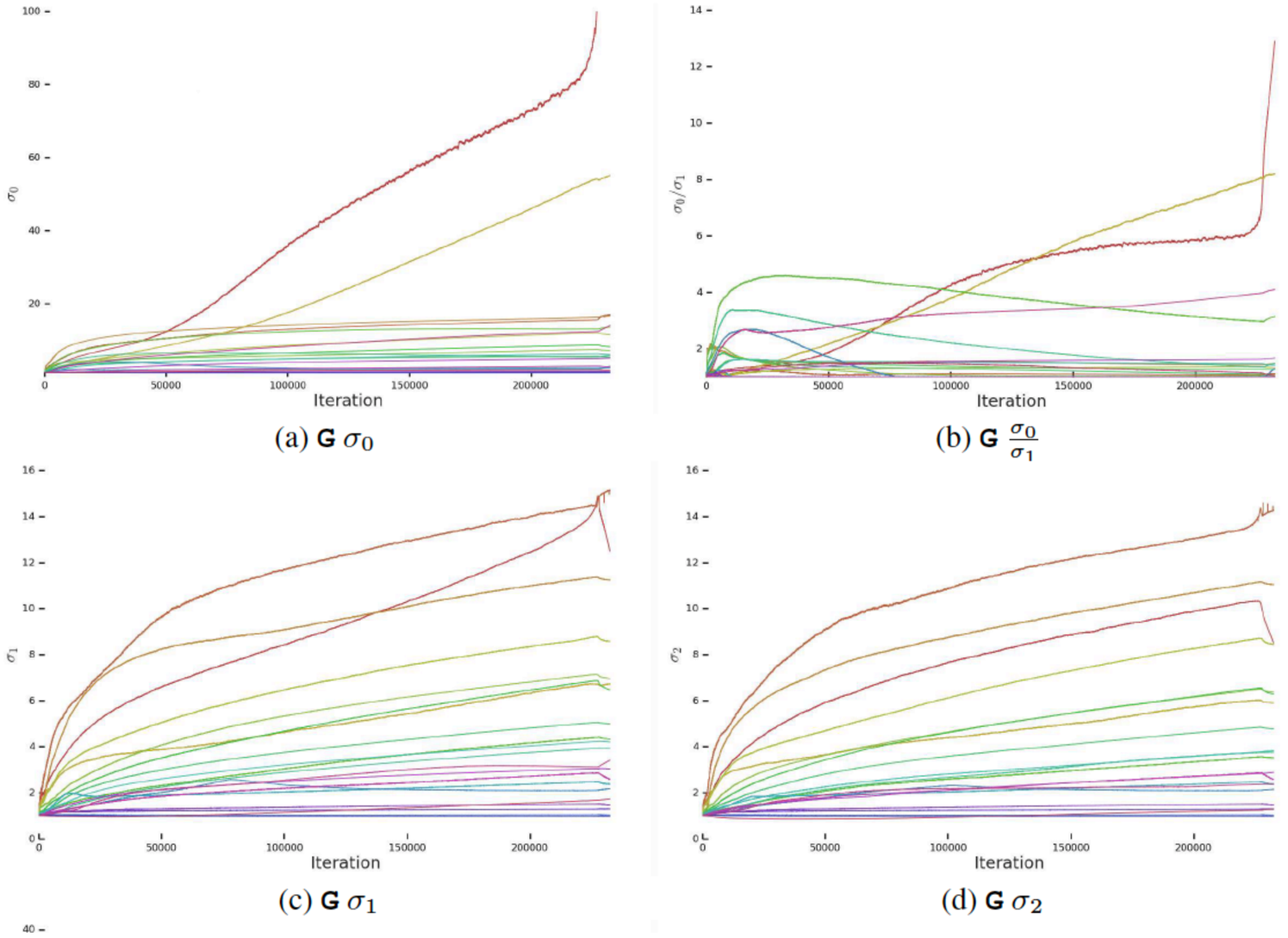
- We find that current GAN techniques are sufficient to enable scaling to large models and distributed, large-batch training.
- We find that we can dramatically improve the state of the art and train models up to 512x512 resolution without need for explicit multiscale methods like Karras et al. (2018).
- Despite these improvements, our models undergo training collapse, necessitating early stopping in practice.
- In the next two sections we investigate why settings which were stable in previous works become unstable when applied at scale.

Analysis of Instability

Characterizing Instability: Generator

- Top three singular values sigma_0, sigma_1, sigma_2 of each weight matrix to be the most informative.

- Most G layers have well-behaved spectral norms, but some layers (typically the first layer in G , which is over-complete and not convolutional) are ill-behaved, with spectral norms that grow throughout training and explode at collapse.

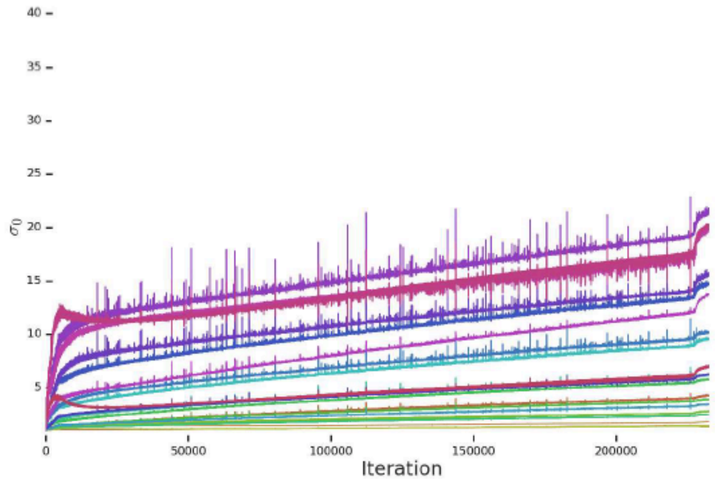


Collapse occurs after 200000 iterations.

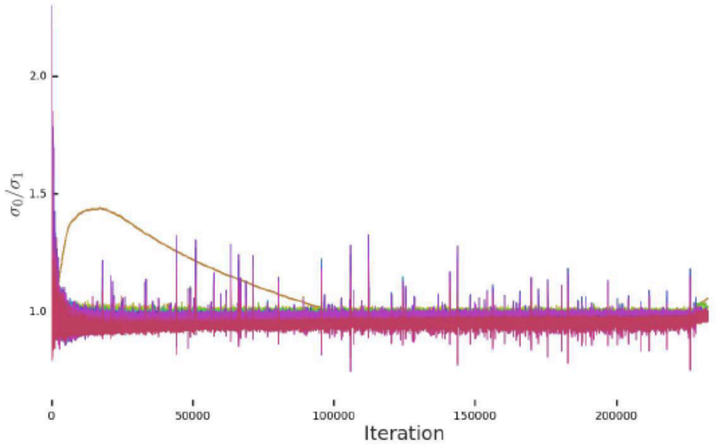
Improvement

- Directly regularize the top singular value sigma_0 of each value, (fixed value or some ratio of second singular value)
- Partial singular value decomposition to instead of clamp. $W = W - \max(0, \sigma_0 - \sigma_{clamp}) v_0 u_0^\top,$
- We observe that both with and without Spectral Normalization these techniques have the effect of preventing the gradual increase and explosion of either sigma_0 or sigma_0/sigma_1, but even though in some cases they mildly improve performance, no combination prevents training collapse.
- This evidence suggests that while conditioning G might improve stability, it is insufficient to ensure stability. We accordingly turn our attention to D .

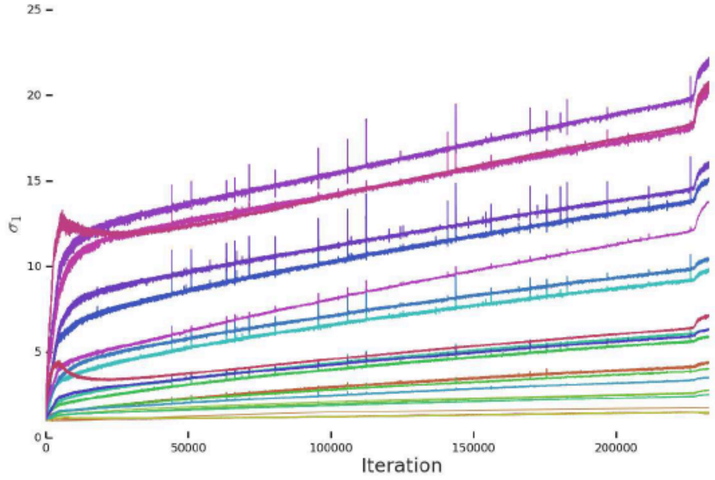
Characterizing Instability: Discriminator



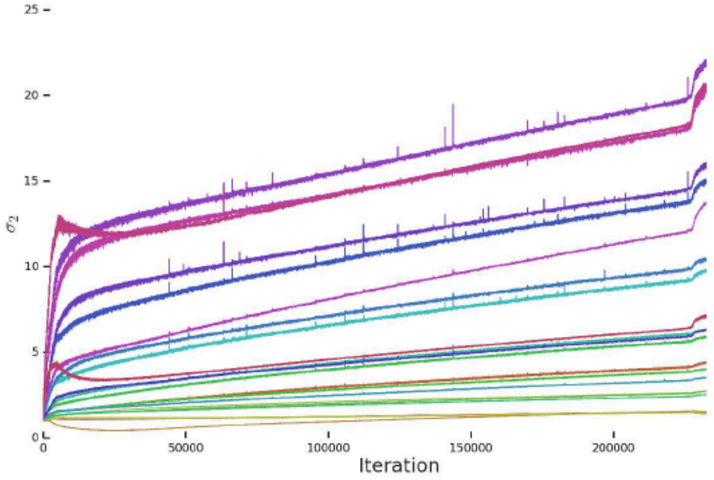
(e) **D** σ_0



(f) **D** $\frac{\sigma_0}{\sigma_1}$



(g) **D** σ_1



(h) **D** σ_2

- The spikes in D ’s spectra might suggest that it periodically receives very large gradients, but we observe that the Frobenius norms are smooth

- Posit that this noise is a result of optimization through the adversarial training process, where G periodically produces batches which strongly perturb D (**add penalty**). Stable and improve smoothness, but performance severely degrades (45%)

- We also observe that D ’s loss approaches zero during training, but undergoes a sharp upward jump at collapse. One possible explanation for this behavior is that D is overfitting to the training set, memorizing training examples rather than learning some meaningful boundary between real and generated images.

Instability Summary

- We find that stability does not come solely from G or D, but from their interaction through the adversarial training process.
- While the symptoms of their poor conditioning can be used to track and identify instability, ensuring reasonable conditioning proves necessary for training but insufficient to prevent eventual training collapse.
- It is possible to enforce stability by strongly constraining D , but doing so incurs a dramatic cost in performance.
- With current techniques, better final performance can be achieved by relaxing this conditioning and allowing collapse to occur at the later stages of training, by which time a model is sufficiently trained to achieve good results.

Results: Different resolution

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	$8.7 \pm .6/98.8 \pm 2.8$	$7.7 \pm .1/126.5 \pm .1$	$9.6 \pm .4/166.3 \pm 1$	$25 \pm 2/206 \pm 2$
BigGAN	256	$8.2 \pm .2/154 \pm 2.5$	$7.7 \pm .1/178 \pm 5$	$9.3 \pm .3/233 \pm 1$	$25 \pm 5/295 \pm 4$
BigGAN	512	10.9 / 154.9	9.3 / 202.5	10.9 / 241.4	24.4/275

Table 2: Evaluation of models at different resolutions. We report scores without truncation (Column 3), scores at the best FID (Column 4), scores at the IS of validation data (Column 5), and scores at the max IS (Column 6). Standard deviations are computed over at least three random initializations.

Samples and example of class leakage

(a) 128×128 (b) 256×256 (c) 512×512 (d)

Figure 4: Samples from our model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).

On JFT-300M at 256x256: different channel

Ch.	Param (M)	Shared	Hier.	Ortho.	FID	IS	(min FID) / IS	FID / (max IS)
64	317.1	✗	✗	✗	48.38	23.27	48.6 / 23.1	49.1 / 23.9
64	99.4	✓	✓	✓	23.48	24.78	22.4 / 21.0	60.9 / 35.8
96	207.9	✓	✓	✓	18.84	27.86	17.1 / 23.3	51.6 / 38.1
128	355.7	✓	✓	✓	13.75	30.61	13.0 / 28.0	46.2 / 47.8

Table 3: Results on JFT-300M at 256×256 resolution. The *FID* and *IS* columns report these scores given by the JFT-300M-trained Inception v2 classifier with noise distributed as $z \sim \mathcal{N}(0, I)$ (non-truncated). The *(min FID) / IS* and *FID / (max IS)* columns report scores at the best FID and IS from a sweep across truncated noise distributions ranging from $\sigma = 0$ to $\sigma = 2$. Images from the JFT-300M validation set have an IS of 50.88 and FID of 1.94.

Increase channels instead of depth…

Appendix…

(a) (b)

Figure 7: Comparing easy classes (a) with difficult classes (b) at 512×512. Classes such as dogs which are largely textural, and common in the dataset, are far easier to model than classes involving unaligned human faces or crowds. Such classes are more dynamic and structured, and often have details to which human observers are more sensitive. The difficulty of modeling global structure is further exacerbated when producing high-resolution images, even with non-local blocks.

Our model convincingly interpolates between disparate samples, and the nearest neighbors for its samples are visually distinct, suggesting that our model does not simply memorize training data.

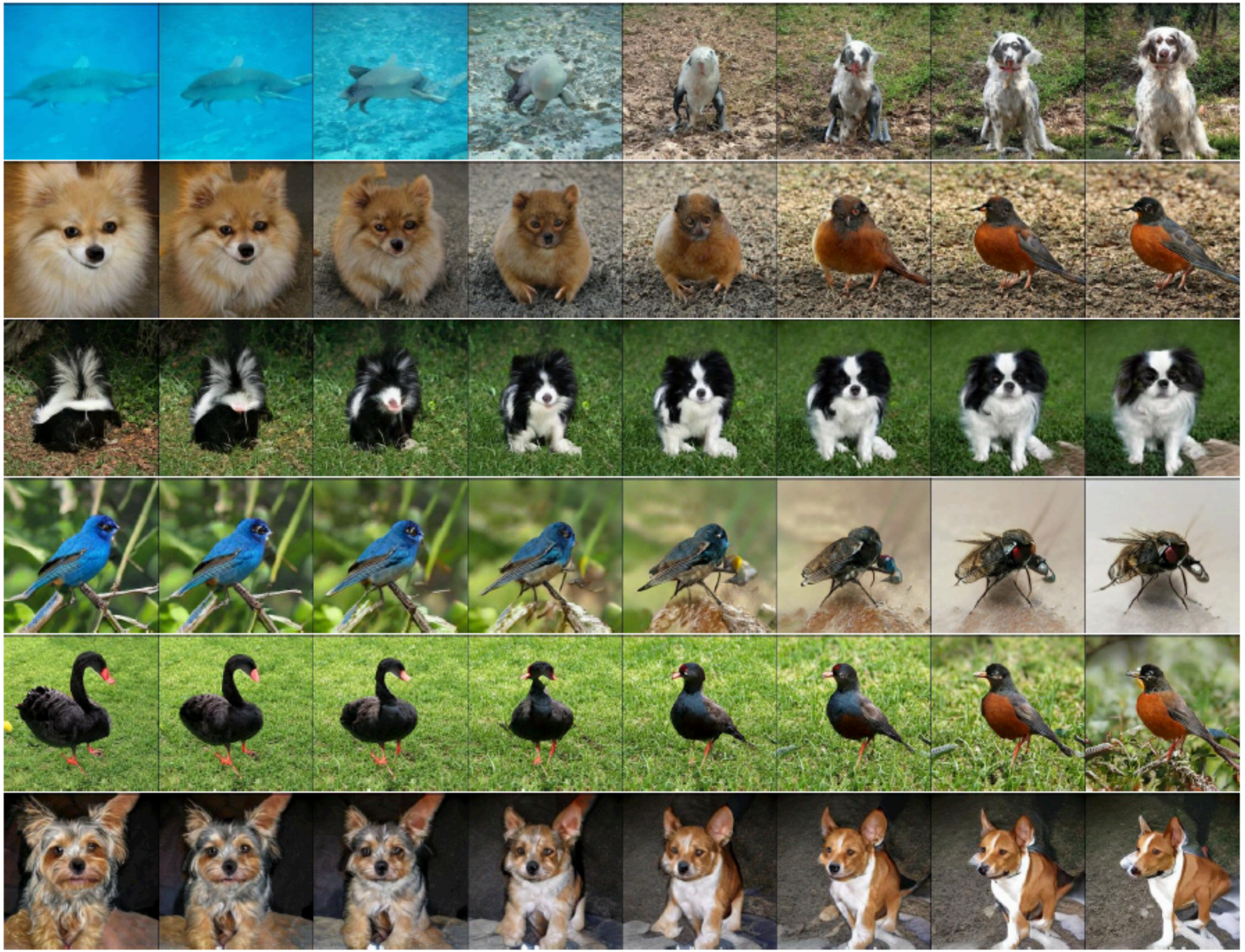


Figure 8: Interpolations between z, c pairs.

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.

Figure 10: Nearest neighbors in VGG-16-fc7 (Simonyan & Zisserman, 2015) feature space. The generated image is in the top left.

Figure 11: Nearest neighbors in ResNet-50-avgpool (He et al., 2016) feature space. The generated image is in the top left.

Figure 12: Nearest neighbors in pixel space. The generated image is in the top left.

Figure 13: Nearest neighbors in VGG-16-fc7 (Simonyan & Zisserman, 2015) feature space. The generated image is in the top left.

Figure 14: Nearest neighbors in ResNet-50-avgpool (He et al., 2016) feature space. The generated image is in the top left.

CHOOSING LATENT SPACES

LATENTS

- $\mathcal{N}(0, I)$. A standard choice of the latent space which we use in the main experiments.
- $\mathcal{U}[-1, 1]$. Another standard choice; we find that it performs similarly to $\mathcal{N}(0, I)$.
- Bernoulli $\{0, 1\}$. A discrete latent might reflect our prior that underlying factors of variation in natural images are not continuous, but discrete (one feature is present, another is not). This latent outperforms $\mathcal{N}(0, I)$ (in terms of IS) by 8% and requires 60% fewer iterations.
- $\max(\mathcal{N}(0, I), 0)$, also called Censored Normal. This latent is designed to introduce sparsity in the latent space (reflecting our prior that certain latent features are sometimes present and sometimes not), but also allow those latents to vary continuously, expressing different degrees of intensity for latents which are active. This latent outperforms $\mathcal{N}(0, I)$ (in terms of IS) by 15-20% and tends to require fewer iterations.
- Bernoulli $\{-1, 1\}$. This latent is designed to be discrete, but not sparse (as the network can learn to activate in response to negative inputs). This latent performs near-identically to $\mathcal{N}(0, I)$.
- Independent Categorical in $\{-1, 0, 1\}$, with equal probability. This distribution is chosen to be discrete and have sparsity, but also to allow latents to take on both positive and negative values. This latent performs near-identically to $\mathcal{N}(0, I)$.
- $\mathcal{N}(0, I)$ multiplied by Bernoulli $\{0, 1\}$. This distribution is chosen to have continuous latent factors which are also sparse (with a peak at zero), similar to Censored Normal but not constrained to be positive. This latent performs near-identically to $\mathcal{N}(0, I)$.
- Concatenating $\mathcal{N}(0, I)$ and Bernoulli $\{0, 1\}$, each taking half of the latent dimensions. This is inspired by Chen et al. (2016), and is chosen to allow some factors of variation to be discrete, while others are continuous. This latent outperforms $\mathcal{N}(0, I)$ by around 5%.
- Variance annealing: we sample from $\mathcal{N}(0, \sigma I)$, where σ is allowed to vary over training. We compared a variety of piecewise schedules and found that starting with $\sigma = 2$ and annealing towards $\sigma = 1$ over the course of training mildly improved performance. The space of possible variance schedules is large, and we did not explore it in depth – we suspect that a more principled or better-tuned schedule could more strongly impact performance.
- Per-sample variable variance: $\mathcal{N}(0, \sigma_i I)$, where $\sigma_i \sim \mathcal{U}[\sigma_l, \sigma_h]$ independently for each sample i in a batch, and (σ_l, σ_h) are hyperparameters. This distribution was chosen to try and improve amenability to the Truncation Trick by feeding the network noise samples with non-constant variance. This did not appear to affect performance, but we did not explore it in depth. One might also consider scheduling (σ_l, σ_h), similar to variance annealing.

NEGATIVE RESULTS

We explored a range of novel and existing techniques which ended up degrading or otherwise not affecting performance in our setting. We report them here; our evaluations for this section are not as thorough as those for the main architectural choices.

- We found that doubling the depth (by inserting an additional Residual block after every up- or down-sampling block) hampered performance.
- We experimented with sharing class embeddings between both **G** and **D** (as opposed to just within **G**). This is accomplished by replacing **D**'s class embedding with a projection from **G**'s embeddings, as is done in **G**'s BatchNorm layers. In our initial experiments this seemed to help and accelerate training, but we found this trick scaled poorly and was sensitive to optimization hyperparameters, particularly the choice of number of **D** steps per **G** step.
- We tried replacing BatchNorm in **G** with WeightNorm (Salimans & Kingma, 2016), but this crippled training. We also tried removing BatchNorm and only having Spectral Normalization, but this also crippled training.
- We tried adding BatchNorm to **D** (both class-conditional and unconditional) in addition to Spectral Normalization, but this crippled training.
- We tried varying the choice of location of the attention block in **G** and **D** (and inserting multiple attention blocks at different resolutions) but found that at 128×128 there was no noticeable benefit to doing so, and compute and memory costs increased substantially. We found a benefit to moving the attention block up one stage when moving to 256×256, which is in line with our expectations given the increased resolution.

NEGATIVE RESULTS

- We tried using filter sizes of 5 or 7 instead of 3 in either **G** or **D** or both. We found that having a filter size of 5 in **G** only provided a small improvement over the baseline but came at an unjustifiable compute cost. All other settings degraded performance.
- We tried varying the dilation for convolutional filters in both **G** and **D** at 128×128, but found that even a small amount of dilation in either network degraded performance.
- We tried bilinear upsampling in **G** in place of nearest-neighbors upsampling, but this degraded performance.
- In some of our models, we observed class-conditional mode collapse, where the model would only output one or two samples for a subset of classes but was still able to generate samples for all other classes. We noticed that the collapsed classes had embedings which had become very large relative to the other embeddings, and attempted to ameliorate this issue by applying weight decay to the shared embedding only. We found that small amounts of weight decay (10^{-6}) instead degraded performance, and that only even smaller values (10^{-8}) did not degrade performance, but these values were also too small to prevent the class vectors from exploding. Higher-resolution models appear to be more resilient to this problem, and none of our final models appear to suffer from this type of collapse.
- We experimented with using MLPs instead of linear projections from **G**'s class embeddings to its BatchNorm gains and biases, but did not find any benefit to doing so. We also experimented with Spectrally Normalizing these MLPs, and with providing these (and the linear projections) with a bias at their output, but did not notice any benefit.
- We tried gradient norm clipping (both the global variant typically used in recurrent networks, and a local version where the clipping value is determined on a per-parameter basis) but found this did not alleviate instability.

HyperParameters

APPENDIX H HYPERPARAMETERS

We performed various hyperparameter sweeps in this work:

- We swept the Cartesian product of the learning rates for each network through [10^{-5}, $5 \cdot 10^{-5}$, 10^{-4}, $2 \cdot 10^{-4}$, $4 \cdot 10^{-4}$, $8 \cdot 10^{-4}$, 10^{-3}], and initially found that the SA-GAN settings (**G**'s learning rate 10^{-4}, **D**'s learning rate $4 \cdot 10^{-4}$) were optimal at lower batch sizes; we did not repeat this sweep at higher batch sizes but did try halving and doubling the learning rate, arriving at the halved settings used for our experiments.
- We swept the R1 gradient penalty strength through [10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 2, 3, 5, 10]. We find that the strength of the penalty correlates negatively with performance, but that settings above 0.5 impart training stability.
- We swept the keep probabilities for DropOut in the final layer of **D** through [0.5, 0.6, 0.7, 0.8, 0.9, 0.95]. We find that DropOut has a similar stabilizing effect to R1 but also degrades performance.
- We swept **D**'s Adam β_1 parameter through [0.1, 0.2, 0.3, 0.4, 0.5] and found it to have a light regularization effect similar to DropOut, but not to significantly improve results. Higher β_1 terms in either network crippled training.
- We swept the strength of the modified Orthogonal Regularization penalty in **G** through [10^{-5}, $5 \cdot 10^{-5}$, 10^{-4}, $5 \cdot 10^{-4}$, 10^{-3}, 10^{-2}], and selected 10^{-4}.

Summary

- We have demonstrated that Generative Adversarial Networks trained to model natural images of multiple categories highly benefit from scaling up, both in terms of fidelity and variety of the generated samples.

- As a result, our models set a new level of performance among ImageNet GAN models, improving on the state of the art by a large margin.

- We have also presented an analysis of the training behavior of large scale GANs, characterized their stability in terms of the singular values of their weights, and discussed the interplay between stability and performance.