Small-GAN: Speeding up GAN Training using Core-Sets

Samarth Sinha [1] **Han Zhang** [2] **Anirudh Goyal** [3] **Yoshua Bengio** [3] **Hugo Larochelle** [2 3] **Augustus Odena** [2]

Abstract

Recent work by Brock et al. (2018) suggests that Generative Adversarial Networks (GANs) benefit disproportionately from large mini-batch sizes. Unfortunately, using large batches is slow and expensive on conventional hardware. Thus, it would be nice if we could generate batches that were *effectively large* though actually small. In this work, we propose a method to do this, inspired by the use of Coreset-selection in active learning. When training a GAN, we draw a large batch of samples from the prior and then compress that batch using Coreset-selection. To create effectively large batches of 'real' images, we create a cached dataset of Inception activations of each training image, randomly project them down to a smaller dimension, and then use Coreset-selection on those projected activations at training time. We conduct experiments showing that this technique substantially reduces training time and memory usage for modern GAN variants, that it reduces the fraction of dropped modes in a synthetic dataset, and that it allows GANs to reach a new state of the art in anomaly detection.

1. Introduction

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have become a popular research topic. Arguably the most impressive results have been in image synthesis (Brock et al., 2018; Salimans et al., 2018; Miyato et al., 2018; Zhang et al., 2018; 2017), but they have also been applied fruitfully to text generation (Fedus et al., 2018; Guo et al., 2018), domain transfer learning (Zhu et al., 2017; Zhang et al., 2017; Isola et al., 2017), and various other tasks (Xian et al., 2018; Ledig et al., 2017; Zhu & Bento, 2017).

[1]University of Toronto, Vector Institute [2]Google Brain [3]Mila, Universite de Montreal. Correspondence to: Samarth Sinha <samarth.sinha@mail.utoronto.ca>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

Recently, Brock et al. (2018) substantially improved the results of Zhang et al. (2018) by using very large mini-batches during training. The effect of large mini-batches in the context of deep learning is well-studied (Smith et al., 2017; Goyal et al., 2017; Keskar et al., 2016; Shallue et al., 2018) and general consensus is that they can be helpful in many circumstances, but the results of Brock et al. (2018) suggest that GANs benefit disproportionately from large batches. In fact, Table 1 of Brock et al. (2018) shows that for the Frechet Inception Distance (FID) metric (Heusel et al., 2017) on the ImageNet dataset, scores can be improved from 18.65 to 12.39 simply by making the batch eight times larger.

Unfortunately, increasing the batch size in this way is not always possible, since it increases the computational resources required to train these models – often beyond the reach of conventional hardware. The experiments from the BigGAN paper require a full 'TPU Pod'. The 'unofficial' open source release of BigGAN works around this by accumulating gradients across 8 different V100 GPUs and only performing a gradient update on the parameters, every 8 gradient accumulation steps. It's important to note that doing this type of gradient accumulation is (due to the parallel nature of modern computing hardware) **essentially 8 times slower than training on TPUs**. Thus, while gradient check-pointing will technically allow you to train these large models on conventional hardware, it's far from an ideal solution.

Future research on GANs would be much easier if we could have the gains from large batches without such pain points. In this paper, we take steps toward accomplishing that goal by proposing a technique that allows for *mimicking* large batches without the computational costs of actually using large batches.

In this work, we use Core-set selection (Agarwal et al., 2005) to sub-sample a large batch to produce a smaller batch. The large batches are then discarded, and the sub-sampled, smaller, batches are used to train the GAN. Informally, this procedure yields small batches with 'coverage' similar to that of the large batch – in particular the small batch tries to 'cover' all the same modes as are covered in the large batch. This technique yields many of the benefits of having large batches with much less computational

overhead. Moreover, it is generic, and so can be applied to nearly all GAN variants.

Our contributions can be summarized as follows:

- We introduce a simple, computationally cheap method to increase the 'effective batch size' of GANs, which can be applied to any GAN variant.
- We conduct experiments on the CIFAR, LSUN, and ImageNet data sets showing that our method can substantially improve FID across different GAN architectures given a fixed batch size.
- We use our method to improve the performance of the technique from Kumar et al. (2019), resulting in state-of-the-art performance at GAN-based anomaly detection.

2. Background and Notation

2.1. Generative Adversarial Networks

A Generative Adversarial Network (or GAN) is a system of two neural networks trained 'adversarially'. The generator, G, takes as input samples from a prior $z \sim p(z)$ and outputs the learned distribution, $G(z)$. The discriminator, D, receives as input both the training examples, X, and the synthesized samples, $G(z)$, and outputs a distribution $D(.)$ over the possible sample source. The discriminator is then trained to maximize the following objective:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))] \tag{1}$$

while the generator is trained to minimize[1]:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p(z)}[\log D(G(z))] \tag{2}$$

Informally, the generator is trained to *trick* the discriminator into believing that the generated samples $G(z)$ actually come from the target distribution, $p(x)$, while the discriminator is trained to be able to distinguish the samples from each other.

2.2. Frechet Inception Distance

We will refer frequently to the Frechet Inception Distance (FID) (Heusel et al., 2017), to measure the effectiveness of an image synthesis model. To compute this distance, one assumes that we have a pre-trained Inception classifier. One further assumes that the activations in the penultimate layer of this classifier come from a multivariate Gaussian. If the activations on the real data are $N(m, C)$ and the activations on the fake data are $N(m_w, C_w)$, the FID is defined as:

$$\|m - m_w\|_2^2 + \text{Tr}\big(C + C_w - 2\big(CC_w\big)^{1/2}\big) \tag{3}$$

Core-set selection: In computational geometry, a Core-set Q of a set P is a subset $Q \subset P$ that approximates the 'shape' of P (Agarwal et al., 2005). Core-sets are used to quickly generate approximate solutions to problems whose full solution on the original set would be burdensome to compute. Given such a problem[2], one computes Q, then computes the solution to the problem for Q and converts that into an approximate solution for the original set P. The general Core-set selection problem can be formulated several ways. Here we consider the the *minimax facility location* formulation (Farahani & Hekmatfar, 2009):

$$\min_{Q:|Q|=k} \max_{x_i\in P} \min_{x_j\in Q} d(x_i, x_j) \tag{4}$$

where k is the desired size of Q, and $d(.,.)$ is a metric on P. Informally, the formula above encodes the following objective: find some set Q of size k such that the maximum distance between a point in P and its nearest point in Q is minimized. Since finding the exact solution to the minimax facility location problem is NP-Hard (Wolsey & Nemhauser, 2014), we will have to make do with a greedy approximation, detailed in Section 3.3.

3. Using Core-set Sampling for GANs (or Small-GAN)

We aim to use Core-set sampling to increase the effective batch size during GAN training. This involves replacing the basic sampling operation that is done implicitly when minibatches are created. This implicit sampling operation happens in two places: First, when we create a minibatch of samples drawn from the prior distribution $p(z)$. Second, when we create a minibatch of samples from the target distribution $p_{\texttt{data}}(x)$. The first of these replacements is relatively simple, while the second presents challenges. In both cases, we have to work around the fact that actually doing Core-set sampling is computationally hard.

3.1. Sampling from the Prior

We need to sample from the prior when we update the discriminator and generator parameters. Our Core-set sampling algorithm doesn't take into account the geometry of the space we sample from, so sampling from a complicated

[1] This is the commonly used "Non-Saturating Cost". There are many others, but for brevity and since our technique we describe is agnostic to the loss function, we will omit them.

[2] As an example, consider computing the diameter of a point-set (Agarwal et al., 2005).

Algorithm 1 `GreedyCoreset`

Input: batch size (k), data points (x where $|x| > k$)
Output: subset of x of size k

$s \leftarrow \{\}$ ▷ Initialize the sampled set
while $|s| < k$ **do** ▷ Iteratively add points to sampled set
  $p \leftarrow \arg\max_{x_i \notin s} \min_{x_j \in s} d(x_i, x_j)$
  $s \leftarrow s \cup \{p\}$
end whilereturn s

density may cause trouble. This problem is not intractable, but it's nicer not to have to deal with it, so in the absence of any evidence that the choice of prior affects the training, we define the prior in our experiments to be a uniform distribution over a hypercube, but core-sets can be applied to different prior distributions, such as a Gaussian distribution (Bachem et al., 2017). To add Core-set sampling to the prior distribution, we randomly sample n points from the prior, where n is greater than the desired batch size, k. We then perform Core-set selection on the large batch of size n to create a batch of size k. By applying Core-set sampling on the randomly over-sampled prior, we obtain a small sparse batch that approximates the shape of the hypercube. The smaller batch is what's actually used to perform an SGD step.

3.2. Sampling from the Target Distribution

Sampling from the target distribution is more challenging. The elements drawn from the distribution are high dimensional images, so taking pairwise distances between them will tend to work poorly due to concentration of distances (Donoho et al., 2000; Sinha et al., 2019), and the fact that Euclidean distances are semantically meaningless in image space (Girod, 1993; Eskicioglu & Fisher, 1995).

To avoid these issues, we instead pre-process our data set by computing the 'Inception Embeddings' of each image using a pre-trained classifier (Szegedy et al., 2017). This is commonly done in the transfer-learning literature, where it is generally accepted that these embeddings have nontrivial semantics (Yosinski et al., 2014). Since this pre-processing happens only once at the beginning of training, it doesn't affect the per-training-step performance.

In order to further reduce the time taken by the Core-set selection procedure, and inspired by the Johnson-Lindenstrauss Lemma (Dasgupta & Gupta, 2003), we take random low dimensional projections of the Inception Embeddings. This means that we multiply the Inception Embeddings by a fixed random matrix. This yields low-dimensional embeddings where pairwise distances are *nearly* preserved (Dasgupta & Gupta, 2003). Combined with Core-set selection, this gives us representations of the training set images which are low-dimensional and for which pairwise Euclidean distances have meaningful semantics. We can then use Core-set sampling on those representations to select images at training time, analogous to how we select images from the prior.

3.3. Greedy Core-set Selection

In the above sections, we have invoked Core-set selection while glossing over the detail that exactly solving the k-center problem is NP-hard. This is important, because we propose to use Core-set selection at *every* training step[3]. Fortunately, we can make do with an approximate solution, which is faster to compute: we use the greedy k-center algorithm (similar to Sener & Savarese (2017)) summarized in Alg. 1.

3.4. Small-GAN

Our full proposed algorithm for GAN training is presented in Alg. 2. Our technique is agnostic to the underlying GAN framework and therefore can replace random sampling of mini-batches for all GAN variants. More implementation details and design choices are presented in Section 4.

4. Experiments

In this section we look at the performance of our proposed sampling method on various tasks: In the first experiment, we train a GAN on a Gaussian mixture dataset with a large number of modes and confirm our method substantially mitigates 'mode-dropping'. In the second, we apply our technique to GAN-based anomaly detection (Kumar et al., 2019) and significantly improve on prior results. Finally, we test our method on standard image synthesis benchmarks and confirm that our technique seriously reduces the need for large mini-batches in GAN training. The variety of settings in these experiments testifies to the generality of our proposed technique.

[3]Though the Core-set sampling does happens on CPU and so could be done in parallel to the GPU operations used to train the model, as long as the Core-set sampling time doesn't exceed the time of a forward and backward pass – which it doesn't.

Algorithm 2 `Small-GAN`

Input: target batch size (k), starting batch size ($n > k$), Inception embeddings (ϕ_I)
Output: a trained GAN
  Initialize networks G and D
  for $step = 1$ to ... **do**
    $z \sim p(z)$ ▷ Sample n points from the prior
    $x \sim p(x)$ ▷ Sample n points from the data distribution
    $\phi(x) \leftarrow \phi_I(x)$ ▷ Get cached embeddings for x
    $\hat{z} \leftarrow \texttt{GreedyCoreset}(z)$ ▷ Get Core-set of z
    $\widehat{\phi(x)} \leftarrow \texttt{GreedyCoreset}(\phi(x))$ ▷ Get Core-set of embeddings
    $\hat{x} \leftarrow \phi_I^{-1}(\widehat{\phi(x)})$ ▷ Get x corresponding to sampled embeddings
    Update GAN parameters as usual
  end for

4.1. Implementation Details

For our Core-set algorithm, the distance function, $d(\cdot, \cdot)$ is the Euclidean distance for both the prior and target distributions. Since we use the current state-of-the-art GAN architectures, each GAN model is trained with the same hyper-parameters and optimizers as was proposed in the corresponding paper, to ensure fair comparison. The only hyper-parameter altered is the batch-size, which is stated for all experiments. For over-sampling, we use a factor of 4 for the prior $p(z)$ and a factor of 8 for the target, $p(x)$, unless otherwise stated. We investigate the effects of different over-sampling factors in the ablation study in Section 5.3.

4.2. Mixture of Gaussians

We first investigate the problem of mode dropping (Arora et al., 2018) in GANs, where the GAN generator is unable to recover some modes from the target data set. We investigate the performance of training a GAN to recover a different number of modes of 2D isotropic Gaussian distributions, with a standard deviation of 0.05. We use a similar experimental setup as Azadi et al. (2018), where our generator and discriminator are parameterized using 4 ReLU-fully connected networks, and use the standard GAN loss in Eq. 1 and 2. To evaluate the performance of the models, we generate $10,000$ samples and assign them to their closest mode. As in Azadi et al. (2018), the metrics we use to evaluate performance are: i) 'high quality samples', which are samples within 4 standard deviations of the assigned mode and ii) 'recovered modes' which are mixture components with at least one assigned sample.

Our results are present in table 1, where we experiment with an increasing number of modes. We see that as the number of modes increases, a normal GAN suffers from increased mode dropping and lower sample quality compared to Core-set selection. With 100 modes, Core-set selection recovers 97.33% of the modes compared to 90.67% for the vanilla GAN. Core-set selection also generates 49.87% 'high quality' samples compared to 23.31% for the vanilla GAN.

4.3. Anomaly Detection

To see whether our method can be useful for more than just GANs, we also apply it to the Maximum Entropy Generator (MEG) from Kumar et al. (2019). MEG is an energy-based model whose training procedure requires maximizing the entropy of the samples generated from the model. Since MEG gives density estimates for arbitrary data points, it can be used for anomaly detection – a fundamental goal of machine learning research (Chandola et al., 2009; Kwon et al., 2017) – in which one aims to find samples that are 'atypical' given a source data set. Kumar et al. (2019) do use MEG successfully for this purpose, achieving results close to the state-of-the-art technique for GAN-based anomaly detection (Zenati et al., 2018). We hypothesized that – since energy estimates can in theory be improved by larger batch sizes – these results could be further improved by using Core-set selection, and we ran an experiment to confirm this hypothesis.

We follow the experimental set-up from Kumar et al. (2019) by training the MEG with all samples from a chosen MNIST digit left-out during training. Those samples then serve as the 'anomaly class' during evaluation. We report the area under the precision-recall curve and average the score over the last 10 epochs. The results are reported in Table 2, which provides clear evidence in favor of our above hypothesis: for all digits tested, adding Core-set selection to MEG substantially improves the results. By performing these experiments, we aim to show the general applicability of Core-set selection, not to suggest that MEG is superior to BiGANs (Zenati et al., 2018) on the task. We think it's likely that similar improvements could be achieved by using Core-set selection with BiGANs.

Number of Modes	% of Recovered Modes (GAN)	% of Recovered Modes (Ours)	% of High-Quality Samples (GAN)	% of High-Quality Samples (Ours)
25	**100**	**100**	95.76	**98.9**
36	**100**	**100**	92.73	**95.34**
49	98.12	**99.85**	84.28	**88.1**
64	96.13	**99.01**	68.81	**82.11**
81	92.59	**98.84**	49.74	**71.75**
100	90.67	**97.33**	23.31	**49.87**

Table 1. Experiments with large number of modes

Held-out Digit	Bi-GAN	MEG	Core-set+MEG
1	0.287	0.281	**0.351**
4	0.443	0.401	**0.501**
5	0.514	0.402	**0.518**
7	0.347	0.29	**0.387**
9	0.307	0.342	**0.39**

Table 2. Experiments with Anomaly Detection on MNIST dataset. The Held-out digit represents the digit that was held out of the training set during training and treated as the anomaly class. The numbers reported is the area under the precision-recall curve.

4.4. Image Synthesis

4.4.1. CIFAR AND LSUN

We also conduct experiments on standard image synthesis benchmarks. To further show the generality of our method, we experiment with two different GAN architectures and two image datasets. We use Spectral Normalization-GAN (Miyato et al., 2018) and Self Attention-GAN (Zhang et al., 2018) on the CIFAR (Krizhevsky et al., 2009) and LSUN (Yu et al., 2015) datasets, respectively. For the LSUN dataset, which consists of 10 different categories, we train the model using the 'outdoor church' subset of the data.

For evaluation, we measured the FID scores (Heusel et al., 2017) of $50,000$ generated samples from the trained models[4]. We compare the performance using SN-GANs with and without Core-set selection across progressively doubling batch sizes. We observe a similar effect to Brock et al. (2018): just by increasing the mini-batch size by a factor of 4, from 128 to 512, we are able to improve the FID scores from 18.75 to 15.68 for SN-GANs. This further demonstrates the importance of large mini-batches for GAN training.

Adding Core-set selection significantly improves the performance of the underlying GAN for all batch-sizes. For a batch size of 128, our model using Core-set sampling significantly outperforms the normal SN-GAN trained with a batch size of 256, and is comparable to an SN-GAN trained with a batch size of 512. The results suggest that the models perform significantly better for any given batch size when Coreset-sampling is used.

However, Core-set sampling does become less helpful as the underlying batch size increases: for SN-GAN, the performance improvement at a batch size of 128 is much larger than the improvement at a batch size of 512. This supports the hypothesis that Core-set selection works by approximating the coverage of a larger batch; a larger batch can already recover more modes of the data - so under this hypothesis, we would expect Core-set selection to help less for batches that are already large.

We see similar results when experimenting with Self Attention GANs (SAGAN) (Zhang et al., 2018) on the LSUN dataset (Yu et al., 2015). Compared to our results with SN-GAN, increasing the batch size results in a smaller difference in the performance for the SAGAN model, but we still see the FID improve from 14.82 to 12.63 as the batch-size is increased by a factor of 4. Using Core-set sampling with a batch size of 64, we are able to achieve a comparable score to when the model is trained with a batch size of 128.

We believe that one reason for a comparably smaller advantage of using Core-set sampling on LSUN is the nature of the data itself: using the 'outdoor church' subset of LSUN reduces the total number of 'modes' *possible* in the target distribution, since images of churches have fewer differences than the images in CIFAR-10 data set. We see similar effects in the mixture of Gaussians experiment (See 4.2) where the relative difference between a GAN trained with and without Core-set sampling increases as the number of

[4]Note that we measure the performance of all the models using the PyTorch version of FID scores, and not the official Tensorflow one. We ran all our experiments with the same code for accurate comparison.

GAN (batch-size = 128)	Small-GAN (batch-size = 128)	GAN (batch-size = 256)	Small-GAN (batch-size = 256)	GAN (batch-size = 512)	Small-GAN (batch-size = 512)
18.75 ± 0.2	$\mathbf{16.73 \pm 0.1}$	17.9 ± 0.1	$\mathbf{16.22 \pm 0.3}$	15.68 ± 0.2	$\mathbf{15.08 \pm 0.1}$

Table 3. FID scores for CIFAR using SN-GAN as the batch-size is progressively doubled. The FID score is calculated using $50,000$ generated samples from the generator.

Small-GAN (batch-size = 64)	GAN (batch-size = 64)	GAN (batch-size = 128)	GAN (batch-size = 256)
13.08	14.82	13.02	12.63

Table 4. FID scores for LSUN using SAGAN as the batch-size is progressively doubled. The FID score is calculated using $50,000$ generated samples from the generator. All experiments were run on the 'outdoor church' subset of the dataset.

modes are increased.

4.4.2. IMAGENET

Finally, in order to test that our method would work 'at scale', we ran an experiment on the ImageNet data set. Using the code at https://github.com/heykeetae/Self-Attention-GAN, we trained two GANs: The first is trained exactly as described in the open-source code. The second is trained using Coreset selection, with all other hyper-parameters unchanged. Simply adding Coreset selection to the existing SAGAN code materially improved the FID (which we compute using 50000 samples): the baseline model had an FID of 19.40 and the Core-set model had an FID of 17.33.

5. Further Analysis

5.1. Timing Analysis

Since random sampling can be done very quickly, it is important to investigate the amount of time it takes to train GANs with and without Core-set sampling. We measured the time for SN-GAN to do 50 gradient steps on the CIFAR dataset with various mini-batch sizes: the results are in Table 5. On average, for each gradient step, the time added by performing Core-Set sampling is only 0.024 seconds. Since we perform Core-set construction on the Inception embeddings, we do not incur an additional dataloading cost while training the models since a mini-batch of the same size is needed from the data-loader. Similarly, GPU memory also scales linearly with the mini-batch size, so using a mini-batch which is quarter in size will lead to quarter of the GPU memory being used for each gradient step.

5.2. Comparison to Gradient Check-pointing

It's worth being very explicit about the cost of gradient check-pointing – the other way of simulating large batches. If I use gradient check-pointing on batches of size 128 to simulate a batch of size 1024, that means we need to do a forward and backward pass, 8 times, in serial. If I use Core-set selection to turn a pseudo-batch of 1024 samples into a batch of 128, then, apart from the cost of actually doing the Core-set sampling (which we found in the last section to be negligible), it is 8 times faster, because the only forward and backward pass that happens happens only once, on a batch of size 128.

5.3. Ablation Study

We conduct an ablation study to investigate the reasons for the effectiveness of Core-set selection. We also investigate the effect of different sampling factors and other hyper-parameters. We run all ablation experiments on the task of image synthesis using SN-GAN (Miyato et al., 2018) with the CIFAR-10 dataset (Krizhevsky et al., 2009). We use the same hyperparameters as in our main image synthesis experiments and a batch size of 128, unless otherwise stated.

5.4. Examination of Main Hyper-Parameters

We examine i) the importance of the chosen target distribution for Core-set selection and ii) the importance of performing Core-set selection on that target distribution. The FID scores are reported in Table 6. The importance of the target distribution is clear, since performing Core-set selection directly on the images (experiment B) performs similar to random-sampling.

Experiment C supports our hypothesis that performing a random projection on the Inception embeddings can preserve semantic information while reducing the dimensionality of the features. This increases the effectiveness of Core-set sampling and reduces sampling time. Our ablation study also shows the importance of performing Core-set selection on both the prior and target distribution. The FID scores of the models are considerably worse when Core-set sampling is used on either distribution alone.

Small-GAN (batch size = 128)	SN-GAN (batch size = 128)	SN-GAN (batch size = 256)	SN-GAN (batch size = 512)
14.51s	13.31s	26.46s	51.64s

Table 5. Timing to perform 50 gradient updates for SN-GAN with and without Core-sets. The time is measured in seconds. All the experiments were performed on a single NVIDIA Titan-XP GPU.

Small-GAN	A	B	C	D	E
16.73	18.75	18.09	17.03	17.88	17.45

Table 6. FID scores for CIFAR using SN-GAN. The experiment list is: A = Training an SN-GAN, B = Core-set selection directly on the images, C = Core-set applied directly on Inception embeddings without a random projection, D = Core-set applied only on the prior distribution, E = Core-set applied only on target distribution.

5.5. Examination of Sampling Factors

Another important hyper-parameter for training GANs using Core-set selection is the sampling factor. In Table 7 we varied the factors by which both the prior and the target distributions were over-sampled. We see that using 4 for the sampling factor for the prior and 8 for the sampling factor for the target distribution results in the best performance. One interesting observation: that the performance eventually starts degrading as the sampling factor is increased. Since the greedy algorithm in Alg. 1 sequentially selects the point that is the furthest away from the the already sampled set, we believe that when the sampling factors are set too high, the algorithm becomes sensitive to outliers.

5.6. Potential Distorting Effects of Core-set Sampling

Theoretically, Core-set sampling of the training distribution can cause some distortion of the distribution learned by the generator. However, there are good ways to mitigate this, as we will discuss. Moreover, in cases where Core-set sampling causes more distortion, a baseline GAN will tend to distort things in the opposite direction.

To examine this phenomenon further, we trained GANs using 4 different algorithms on a mixture of two Gaussians with mixing coefficients of 0.01 and 0.99. We will refer to these as the 0.01-mode and the 0.99-mode in the following discussion. For all 4 experiments, we use a batch size of 10. The results are as follows:

5.6.1. Vanilla GAN

Training a 'vanilla' GAN on this mixture of two Gaussians results in the 0.01-mode being completely dropped. This is consistent with existing results on GANs. It's also important because it suggests that, even if Core-set sampling is unavoidably distorting (which it's not - see below), there might **not** be a non-distorting baseline that works in the same situation.

Core-set Sampling Training a GAN with Core-set sampling (sampling factor of 20) results in considerable distortion: 90.9% of samples are from the 0.99-mode and 8.9% of samples are from the 0.01-mode (the rest of the samples were over 4 standard deviations from either mean) - so the 0.01 mode is over-represented by about a factor of 10. This makes a lot of sense: the final batch size is 10, and we take 200 samples to start, so the expected number of 0.01-mode samples in our original batch is 2. Core-set sampling will ensure we keep at least 1 of those 2 in our final batch of 10, yielding a 10% representation for a mode that should only have 1% representation. It's worth pointing out that you have to make the skew over mixture coefficients quite high relative to the batch size for this to matter, but there likely exist interesting data-sets that have this characteristic.

Thus, we test two methods for reducing the distortion:

5.6.2. Annealing Sampling Factors

Training a GAN with Core-set sampling but annealing the sampling factor from 20 to 2 during training results in 98.3% and 1.6% of samples from the 0.99 and 0.01 modes, respectively. This is a substantial reduction in distortion. Moreover, this result is far preferable to the result we got training on the vanilla GAN, which completely dropped the 0.01 mode.

5.6.3. Fine-Tuning on Randomly Sampled Data

Training a GAN with Core-set sampling to start and then random sampling near the end results in 99.5% and 0.5% of samples from the 0.99 and 0.01 modes, respectively, for our best try. This method resulted in some instability - the longer the GAN is training using random samples, the more likely the GAN was to drop the 0.01 mode. For this reason we would probably recommend the annealing method in practice.

A	B	C	D	E	F	G	H	I
18.01	17.8	17.59	17.12	16.83	**16.73**	16.9	17.95	20.79

Table 7. FID scores for CIFAR using SN-GAN. Each of the experiment shows a different pair of over-sampling factors for the prior and target distributions. The factors are listed as: sampling factor for prior distribution $\times$ sampling factor for target distribution. A = 2×2; B = 2×4; C = 4×2; D = 4×4; E = 8×4; F = 4×8; G = 8×8; H = 16×16; I = 32×32

5.6.4. Final Observations:

We have two more observations to make about this: First, the FID should in principle be sensitive to such distortions, and so the fact Core-set sampling was able to improve the FID in some sense suggests that – on CIFAR, LSUN, and ImageNet — the distortion was not significant. Second, Discriminator Rejection Sampling Azadi et al. (2018) seems like another promising way to reduce distortion. We have not omitted accidentally, but because the annealing method described above seems satisfactory for our purposed and is also much simpler.

6. Related Work

6.1. Variance Reduction in GANs

Researchers have proposed reducing variance in GAN training from an optimization perspective, by directly changing the way each of the networks are optimized. Some have proposed applying the extra-gradient method (Chavdarova et al., 2019), and others have proposed casting the *minimax* two-player game as a variational-inequality problem (Gidel et al., 2018). Brock et al. (2018) recently proposed reducing variance by using large mini-batches.

6.2. Stability in GAN Training

Stabilizing GANs has been extensively studied theoretically. Researchers have worked on improving the dynamics of the two player minimax game in a variety of ways (Nagarajan & Kolter, 2017; Mescheder et al., 2018; Mescheder, 2018; Li et al., 2017b; Arora et al., 2017).

Training instability has been linked to the architectural properties of GANs: especially to the discriminator (Miyato et al., 2018). Proposed architectural stabilization techniques include using Convolutional Neural Networks (CNNs) (Radford et al., 2015), using very large batch sizes (Brock et al., 2018), using an ensemble of the discriminators (Durugkar et al., 2016), using spectral normalization for the discriminator (Miyato et al., 2018), adding self-attention layers for the generator and discriminator networks (Vaswani et al., 2017; Zhang et al., 2018) and using iterative updates to a *global* generator and discriminator using an ensemble(Chavdarova & Fleuret, 2018).

Different objectives have also been proposed to stabilize and improve GAN training (Arjovsky et al., 2017; Gulrajani et al., 2017; Li et al., 2017a; Mao et al., 2017; Mroueh & Sercu, 2017; Bellemare et al., 2017; Dieng et al., 2019).

6.3. Core-set Selection

Core-set sampling has been widely studied from an algorithmic perspective in attempts to find better approximate solutions to the original NP-Hard problem (Agarwal et al., 2005; Clarkson, 2010; Pratap & Sen, 2018).

The optimality of the sub-sampled solutions have also been studied theoretically (Barahona & Chudak, 2005; Goldman, 1971). Lucic et al. (2017) theoretically show how training a Gaussian Mixture Models trained using a subset of points found using Core-sets will have similar likelihood to the one trained on the full set of data. Bachem et al. (2017) further demonstrates how Core-sets can be applied to data distributions with non-trivial manifold geometry, such as a Gaussian distribution. See Phillips (2016) for a recent survey on Core-set selection algorithms.

Core-sets have been applied to many machine learning problems such as k-means and approximate clustering (Har-Peled & Mazumdar, 2004; Har-Peled & Kushal, 2007; Bādoiu et al., 2002)), active learning for SVMs (Tsang et al., 2005; 2007), unsupervised subset selection for hidden Markov models (Wei et al., 2013) scalable Bayesian inference, (Huggins et al., 2016) and mixture models (Feldman et al., 2011). We are not aware of Core-set selection being applied to GANs.

6.4. Core-set Selection in Deep Learning

Core-set selection is largely under-explored in the Deep Learning literature, but interest has recently increased. Sener & Savarese (2017) proposed to use Core-set sampling as a batch-mode active learning sampler for CNNs. Their method used the 'embeddings' of a trained network to sample from.

Mussay et al. (2019) proposed using Core-set selection on the activations of a neural network for network compression. Core-set selection has also been used in continual learning to sample points for episodic memory (Nguyen et al., 2017).

7. Conclusion

In this work we present a general way to mimic using a large batch-size in GANs while minimizing computational overhead. This technique uses Core-set selection and improves performance in a wide variety of contexts. Importantly, it is N times faster than using gradient accumulation across N batches. This work also suggests further research: a similar method could be applied to other learning tasks where large mini-batches may be useful, including supervised learning.

We hope that this technique becomes a standard part of the GAN practitioner toolkit. We also hope that it can help drive further advances in research on large, sophisticated GAN models, which without some technique like this are difficult for most researchers to train due to computational constraints.

8. Acknowledgements

We would like to thank Colin Raffel, Abhijeet Jagdev, Nicolas Gagné and Hugo Berard for invaluable feedback on the paper. We would also like to thank Nvidia for donating NVIDIA DGX-1, and Compute Canada for providing resources for this research.

References

Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 224–232. JMLR. org, 2017.

Arora, S., Risteski, A., and Zhang, Y. Do gans learn the distribution? some theory and empirics. 2018.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*, 2018.

Bachem, O., Lucic, M., and Krause, A. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

Bādoiu, M., Har-Peled, S., and Indyk, P. Approximate clustering via core-sets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 250–257. ACM, 2002.

Barahona, F. and Chudak, F. A. Near-optimal solutions to large-scale facility location problems. *Discrete Optimization*, 2(1):35–50, 2005.

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3): 15, 2009.

Chavdarova, T. and Fleuret, F. Sgan: An alternative training of generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9407–9415, 2018.

Chavdarova, T., Gidel, G., Fleuret, F., and Lacoste-Julien, S. Reducing noise in gan training with variance reduced extragradient. *arXiv preprint arXiv:1904.08598*, 2019.

Clarkson, K. L. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):63, 2010.

Dasgupta, S. and Gupta, A. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

Dieng, A. B., Ruiz, F. J., Blei, D. M., and Titsias, M. K. Prescribed generative adversarial networks. *arXiv preprint arXiv:1910.04302*, 2019.

Donoho, D. L. et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, 1(2000):32, 2000.

Durugkar, I., Gemp, I., and Mahadevan, S. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

Eskicioglu, A. M. and Fisher, P. S. Image quality measures and their performance. *IEEE Transactions on communications*, 43(12):2959–2965, 1995.

Farahani, R. Z. and Hekmatfar, M. *Facility location: concepts, models, algorithms and case studies*. Springer, 2009.

Fedus, W., Goodfellow, I., and Dai, A. M. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

Feldman, D., Faulkner, M., and Krause, A. Scalable training of mixture models via coresets. In *Advances in neural information processing systems*, pp. 2142–2150, 2011.

Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S. A variational inequality perspective on generative adversarial networks. *arXiv preprint arXiv:1802.10551*, 2018.

Girod, B. What's wrong with mean-squared error? *Digital images and human vision*, pp. 207–220, 1993.

Goldman, A. Optimal center location in simple networks. *Transportation science*, 5(2):212–221, 1971.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., and Wang, J. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Har-Peled, S. and Kushal, A. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 291–300. ACM, 2004.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Huggins, J., Campbell, T., and Broderick, T. Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pp. 4080–4088, 2016.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Kumar, R., Goyal, A., Courville, A., and Bengio, Y. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., and Kim, K. J. A survey of deep learning-based network anomaly detection. *Cluster Computing*, pp. 1–13, 2017.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pp. 2203–2213, 2017a.

Li, J., Madry, A., Peebles, J., and Schmidt, L. Towards understanding the dynamics of generative adversarial networks. *arXiv preprint arXiv:1706.09884*, 2017b.

Lucic, M., Faulkner, M., Krause, A., and Feldman, D. Training gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1): 5885–5909, 2017.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017.

Mescheder, L. On the convergence properties of gan training. *arXiv preprint arXiv:1801.04406*, 1:16, 2018.

Mescheder, L., Geiger, A., and Nowozin, S. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Mroueh, Y. and Sercu, T. Fisher gan. In *Advances in Neural Information Processing Systems*, pp. 2513–2523, 2017.

Mussay, B., Osadchy, M., Braverman, V., Zhou, S., and Feldman, D. Data-independent neural pruning via coresets, 2019.

Nagarajan, V. and Kolter, J. Z. Gradient descent gan optimization is locally stable. In *Advances in Neural Information Processing Systems*, pp. 5585–5595, 2017.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Phillips, J. M. Coresets and sketches. *arXiv preprint arXiv:1601.00617*, 2016.

Pratap, R. and Sen, S. Faster coreset construction for projective clustering via low-rank approximation. In *International Workshop on Combinatorial Algorithms*, pp. 336–348. Springer, 2018.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Salimans, T., Zhang, H., Radford, A., and Metaxas, D. Improving gans using optimal transport. *arXiv preprint arXiv:1803.05573*, 2018.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Sinha, S., Ebrahimi, S., and Darrell, T. Variational adversarial active learning. *arXiv preprint arXiv:1904.00370*, 2019.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Tsang, I. W., Kwok, J. T., and Cheung, P.-M. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.

Tsang, I. W., Kocsor, A., and Kwok, J. T. Simpler core vector machines with enclosing balls. In *Proceedings of the 24th international conference on Machine learning*, pp. 911–918. ACM, 2007.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wei, K., Liu, Y., Kirchhoff, K., and Bilmes, J. Using document summarization techniques for speech data subset selection. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 721–726, 2013.

Wolsey, L. A. and Nemhauser, G. L. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

Xian, Y., Lorenz, T., Schiele, B., and Akata, Z. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5542–5551, 2018.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Zenati, H., Foo, C. S., Lecouat, B., Manek, G., and Chandrasekhar, V. R. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5907–5915, 2017.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

Zhu, J.-J. and Bento, J. Generative adversarial active learning. *arXiv preprint arXiv:1702.07956*, 2017.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.