Master Thesis

An Investigation into the use of Growing Mini-Batches for Faster GAN Training

Paul H. Disbeschl

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Data Science for Decision Making
at the Department of Data Science and Knowledge Engineering
at Maastricht University

Thesis Committee:

Dr. B. Franci
Dr E. Hortal
Dr. M. Staudigl

Maastricht University
Faculty of Science and Engineering
Department of Data Science and Knowledge Engineering

August 25th, 2022

Abstract

Generative adversarial networks (GANs) are state-of-the-art machine learning frameworks capable of generating highly realistic samples across multiple domains. The complex training process of a GAN comes at a considerable computational cost, and is not reliably guaranteed to converge to a minimum. In this thesis, we investigate possible ways to reduce this computational expense by exploring more efficient processing of mini-batches during training, measuring performance through use of the FID score in terms of training time. We make use of two recent variance reduction strategies applied to GANs which relax the assumptions required for guaranteed convergence during training, these being the stochastic relaxed forward-backward (SRFB) algorithm and the stochastic variance reduced extragradient (SVRE) algorithm. We compare these two approaches, finding that the SRFB method is notably faster to converge compared to the SVRE method. For the SVRE method, we investigate the impact of adjusting the frequency of full batch evaluations on training performance and training time, finding that adjusting the full batch interval for the SVRE method does not lead to significant improvements with regards to training time or FID score. We also examine the effects of growing mini-batch size on the training of the models with respect to model performance and training time. Here, we find that growing mini-batches during training will lead to quicker covering of the number of epochs specified for training, but that performance over time is not improved compared to an approach using fixed mini-batch sizes.
A repository containing the programming code accompanying this thesis can be found at `https://github.com/pdisbeschl/paul_d_growing_minibatches`

Acknowledgements

Thank you.

This research was made possible, in part, using the Data Science Research Infrastructure (DSRI) hosted at Maastricht University.

Contents

List of Figures

List of Abbreviations

- EG - Extragradient
- FID - Fréchet Inception Distance
- IS - Inception Score
- SGD - Stochastic Gradient Descent
- SRFB - Stochastic Relaxed Forward-Backward algorithm
- SVRE - Stochastic Variance Reduced Extragradient algorithm

Chapter 1

Introduction

Machine learning models typically involve the minimization of a loss function, denoted by $\mathcal{L}$, to optimize the cost of the model's predictions relative to the actual values. The loss function is also known as the cost or error function. Finding the local minimum of this function is commonly done in an iterative approach, making repeated steps from point to point in the opposite direction of the gradient, denoted by $\nabla\mathcal{L}$. This approach is called (batch) gradient descent. This algorithm makes use of the entire (training) dataset $\mathcal{D}$ to calculate the gradient of the loss function before taking a next step towards an expected minimum, using step size η [1]. The vector of weight parameters θ at the current iteration i are used to update the weight parameters at the next iteration $i+1$, using the function $\theta_{i+1} = \theta_i - \eta\big(\nabla\mathcal{L}(\theta)\big)$. This continues until a minimum has been reached.

In scenarios where a dataset is relatively large and contains many similar training examples, using the entire dataset to compute the exact gradient before taking a single optimization step can come at a significant computational cost with little to no added benefit compared to using a subset of this dataset [2]. Using the entire dataset can even render the calculation prohibitively expensive, potentially exhausting computer resources [3]. This has led to the development of stochastic gradient descent (SGD) [4], in which a single, randomly selected training example $x^{(j)} \in \mathcal{D}$ is used to calculate the pseudogradient, an approximation of the (true) gradient, at iteration i, using the function $\theta_{i+1} = \theta_i - \eta\big(\nabla\mathcal{L}(x^{(j)})\big)$. This continues until a minimum has been reached. The main advantage of this method is that the parameters are updated a lot quicker compared to gradient descent since only one sample is evaluated per iteration, potentially leading to faster training. Thus, where gradient descent makes redundant computations of the gradient for similar samples, SGD bases its parameter updates on a single sample representing the whole dataset [2]. Taking optimization steps using only one sample at a time may however introduce more noise in the training process, which causes there to be variance in the parameter updates and fluctuations in the loss function, reducing training efficiency [5]. This in turn may lead to the model to potentially overshoot a minimum, which in turn might lead to unnecessarily many computations being performed or the model not converging [3].

A compromise between these two methods is to make use of mini-batches, reducing the variance in the parameter updates introduced by SGD. In this scenario, the training dataset is divided into equally sized collections B of size n, such that $\{x^{(1)}, ..., x^{(n)}\} \in B$ and $B \subset \mathcal{D}$, with optimization steps taking place once all samples in a single mini-batch have been evaluated [2]. While this still uses an approximation of the gradient, the impact of noise is expected to be reduced compared to SGD, since multiple samples are used for the estimate instead of only one [3]. This results in the equation $\theta_{i+1} = \theta_i - \eta\big(\nabla\mathcal{L}(B)\big)$.

Where gradient descent uses the full batch of training data and SGD uses a single sample from the training data, the size of a mini-batch (this referring to the number of samples it contains) is not predefined. Thus it is vital to choose an appropriate mini-batch size which allows the model to achieve an adequate level of accuracy while optimizing computational efficiency. Using a relatively small mini-batch size reduces the number of calculations executed during the optimization steps, allowing these to be made quickly. On the other hand, using a relatively large mini-batch size will likely be more beneficial in terms of accuracy and noise reduction and in causing the model to converge faster to a minimum. Additionally, advances in computing hardware allow for multiple training examples to be processed in parallel for greater computational efficiency, encouraging the use of batch sizes which are a power of two [3].

1.1 GANs: Generative Adversarial Networks

A generative adversarial network (GAN) is an (unsupervised) generative modeling technique, consisting of two artificial neural networks set up to compete in an adversarial fashion [6]. The first of these neural networks is the generator G, tasked with learning how to generate new samples from a distribution similar to that of the training data $\mathcal{D}$, without actually observing the training examples $x \in \mathcal{D}$ directly. Samples created by the generator are fed into the other neural network, the discriminator D, together with real training examples from the training data. The discriminator acts as a binary classifier in this case, classifying the inputs it receives as either fake or real. It outputs an estimated probability score between 0 and 1 inclusive for the likelihood that the input it has received is from the training dataset [7].

GANs can be modeled as a minimax game, with the generator's goal being to deceive the discriminator into classifying generated images as real, and the discriminator aiming to classify generated images as fake. In this scenario, when the discriminator incorrectly classifies a generated example as real, its weights are updated so that it may learn from its mistake. Likewise, the generator is penalized when the discriminator is not deceived by a generated example, and its weights are subsequently updated. This continues up to the point where the discriminator no longer can accurately differentiate whether the input it receives is generated or real. At this point it is anticipated that the model has

converged and that the generator has successfully learned to accurately model the probability distribution present in the training data p_{data}.
Both networks try to minimize their loss functions $\mathcal{L}^G$ and $\mathcal{L}^D$ in terms of their weight parameters θ and φ respectively [8]:

$$\theta^* \in \arg\min_{\theta\in\Theta} \mathcal{L}^G(\theta, \varphi^*) \qquad \text{and} \qquad \varphi^* \in \arg\min_{\varphi\in\Phi} \mathcal{L}^D(\theta^*, \varphi)$$

When $\mathcal{L}^D = -\mathcal{L}^G =: \mathcal{L}$, this can be viewed as a zero-sum game between the two networks, which leads to the minimax problem:

$$\min_{\theta\in\Theta} \max_{\varphi\in\Phi} \mathcal{L}(\theta, \varphi)$$

Since their inception in 2014, GANs have have been applied to many problems across various fields, most notably in the generation of images, such as the generation of pictures of realistic human faces [9], but also in the generation of audio [10] and in creating 3-D objects [11].

1.2 Problem Statement

An inherent problem of GANs is that they are computationally expensive to train successfully. The training of a GAN involves finding a Nash equilibrium to a two-player non-cooperative game, which is a complex problem, since a GAN's learning parameters are continuous and the parameter space is relatively large [12]. Besides this, GAN loss functions often are non-convex, and thus traditional full batch and stochastic gradient descent approaches are not guaranteed to converge unless the (pseudo-) gradient is strongly monotone [13]. Research endeavors have led to approaches which relax this requirement to an almost-sure convergence guarantee for monotone mappings. One of these approaches is the stochastic relaxed forward-backward (SRFB) algorithm, a variant of the stochastic forward-backward (SFB) algorithm [14] which requires a single evaluation of the pseudogradient before updating the model parameters [13]. Another approach is the stochastic variance reduced extragradient (SVRE) [8] - a combination of the stochastic variance reduced gradient (SVRG) [15] and extragradient (EG) [16] approaches. In the SVRE approach, the full batch is periodically evaluated to estimate the gradient, and an extrapolation step of the pseudogradient is performed during each training iteration to more accurately steer the model parameters towards a point of convergence [8]. Both SRFB and SVRE make use of mini-batches.
Owing to the iterative nature of the optimization algorithms used for GAN training, multiple passes over the entire dataset (called epochs) must be carried out in order to improve the model, with each pass requiring many computationally expensive calculations. This research explores the possibilities of reducing this computational expense by investigating the optimization of mini-batch evaluations, applied to the image generation problem. First, the two aforementioned

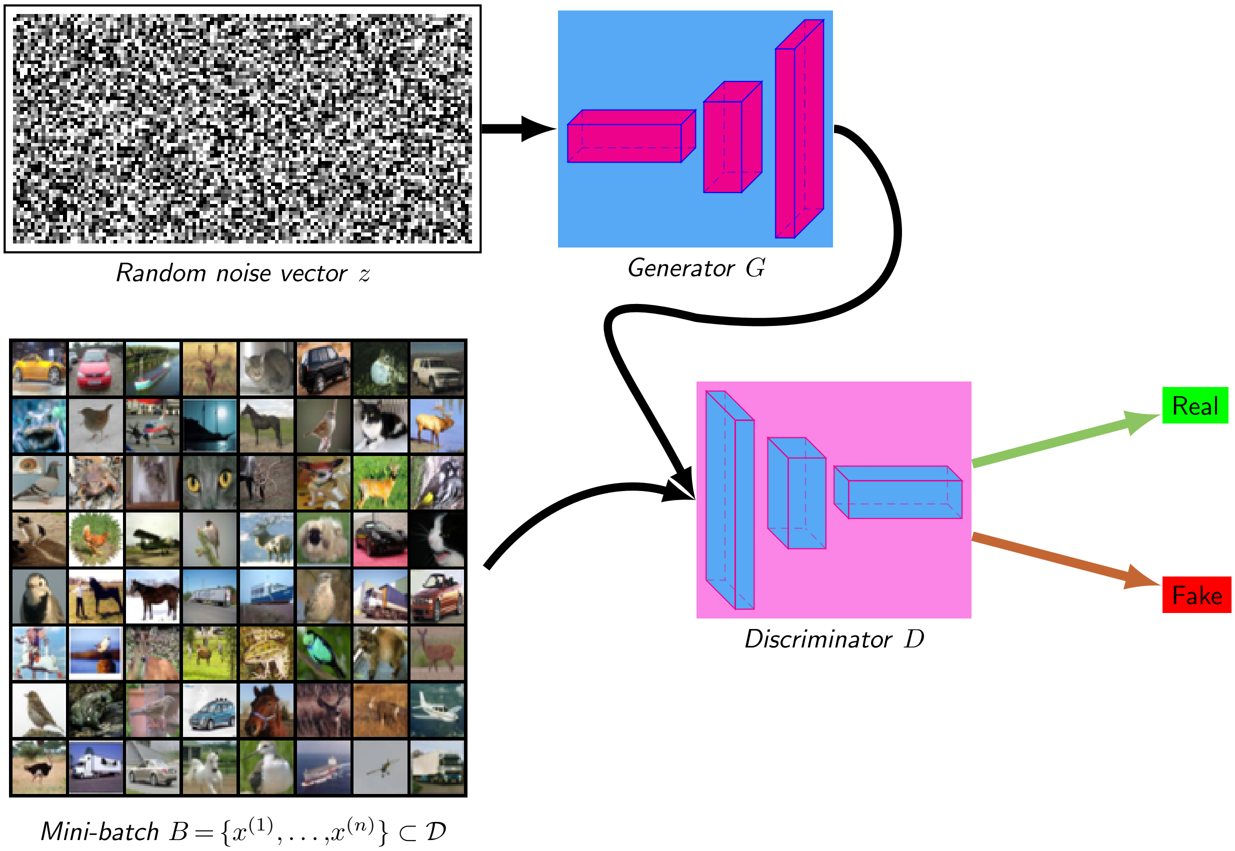

Random noise vector z
Generator G
Discriminator D
Real
Fake
Mini-batch $B = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathcal{D}$

Figure 1.1: A visualization of a single iteration of GAN training. A randomly generated noise vector z is used as an input to the generator G, which outputs a mini-batch of generated samples $G(z)$. This generated mini-batch is then used as an input to the discriminator D, together with a mini-batch B containing real samples from the training dataset $\mathcal{D}$. The discriminator subsequently classes these inputs as real or fake.

variance reduction techniques for GAN training are compared in terms of computational expense. With the most costly operation for the SVRE approach being the periodical evaluation of the full batch, we investigate the impact of altering the frequency of this full batch evaluation on the computational expense. For the SRFB method, we explore the progressive growing of mini-batches from small to large throughout the training process in terms of training time and iteration complexity. The goal is to evaluate the possibilities and advantages of using growing mini-batch sizes in GANs employing SRFB to reduce computational complexity while maintaining convergence guarantees under the same assumptions.

This all leads to the following research questions:

1.3 Research Questions

1. How do two recent variance reduction strategies for GAN training (SRFB and SVRE) compare in terms of computational cost?
   (a) How do these two methods compare in terms of iteration complexity?
2. Can the computational expense of employing variance reduction strategies in GAN training be reduced through the progressive growing of mini-batches?
3. How does growing mini-batch size affect the complexity of the training process in terms of iteration complexity and as a function of time?

1.4 Outline

The background behind the problem is explained in Chapter 2, together with a more detailed description of GANs and the algorithms listed above, and an overview of related work. Following this, in Chapter 3, is an explanation of how the problem is approached, followed by a description of the experiments performed and results obtained in Chapter 4. The results are discussed in Chapter 5, followed by the conclusion and potential future work in Chapter 6.

Chapter 2

Background and Related Work

At the start of this chapter, the GAN training process is described in further detail, in Section 2.1. This is followed by a description of related work in the field and descriptions of the algorithms studied, in Section 2.2.

2.1 Training a GAN

GANs employ an iterative optimization algorithm, such as gradient descent, for updating the parameter weights (θ, φ) of the generator and discriminator networks (G, D) with respect to their losses ($\mathcal{L}_G$, $\mathcal{L}_D$). As is typical for iterative learning methods, the training process occurs in epochs E. One epoch corresponds to the period in which all the samples in the training dataset have been used for a single forward and backward pass of the networks, $E = |\mathcal{D}|$.

When using gradient descent, the error is calculated for each training example $x^{(j)} \in \mathcal{D}$, but the model is only updated once at the end of the epoch. While it is possible to train the networks on the entire dataset as a single batch, this may come at a prohibitive computational cost when compared to a smaller-sized batch, especially if the dataset is relatively large - as is typical for GANs [8]. In the case of GANs, performing all these extra computations in the early phases of the training process may also not lead to a significant improvement (if any) over training on a subset of this full batch. In the initial stages of training, both models are still untrained, and using the full batch to update the model parameters typically does not lead to improved performance over using a smaller batch, since the noise between samples in the training data does not yet play a significant role.

An alternative to training using the full batch is a pure stochastic approach. In the case of stochastic gradient descent (SGD), the error is also calculated for each training example, but the model is updated immediately afterwards. While this may reduce the computational expense of parameter updating, the noise between examples may result in an undesirably high variance of the error function over time, leading to more updates being required for the models to converge. It can also lead to a situation where the GAN oscillates between generating different kinds of samples without reaching an equilibrium [17]. Thus, using a pure stochastic approach might solve issues present when using a full batch, but introduces new issues along with it. Hence, a popular approach for GAN training using iterative methods is to make use of mini-batches of training examples as a compromise between the two methods.

Mini-batches are collections of training samples $B = \{x^{(1)}, \dots, x^{(n)}\}$, where the mini-batch method is to first calculate the approximated gradients of the loss function using an average over the samples in the collection, and subsequently update the model parameters afterwards. Through the use of parallelization, calculating the approximated gradient for a relatively small mini-batch of samples is not too costly compared to a stochastic approach, yet the reduction in the number of parameter updates makes this approach considerably faster [12]. How mini-batches are used in a single instance of GAN training is described in Section 2.1.1.

2.1.1 Description of a Single Training Iteration

A single iteration refers to a single update of both the generator and discriminator parameters; a visualization of this can be seen in Figure 1.1, with pseudocode of the Vanilla GAN [6] in Algorithm 1. At the start of a training iteration, a single mini-batch B of instances $\{x^{(1)}, \dots, x^{(n)}\}$ is sampled from the training data $\mathcal{D}$. At the same time, a mini-batch z (of the same size and dimensions as the mini-batch B) of random noise vectors $\{z^{(1)}, \dots, z^{(n)}\}$ is used as an input for the generator network, which in turn outputs a corresponding mini-batch of generated samples $G(z)$. Both of these mini-batches are then used as an input for the discriminator network D. The value the discriminator outputs $D(\cdot) \in [0, 1]$ is the predicted likelihood of the samples in the mini-batch being classified as real, with a value closer to 1 indicating it is more likely to be real. Both prediction values $D(B) \in [0, 1]$ and $D(G(z)) \in [0, 1]$ are then used as inputs to a loss equation to calculate the losses at this step. Then, the gradient is estimated at this point, followed by the updating of the model parameters.

Algorithm 1 Vanilla GAN [6]

```
1:  Input: Stopping iteration $I$ learning rates $\eta_\theta$, $\eta_\varphi$ , initial parameter weights
    $\theta_0$, $\varphi_0$
2:  for $i = 0$ to $I$ do
3:      Sample mini-batch of $n$ noise samples $z^{(1)}, \ldots, z^{(n)}$ from normal distribution
4:      Sample mini-batch of $n$ generated outputs $G(z^{(1)}), \ldots, G(z^{(n)})$
5:      Sample mini-batch of $n$ real samples $x^{(1)}, \ldots, x^{(n)}$ from the training dataset $\mathcal{D}$
6:      Calculate the discriminator's loss function:
        $\mathcal{L}^D = \frac{1}{n} \sum_{j=1}^{n} \big[\log D\big(x^{(j)}\big) + \log\big(1 - D\big(G\big(z^{(j)}\big)\big)\big)\big]$
7:      Update the discriminator's parameter weights:
        $\varphi_{i+1} = \varphi_i - \eta_\varphi(\nabla \mathcal{L}^D))$
8:      Repeat steps 3 and 4
9:      Calculate the generator's loss function:
        $\mathcal{L}^G = \frac{1}{n} \sum_{j=1}^{n} \log\big(1 - D\big(G\big(z^{(j)}\big)\big)\big)$
10:     Update the generator's parameter weights:
        $\theta_{i+1} = \theta_i - \eta_\theta(\nabla \mathcal{L}^G))$
11: end for
```

2.2 Optimizing GAN Training with Variance Reduction Methods

Traditionally, GANs have employed a form of the gradient descent algorithm for the optimization of the loss functions of the generator and discriminator networks. A drawback of this approach is that it requires strong monotonicity in order to guarantee convergence, which forms a challenge in their use for training GANs since these tend to be non-convex [13]. Extensions have been made in the deterministic realms to the traditional FB approach in order to relax this strong monotonicity requirement. The first of these is the extragradient (EG) algorithm, which makes use of an extra look-ahead step of the gradient when updating the model parameters [16]. The second is the forward-backward-forward algorithm (FBF) [18], which distinguishes itself from the EG method by only requiring a single projection step instead of the two required by the EG method [19]. While both of these methods guarantee convergence for monotone mappings under traditional circumstances, they both require two evaluations of the gradient instead of just one, increasing the computational cost. Additionally, it is assumed in both of these approaches that the full batch is used to evaluate the gradient. Since GANs typically require a relatively large dataset for training, the computational expense is increased significantly; besides this, they are non-deterministic. Thus, stochastic variants of the two algorithms have been proposed, which allow for these approaches to be used without the full batch requirement [19] [20]. As mentioned in Section 2.1, a drawback of stochastic methods is the increased impact noise has in the estimation of the gradient. This increased noise can lead the algorithm to diverge from the minimum, even in cases where the full batch does converge [8]. Since GANs typically require the dataset not only to be sufficiently large but also relatively diverse (potentially introducing more variance between samples), implementation of these methods in GAN training requires variance reduction techniques to be employed. With the goal of reducing the computational cost of GAN training without sacrificing the ability of the algorithm to converge, two methods from the literature are explored. These are the stochastic relaxed forward-backward algorithm and the stochastic variance reduced extragradient algorithm, described briefly below in Section 2.2.1 and Section 2.2.2 respectively. Mathematical explanations and implementation details can be found at the start of the next chapter, in Section 3.1.

2.2.1 SRFB: Stochastic Relaxed Forward-Backward Algorithm

Franci and Grammatico's stochastic relaxed forward-backward algorithm (SRFB) [13] is an extension of the FB algorithm. Where the FB algorithm assumes strong monotonicity for convergence, SRFB relaxes this assumption, guaranteeing an almost-sure convergence to a neighborhood of the minimum for monotone mappings. Where the FBF and EG algorithms involve two costly evaluations of the pseudogradient mapping per iteration, the SRFB algorithm only requires one, reducing the computational expense of a single iteration of GAN training.

2.2.2 SVRE: Stochastic Variance Reduced Extragradient

The stochastic variance reduced extragradient (SVRE) algorithm, created by Chavdarova et al. [8], is an extension of the extragradient (EG) algorithm [16], combining it with the stochastic variance reduced gradient (SVRG) [15]. The EG look-ahead step is performed during each iteration of the training process for a fixed mini-batch size, improving convergence performance by increasing stability [20]. Periodically, according to a random geometric distribution, the full batch is also sampled during an iteration to provide an unbiased estimate of the gradient. While this comes at a significant computational expense, this allows for a more stable convergence.

Chapter 3

Methodology

This research aims to investigate the impact of mini-batch size in GAN training on the computational expense of the process. One of these aims is to explore the use of growing mini-batches in the training process. The underlying hypothesis is that using smaller mini-batches in GAN training is likely more beneficial in the early stages of GAN training, since both networks are still untrained. This is when the generator outputs mostly random noise and the discriminator classifies almost all instances of the generated outputs as fake. It is thus anticipated that the higher relative variance in noise between mini-batches does not significantly hamper the ability for either network to improve in these early stages, since in both cases the loss functions need to be encouraged to head in the correct direction towards a minimum, rather than to a specific minimum. As the networks improve however, it is anticipated that this noise can play a detrimental role in the training process, causing the model to converge slower or not at all. This could potentially increase the number of parameter updates even more as a result. This is where larger mini-batch sizes have the advantage, since they are likely to converge faster at the cost of having to do more calculations per parameter update, which is not beneficial early on in the training process. As such, it seems that there may be advantages to using multiple mini-batch sizes throughout the training process in terms of computational efficiency. Thus, it is the aim of this research to investigate the role of growing mini-batch sizes on the training process of a GAN in terms of computational efficiency and performance, using the SRFB algorithm described in Section 2.2.1. The goal is to investigate how this compares to an approach where, at random intervals, the full batch is used instead of the mini-batch - the size of which is kept constant, using the SVRE algorithm described in Section 2.2.2. Also explored is the impact of adjusting the interval at which the SVRE algorithm evaluates the full batch on the computational complexity of the training process.

This chapter starts with details on the implementation of the SRFB and SVRE variance reduction algorithms for GANs in the context of the image generation problem, in Section 3.1. Next are details of the neural network architectures used for the GANs in Section 3.2, followed by a description of the dataset used in Section 3.3. Afterwards, an overview is given of the metrics used to evaluate the performance of the GANs during training, in Section 3.4.

3.1 Variance Reduction Methods

As explained in Section 2.1, the main challenge in GAN training is the issue of non-convergence. While the underlying goal of this thesis is to explore whether the computational expense of GAN training can be reduced through more efficient usage of mini-batches, this must not come at the cost of non-convergence. Thus, two variance reduction methods are explored which reduce the impact of stochastic gradient noise in the training process. These are the stochastic relaxed forward-backward (SRFB) and stochastic variance reduced extragradient (SVRE) algorithms, already introduced in Section 2.2.1 and Section 2.2.2, respectively. How these are implemented in the training process of the GAN is described below in Section 3.1.1 and Section 3.1.2.

3.1.1 Stochastic Relaxed Forward-Backward Algorithm

We replicate the SRFB method for GANs defined by Franci and Grammatico [13], seen in Algorithm 2. At first, a set of parameters (θ, φ) is randomly initialized together with a similar set of shadow parameters $(\bar{\theta}, \bar{\varphi})$ respectively for the generator and discriminator. A constant relaxation parameter δ is also defined. At each update iteration, mini-batches are sampled and used to calculate losses, after which the approximated gradient is calculated and the shadow parameters are updated. Each shadow parameter $\bar{\theta}_i, \bar{\varphi}_i$ is updated to be the difference between 1 and δ, multiplied by its non-shadow parameter counterpart at this iteration θ_i, φ_i, plus the resulting product of δ and the value of this shadow parameter at the previous state $\bar{\theta}_{i-1}, \bar{\varphi}_{i-1}$. These updated shadow parameters are subsequently used to update the (non-shadow) parameters of the generator and discriminator at the next iteration. This is done by taking the projection over the difference between the updated shadow parameters and the product of time-varying weights λ with the approximations of the gradients with respect to their parameters $\nabla\mathcal{L}^G, \nabla\mathcal{L}^D$. More details can be found in the original work, [13].

Algorithm 2 SRFB [13]

```
1: Input: Dataset $\mathcal{D}$ and noise dataset $\mathcal{Z}$, where ($|\mathcal{Z}| = |\mathcal{D}| = m$), iteration limit $I$, mini-batch size $|B|$, initial parameter weights $\theta_0 \in \Theta$, $\varphi_0 \in \Phi$, $\delta \in [\frac{2}{1+\sqrt{5}}, 1]$
2: Iteration $i$:
    1: Sample mini-batches $(B_{\mathcal{D}}, B_{\mathcal{Z}})$
    2: Calculate losses $\mathcal{L}^G, \mathcal{L}^D$ for $(B_{\mathcal{D}}, B_{\mathcal{Z}})$, using Equations 3.4 and 3.3
    3: Calculate approximated $\nabla\mathcal{L}^G, \nabla\mathcal{L}^D$ using Equations 3.6 and 3.5
    4: Update parameter weights $\theta \in \Theta, \varphi \in \Phi$:
```

$$\bar{\theta}_i = (1-\delta)\theta_i + \delta\bar{\theta}_{i-1} \qquad \bar{\varphi}_i = (1-\delta)\varphi_i + \delta\bar{\varphi}_{i-1} \tag{3.1}$$

$$\theta_{i+1} = \mathrm{proj}_\Theta\Big[\bar{\theta}_i - \lambda_i\nabla\mathcal{L}^G\Big] \qquad \varphi_{i+1} = \mathrm{proj}_\Phi\Big[\bar{\varphi}_i - \lambda_i\nabla\mathcal{L}^D\Big] \tag{3.2}$$

While mostly similar to the vanilla GAN described in Section 2.1.1, the SRFB method updates the parameters differently, using the steps from Equations 3.1 and 3.2 after the evaluation of the pseudogradients. The time varying weights λ depend on the learning algorithm chosen, which in this case is the Adam optimization method.

3.1.2 Stochastic Variance Reduced Extragradient Algorithm

As described in Section 2.2.2, the SVRE method is a combination of the SVRG and EG methods. We replicate the SVRE-GAN setup of Chavdarova et al. [8], seen in Algorithm 3. Similar to the vanilla GAN described in 2.1.1, the SVRE adds the extrapolation steps (lines $7-9$ below) before updating the parameters for new samples of mini-batches. A geometric distribution is queried in each training iteration; with a probability of $\frac{1}{N}$ the full batch is used to calculate the estimation of the gradient, where N is the number of epochs $N = \frac{|\mathcal{D}|}{|B|}$. The full batch estimate for both the generator and the discriminator's parameters is then calculated and saved as a snapshot, respectively represented as θ^S and φ^S. μ represents the full batch gradient estimate, computed using Equations 3.5 and 3.6. Since use is made of the SVRG algorithm, $\mathbf{d}$ represents the estimated variance reduced gradient at time i, as seen in Equations 3.7 and 3.8. More details can be found in the original work, [8].

Algorithm 3 Pseudocode for GAN Training using SVRE [8]

Input: Dataset $\mathcal{D}$, noise dataset $\mathcal{Z}(|\mathcal{Z}| = |\mathcal{D}| = m)$, learning rates $\eta_\theta, \eta_\varphi$, iteration limit I, mini-batch size $|B|$, initial weights θ_0, φ_0

```
for $e = 0$ to $I$ do
    $\varphi^S = \varphi$ and $\mu_\varphi = \frac{1}{m}\sum_j^m \sum_k^m \nabla\mathcal{L}^D(\theta^S, \varphi^S, \mathcal{D}_k, \mathcal{Z}_j)$
    $\theta^S = \theta$ and $\mu_\theta = \frac{1}{m}\sum_j^m \nabla\mathcal{L}^G(\theta^S, \varphi^S, \mathcal{Z}_j)$
    $N \sim Geom\left(\frac{|B|}{m}\right)$
    for i=0 to $N - 1$ do
        Sample mini-batches $(B_\mathcal{D}, B_\mathcal{Z})$; do extrapolation:
        $\tilde{\varphi} = \varphi - \eta_D \mathbf{d}_\varphi(\theta, \varphi, \theta^S, \varphi^S, B_\mathcal{D}, B_\mathcal{Z})$
        $\tilde{\theta} = \theta - \eta_G \mathbf{d}_\theta(\theta, \varphi, \theta^S, \varphi^S, B_\mathcal{Z})$
        Sample new mini-batches $(B_\mathcal{D}, B_\mathcal{Z})$; do update:
        $\varphi = \varphi - \eta_D \mathbf{d}_\varphi(\tilde{\theta}, \tilde{\varphi}, \theta^S, \varphi^S, B_\mathcal{D}, B_\mathcal{Z})$
        $\theta = \theta - \eta_G \mathbf{d}_\theta(\tilde{\theta}, \tilde{\varphi}, \theta^S, \varphi^S, B_\mathcal{Z})$
    end for
end for
```

3.2 Neural Network Architectures

We replicate the neural network setup of Chavdarova et al. [8] in this research for all experiments, shown in Table 3.1. The neural network architectures for the generator and discriminator networks are based on the SAGAN architectures developed by Zhang et al. [21]. Where the vanilla GAN described in Section 1 uses multilayer perceptron networks, the SAGAN architecture makes use of convolutional layers, making its use ideal for the image generation problem. Additionally, it contains Self-Attention Blocks, which allow the networks to focus their attention on the most important features in the images [21]. Batch normalization and spectral normalization are used to stabilize the learning process. Hinge loss equations are used for training; these can be found below in Equations 3.3 and 3.4. Using the same notation as Chapter 2, x is a training instance, $D(x)$ is the output of the discriminator after receiving this instance as an input, with $D(G(z))$ being the output of the discriminator after receiving as an input an image created by the generator from its random noise input z.

$$\mathcal{L}^D = \mathbb{E}_{x \sim p_{data}} \max(0, 1 - D(x)) + \mathbb{E}_{z \sim p_z} \max(0, 1 + D(G(z))) \tag{3.3}$$

$$\mathcal{L}^G = -\mathbb{E}_{z \sim p_z} D(G(z)) \tag{3.4}$$

The approximated gradients of these loss functions can be found below in Equations 3.5 and 3.6. $B_\mathcal{D}$ and $B_\mathcal{Z}$ refer to mini-batches of size $|B|$ of the training data and the noise data, of which there are N in total for both, $N = \frac{|\mathcal{D}|}{|B|}$. The gradients are computed over all the samples within these mini-batches.

$$\nabla \mathcal{L}^D(\theta, \varphi, B_{\mathcal{D}}, B_{\mathcal{Z}}) = \frac{1}{|B_{\mathcal{Z}}|} \frac{1}{|B_{\mathcal{D}}|} \sum_{j \in B_{\mathcal{Z}}} \sum_{k \in B_{\mathcal{D}}} \nabla \mathcal{L}^D(\theta, \varphi, B_{\mathcal{D}}^{(k)}, B_{\mathcal{Z}}^{(j)}) \tag{3.5}$$

$$\nabla \mathcal{L}^G(\theta, \varphi, B_{\mathcal{Z}}) = \frac{1}{|B_{\mathcal{Z}}|} \sum_{j \in B_{\mathcal{Z}}} \nabla \mathcal{L}^G(\theta, \varphi, B_{\mathcal{Z}}^{(j)}) \tag{3.6}$$

When using the SVRG method, the gradient is calculated differently, taking into account the full batch gradient μ and the snapshots of the full batch parameters θ^S and φ^S.

$$\mathbf{d}_{\varphi}(\theta, \varphi, \theta^S, \varphi^S, B_{\mathcal{D}}, B_{\mathcal{Z}}) = \mu_D + \nabla \mathcal{L}^D(\theta, \varphi, B_{\mathcal{D}}, B_{\mathcal{Z}}) - \nabla \mathcal{L}^D(\theta^S, \varphi^S, B_{\mathcal{D}}, B_{\mathcal{Z}}) \tag{3.7}$$

$$\mathbf{d}_{\theta}(\theta, \varphi, \theta^S, \varphi^S, B_{\mathcal{D}}, B_{\mathcal{Z}}) = \mu_G + \nabla \mathcal{L}^G(\theta, \varphi, B_{\mathcal{Z}}) - \nabla \mathcal{L}^G(\theta^S, \varphi^S, B_{\mathcal{Z}}) \tag{3.8}$$

Generator	**Discriminator**
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, 1)$	*Input*: $x \in \mathbb{R}^{3 \times 32 \times 32}$
Transposed Convolutional Layer (ker: $4 \times 4, 128 \rightarrow 256$, str: 1) Spectral Normalization Batch Normalization ReLU	Convolutional Layer (ker: $4 \times 4, 3 \rightarrow 64$, str: 2, pad: 1) Spectral Normalization LeakyReLU (negative slope: 0.1)
Transposed Convolutional Layer (ker: $4 \times 4, 256 \rightarrow 128$, str: 2, pad 1) Spectral Normalization Batch Normalization ReLU	Convolutional Layer (ker: $4 \times 4, 64 \rightarrow 128$, str: 2, pad: 1) Spectral Normalization LeakyReLU (negative slope: 0.1)
Self-Attention Block (128)	
Transposed Convolutional Layer (ker: $4 \times 4, 128 \rightarrow 64$, str: 2, pad: 1) Spectral Normalization Batch Normalization ReLU	Convolutional Layer (ker: $4 \times 4, 128 \rightarrow 256$, str: 2, pad: 1) Spectral Normalization LeakyReLU (negative slope: 0.1)
Self-Attention Block (64)	Self-Attention Block (256)
Transposed Convolutional Layer (ker: $4 \times 4, 64 \rightarrow 3$, str: 2, pad: 1) $\tanh(\cdot)$	Convolutional Layer (ker: $4 \times 4, 256 \rightarrow 1$, str: 1)

Table 3.1: The respective neural network architectures for the Generator and Discriminator.

The optimization algorithm used for the neural networks is Adam, an adaptive moment estimation algorithm [22]. Unlike SGD, which maintains a single learning rate, Adam computes adaptive learning rates for individual parameters from estimates of first and second moment estimates of the gradients. Adam is commonly used in GAN training, is also used in the works of Chavdarova et al. [8] and Franci and Grammatico [13] for the SVRE and SRFB algorithms, respectively. The default settings for Adam in the original SAGAN paper, also used in the two aforementioned works, are replicated here: a learning rate η of 0.0002, a first momentum β_1 of 0.5, a second momentum of β_2 of 0.9.

3.3 Dataset

In order to evaluate the models, use is made of the CIFAR10 image dataset compiled at the Canadian Institute For Advanced Research by Krizhevsky, Hinton, and Nair [23]. This dataset contains 50, 000 training images of 10 mutually exclusive image classes [23]. The classes refer to the subjects of the images, composed of various animals and modes of transport. An overview of the classes can be seen in Figure 3.1 together with 10 example images of each class. CIFAR10 contains 5, 000 color images of each class of size 32 by 32 pixels, and thus has the advantage of being fairly lightweight. The dataset is commonly used in machine learning and computer vision research, and various deep learning frameworks, such as PyTorch and Keras, have built-in commands to load the dataset automatically. Franci and Grammatico [13] and Chavdarova et al. [8] also use the dataset in their works for their experiments using the SRFB and SVRE respectively.

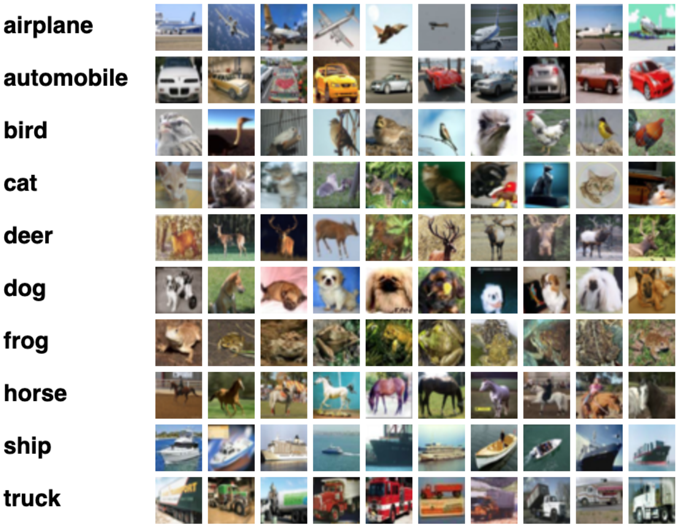


Figure 3.1: An overview of the various image classes inside the CIFAR10 dataset, with examples for each of these [23]

3.4 Metrics

Measuring the quality of a generated images is a subjective matter. In order to measure the output quality of the images, two metrics are used, the inception score (IS) and the Fréchet inception distance (FID), detailed in Sections 3.4.1 and 3.4.2 respectively.
In the PyTorch implementation of the GANs used in this research, both the IS and the FID scores are calculated for generated images and images from the training dataset using the `pytorch-gan-metrics` library [1]. This library computes the FID and IS at predefined intervals during training. It should be noted that multiple implementations for calculating FID and IS scores exist, and thus the results obtained in this research can be compared to each other, but cannot directly be compared to other implementations.

3.4.1 Inception Score

Inception score (abbreviated as IS) is a metric used to estimate the quality of a generated image [12]. IS uses a pre-trained version of the Inception-v3 network (itself an image analysis and object detection network) to evaluate the distribution of generated images, with a higher score indicating a higher quality image. Applying this network to generated images provides a conditional label distribution $p(y|x)$, corresponding to the probability that an image x has label y. The underlying theory is that an image labeled as belonging to one class with a high probability over other classes (low entropy) is more likely to contain meaningful objects than images where this is not the case, and thus is more likely to be of higher quality. Generated images must also be sufficiently diverse however; the marginal label distribution $p(y) = \int_x p(y|x)p_g(x)$ should have a high entropy to ensure that images receive appropriate labels. The Kullback-Leibler divergence (denoted by D_{KL}) is a statistical distance to show how different two probability distributions are over all the samples. Thus, the IS formula in Equation 3.9 calculates the KL divergence between the distributions over all samples in the generated probability distribution $x \sim p_g$, using the following equation:

$$\text{IS} = \exp\left(\mathbb{E}_{x\sim p_g} D_{KL}(p(y|x)||p(y))\right) \tag{3.9}$$

The higher the IS, the more likely it is that images generated using the probability distribution p_g are 'realistic'. A drawback of this metric is that the calculation of how 'realistic' an image is does not involve any comparison between generated images and real images, since it is only based on what the Inception-v3 network classifies the images as.[24] This limitation is addressed by the Fréchet inception distance, which calculates the similarity of generated images to real images in the training data to establish an estimate of the quality of generated images, and thus the performance of a generator network.

[1] https://github.com/w86763777/pytorch-gan-metrics

3.4.2 FID: Fréchet Inception Distance

Fréchet inception distance, henceforth referred to as FID, is a metric for calculating the quality of images generated by machine learning models compared to images from the training dataset [24]. Where the IS uses the full Inception-v3 network, FID uses the activations from the coding layer of this network to extract computer vision relevant features of images, both generated and real. These features are summarized as multidimensional Gaussians, calculated using the mean (μ) and covariance matrices (C) of the feature vectors. Thus, by calculating the Fréchet distance between these Gaussians, the relative likeness between samples from the real (r) and generated (g) distributions is calculated. The lower the FID score, the more 'realistic' the image is.

$$\text{FID score} = ||\mu_r - \mu_g||_2^2 + Tr\big(C_r + C_g - 2(C_r C_g)^{\frac{1}{2}}\big) \tag{3.10}$$

While the FID score currently is the industry standard metric for calculating GAN performance, a major limitation of FID is that it remains an approximation of human judgment; evaluating the quality of a generated image is a subjective endeavor.

Chapter 4

Experiments and Results

This chapter describes the experiments performed to evaluate the use of growing mini-batches in GAN training using the SRFB algorithm, and the changing of the full batch interval using the SVRE algorithm. Each section corresponds to an experiment performed, with details regarding the experimental setup, followed by a visualization of the results and a subsequent description of these. Section 4.1 corresponds to the initial investigation into the effects of growing batch sizes in GAN training using the SRFB algorithm. Section 4.1.2 details the initial investigations into changing when the full batch evaluation is scheduled in GAN training using the SVRE algorithm. Section 4.2 describes a deeper exploration in mini-batch growing for the SRFB algorithm.
As detailed in Chapter 3, the experiments are performed in the context of the image generation problem using the CIFAR10 dataset, with FID score as a performance metric, and using SAGAN architectures with Adam (as described in Section 3.2) for both the SRFB and the SVRE algorithms. All simulations are performed using PyTorch 1.11 in JupyterLab (Python 3.8.12) on a single Nvidia Tesla V100 GPU with 32 GB of RAM. The experiments were performed using the Data Science Research Infrastructure at Maastricht University. For the experiments related to the SVRE, a mini-batch size of 64 images per batch was used, replicating the experimental setup of Chavdarova et al. [8]. The FID score is calculated every $5,000$ iterations, corresponding to 6.4 epochs when training on a batch size of 64. Since calculating the FID is a time intensive process in and of itself, this interval was chosen to reduce the impact of FID calculations on the results with regards to time.

4.1 Initial Investigations in Mini-Batch Optimization of GAN Training

This section details the initial experiments conducted to explore the possibilities of improving the SRFB and SVRE approaches in terms of training time and iteration complexity by adjusting their mini-batch evaluation behaviors in GAN training. The goal is to investigate whether the process can be made more computationally efficient while maintaining convergence to an adequate result. Since deep learning models are typically trained on GPUs which can process mini-batches in parallel and thus prefer mini-batch sizes to be a power of two, the mini-batch size increases by a factor of two each time it is scheduled to grow. The motivation behind this doubling strategy is to maximize the efficient benefits of parallelization.

4.1.1 Scheduling Mini-Batch Growth for the SRFB Algorithm

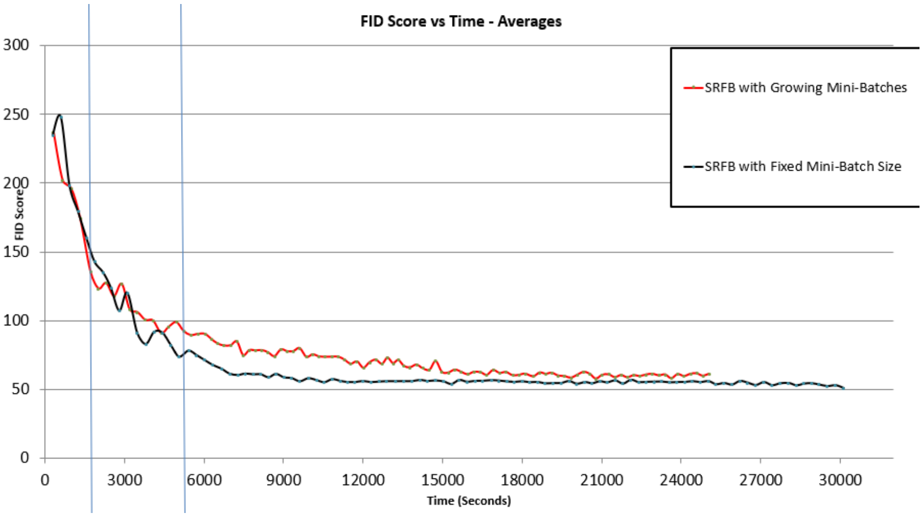
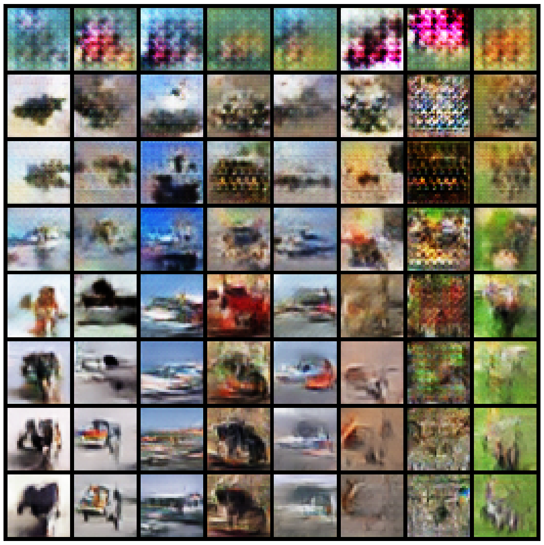


Figure 4.1: A plot of the averaged results of GAN training using the SRFB algorithm, based on two trials for both fixed-size mini-batches and growing mini-batches. All four trials initially use a mini-batch size of 64 samples, but in the case of the growing mini-batch trials, the mini-batch size grows to 128 samples after 32 epochs, as marked by the first blue line. The second blue line corresponds to the second mini-batch size growing stage, after 96 epochs, when the mini-batches are grown to 256 samples.

In order to establish a baseline for the SRFB's performance over time, control tests were performed using a fixed mini-batch size of 64 images for 640 epochs. The averaged result corresponds to the black line seen in Figure 4.1, a plot of the FID score over time, in seconds. Generally, the FID score steadily decreases over time until $1,781$ seconds, when it has covered 32 epochs, corresponding to $25,000$ parameter updates, after which the gradient of FID over time starts to become slightly less steep. Following this, after two slight fluctuations, the gradient of FID over time become noticeably less steep once again, in the region of $5,231$ seconds, after 96 epochs, corresponding to $75,000$ parameter updates.

Afterwards, the FID starts decreasing slowly over time with little to no fluctuations until the halting condition of 640 epochs has been reached at after running for over 30,000 seconds (8 hours and 20 minutes). A selection of generated samples can be found in Figure 4.2, showing the output of the GAN using the SRFB approach with a fixed mini-batch size of 64 [1]. The changes in the gradient of the FID over time visible in Figure 4.1 correspond to the changes visible between the generated samples. Where at first the images are still fairly noisy, they do improve over time, with more variety between the images generated and more details appearing in the images. By the third row, after 32 epochs, the images become noticeably less noisy than before, and start to contain more realistic looking objects. This is especially true for images in the sixth row (after 96 epochs of training), and beyond. There is also with more variety amongst the images themselves and more details appearing in the images. As an initial test, the effects of growing the mini-batch size at these two points heuristically chosen by visual inspection (32 and 96 epochs) was investigated. Starting with a batch size of 64 images per mini-batch, the mini-batch size was doubled at 25,000 iterations to 128 samples, and then doubled further at 50,000 iterations to 256 samples per mini-batch. Note that this second doubling occurs after 96 epochs, which is when the fixed mini-batch GAN has covered 64 epochs, at 75,000 iterations. Initially, the FID scores of samples generated by



Epochs	Time (s)	FID
6.4	336	262
25.6	1435	170
32	1781	142
36.4	2113	127
64	3448	97
96	5231	76
128	6963	61
160	8611	59

Figure 4.2: A selection of image samples generated at several stages of GAN training using the SRFB algorithm with a fixed mini-batch size of 64. Images are generated using a fixed random noise vector z as an input, allowing the learning process to be visualized

[1] More images sampled throughout the training process can be found in Appendix A.

the growing mini-batch trials appear to be lower than the scores obtained in trials using a fixed mini-batch size, despite the configurations of these networks being identical for the first 32 epochs. This can be attributed to the randomness of the models involved, since there are no other differentiating factors influencing model performance at this state of the training process. The results of the individual trials, seen in Figure 4.3, encourage this hypothesis. Noticeable are the fluctuations throughout the training process; the fluctuations of the growing mini-batch trials preceding batch growth which are not visible in the first fixed-batch size trial can only be attributed to the stochastic nature of the training process. As for the jittery gradient towards the end of the experiments for the growing mini-batch trials, this may be attributed to how the data is collected, since the FID is calculated on a per-epoch basis (once every 6.4 epochs) - if done on a per-parameter-update basis (for instance, once every 5, 000 parameter updates), the gradients are smoother. This further demonstrates the role of stochasticity and noise in the training process. After a few larger fluctuations in the FID score in the initial stages of training, the growing mini-batch trials start showing signs of convergence after 14, 000 seconds, close to 352 epochs (100, 000 iterations), with slight variations in FID continuing to occur until the end of training at 640 epochs (156, 250 iterations), equivalent to when the fixed mini-batch GAN has been trained for 500, 000 iterations.

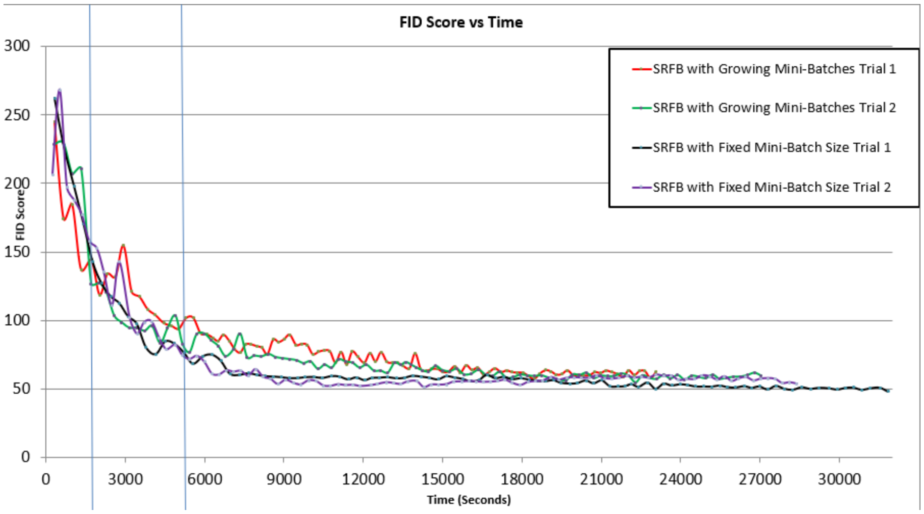


Figure 4.3: A plot of the results of GAN training using the SRFB algorithm. All four trials initially use a mini-batch size of 64 samples, but in the case of the growing mini-batch trials, the mini-batch size grows to 128 samples after 32 epochs, as marked by the first blue line. The second blue line corresponds to the second mini-batch size growing stage, after 96 epochs, when the mini-batches are grown to 256 samples.

When comparing fixed mini-batches with growing mini-batches in terms of computation time, there is a clear difference between them, as is seen in Figure 4.1. The growing mini-batch methods are faster to cover the 640 epochs over the dataset (as is expected with a larger mini-batch size due to less computations of the pseudogradient being required per epoch). The fixed mini-batch GAN performs better than the growing mini-batch on average, obtaining an average FID score of 55 after 12, 590 seconds (3 hours, 29 minutes and 50 seconds) of training which the growing mini-batch GANs only achieve after 19, 766 seconds, (5 hours, 29 minutes and 26 seconds) of training. Looking at 4.3, the first fixed mini-batch trial keeps improving right up to the end of the training process, obtaining an average FID score of 48. Conversely, the second fixed mini-batch trial manages to obtain a relatively low score of 51 after 14, 289 seconds (3 hours, 58 minutes and 9 seconds), but then steadily increases in score afterwards. Comparing the two trials of the growing mini-batch, the first trial is finished considerably earlier compared to the second trial, despite the experimental conditions being identical. This further motivates the influence of stochasticity and variance on the training process. Following these initial results, further testing was performed, as detailed in Section 4.2.

4.1.2 Adjusting the Full Batch Interval for the SVRE Algorithm

In this section, we investigate the effects of changing the full batch interval for a GAN using the SVRE algorithm on the training time and image quality. Similarly to the setup used above in Section 4.1, training is performed for 500, 000 iterations, with a mini-batch size of 64. It should be noted that the extrapolation step involves sampling a second mini-batch, and the full batch is occasionally sampled according to a probability distribution, such that comparatively, at least twice the amount of mini-batches are sampled in the same number of iterations. Since the full batch evaluation comparatively is the most computationally expensive component of the SVRE algorithm, the experiments in this section investigate the impact of adjusting the interval at which the full batch is evaluated. The full batch is evaluated according to a Bernoulli distribution based on the size of the dataset. The CIFAR10 dataset used for these experiments, described in Section 3.3, contains 50, 000 images. Thus, the probability of sampling the full batch during a training iteration is $\frac{64}{50000} \approx \frac{1}{781}$, replicating the experimental setup of Chavdarova et al. [8]. This probability corresponds to a single evaluation of the full batch to be expected per epoch. We explore the impact on training time and FID score when the full batch is sampled half as often, and also when sampled twice as often, and compare this to when the full batch is sampled normally. The same random seed is used for sampling the full batch for all trials for fair comparison. A visualization of the results can be seen in Figure 4.4.

Running a single trial of the SVRE algorithm with the standard likelihood per iteration of evaluating the full batch for 500, 000 iterations takes 28 hours, 32 minutes and 26 seconds. In Figure 4.4, it starts to show signs of convergence after 40, 000 seconds (11 hours), when it has executed 200, 000 iterations of

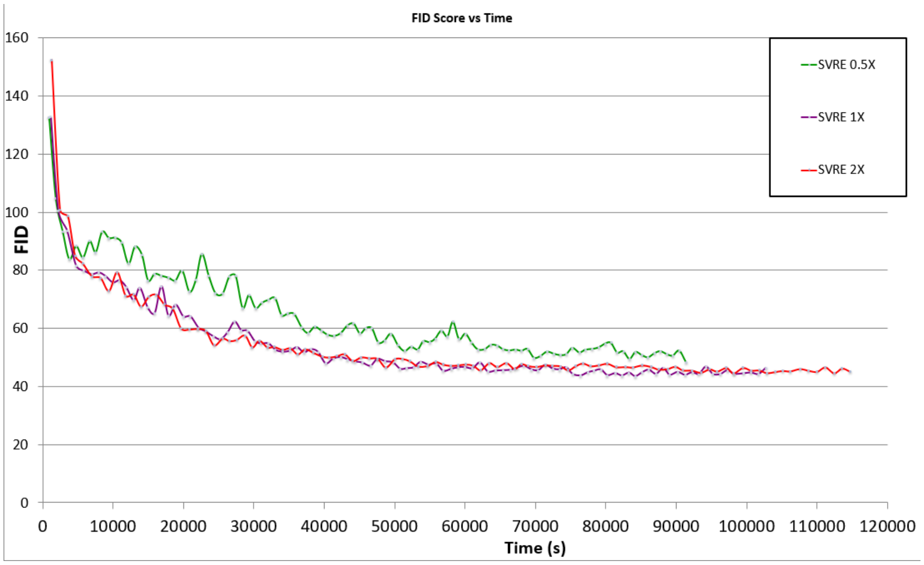


Figure 4.4: A plot of the results of GAN training using the SVRE algorithm. All three trials use a mini-batch size of 64 samples, and differ by the frequency of how often the full batch is evaluated.

training. In total, the full batch is queried 654 times throughout training.
Evaluating the full batch half as often, with a likelihood per iteration of $\frac{1}{1562}$, leads to a small increase in training speed compared to evaluating the full batch normally. This increase in training speed refers to how fast it carries out $500,000$ training iterations, corresponding to 640 epochs. Throughout the training process, it obtains a higher FID score than when the full batch is queried normally, implying the images generated are of lower quality. In total, the full batch is evaluated 353 times.
Evaluating the full batch twice as often, with a likelihood per iteration of $\frac{1}{390}$, leads to a small decrease in training speed compared to evaluating the full batch normally. Throughout the training process, it obtains a similar FID score to when the full batch is queried normally, albeit with a smoother gradient, suggesting more frequent full batch evaluations lead to more stable training. In total, the full batch is evaluated $1,254$ times. The scores obtained suggest there is no major advantage in terms of image quality and training time to be gained by adjusting the frequency of the full batch evaluations per training iteration compared to using the default frequency from the literature.

4.1.3 Comparison Between SRFB and SVRE

As seen in Figure 4.5, where GAN training using the SRFB approach first reaches an average FID of 53 after $15,329$ seconds (corresponding to 4 hours, 15 minutes and 29 seconds) of training, it takes the GAN employing the SVRE

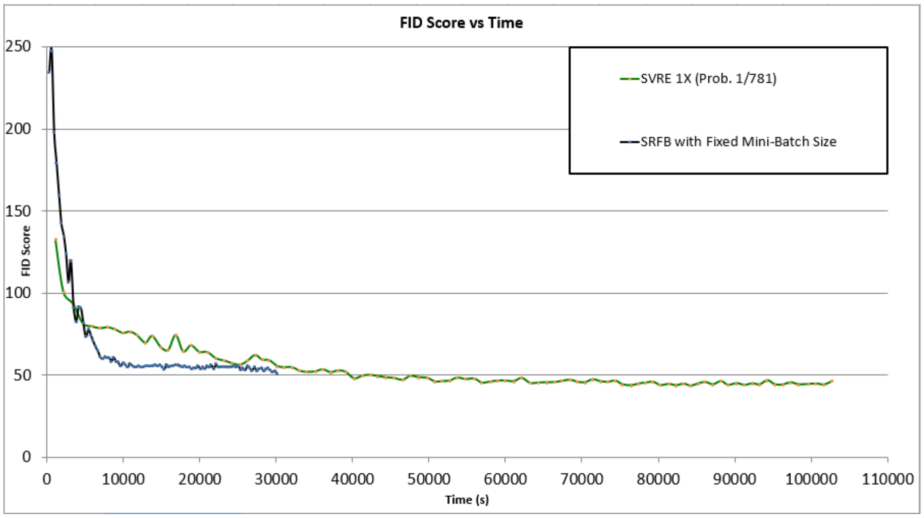


Figure 4.5: A plot of the results of GAN training using the SVRE and SRFB algorithms with regards to FID over training time. Both use a mini-batch size of 64 samples and are trained for 500, 000 iterations.

algorithm 33, 072 seconds (over 9 hours) before it first reaches this level of generated image quality. While eventually the SVRE method does obtain a marginally lower, and thus better, FID score of 46 compared to the SRFB method's FID average score of 50, it takes a lot longer before this is achieved. Comparing the two methods based on FID scores obtained per iteration, as seen in Figure 4.6, shows the impact of the extra calculations performed in each iteration of the SVRE algorithm. Comparing the two methods based on FID score obtained during training based on iterations executed, the SVRE method manages to obtain this marginally lower FID score in fewer iterations compared to the SRFB method.

4.2 Investigating the Role of Growing Mini-Batches on GAN Training

In order to investigate the influence of growing mini-batch sizes on GAN training using the SRFB approach, a comparison is made between the growing of mini-batches at specific intervals throughout the training process compared to GAN training using a fixed mini-batch size. The trials involve the growing of the mini-batches starting from various sizes (16, 32, and 64 batches) doubling these based on the number of epochs observed. The motivation behind choosing relatively small initial mini-batch sizes is to be able to observe the differences between GAN training using relatively small and large sized mini-batches. When to double the mini-batch size is heuristically chosen to occur throughout the

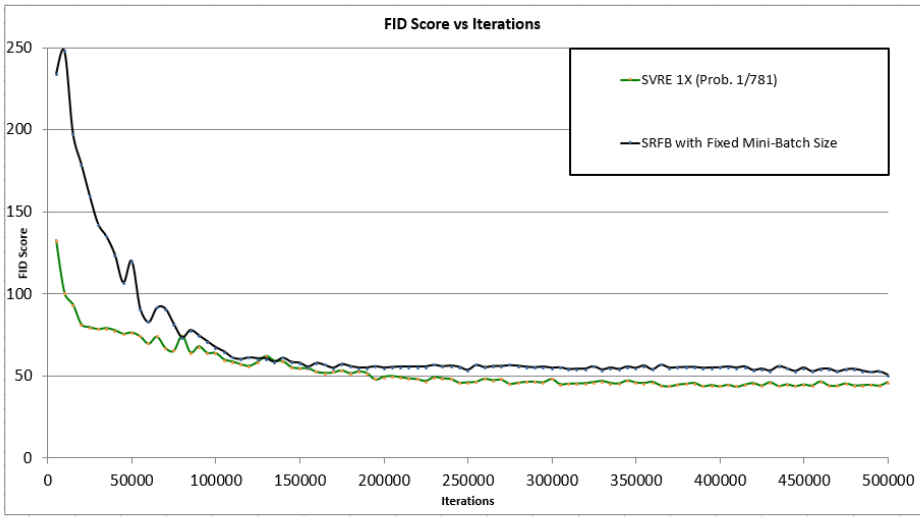


Figure 4.6: A plot of the results of GAN training using the SVRE and SRFB algorithms compared in terms of FID score over training iterations. Both use a fixed mini-batch size of 64 samples and are trained for 500, 000 iterations.

training process. The following plots are each of the FID score relative to time, seen in Figures 4.7, 4.8, and 4.9. The trends in these plots show similar results for each of the three different starting mini-batch sizes compared to their respective fixed mini-batch size counterparts. In each result, the growing mini-batch trial manages be slightly faster compared to the fixed-size mini-batch trial, but obtains a higher FID score throughout the training process, consistent with the findings from the initial investigation into growing mini-batches described in Section 4.1. Also consistent with the initial investigation is the presence of stochastic behavior in the results, such that the initial results in the plot preceding the first mini-batch doubling is worse for the growing mini-batch trials compared to the fixed-batch trials. This occurs despite the experimental set-ups being identical until the first mini-batch doubling.

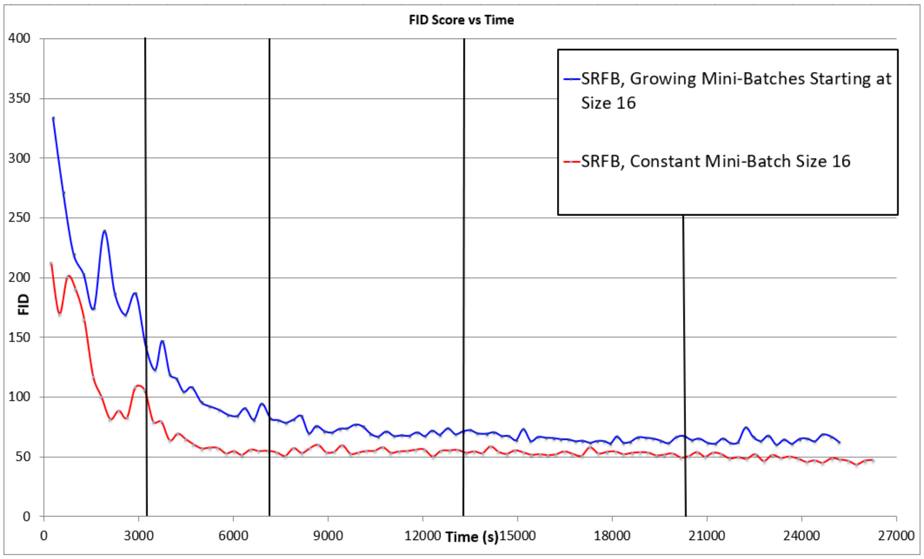


Figure 4.7: A plot of the results of GAN training using the SRFB algorithm, starting with a mini-batch size of 16. Both trials are trained for 160 epochs, with the mini-batch size growing from 16 to 32 after 16 epochs, 32 to 64 after 40 epochs, 64 to 128 after 80 epochs, and 128 to 256 after 128 epochs.

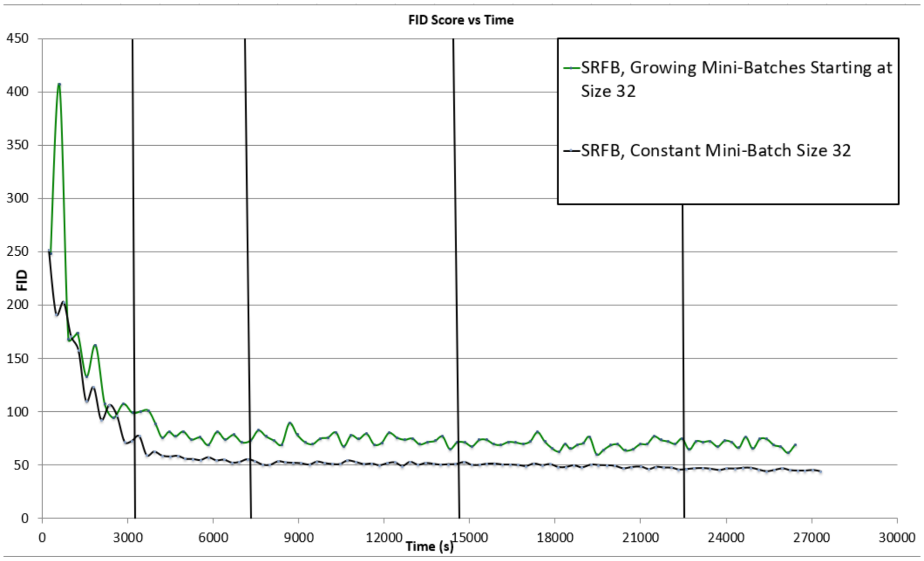


Figure 4.8: A plot of the results of GAN training using the SRFB algorithm, starting with a mini-batch size of 32. Both trials are trained for 320 epochs, with the mini-batch size growing from 32 to 64 after 32 epochs, 64 to 128 after 80 epochs, 128 to 256 after 160 epochs, and 256 to 512 after 256 epochs.

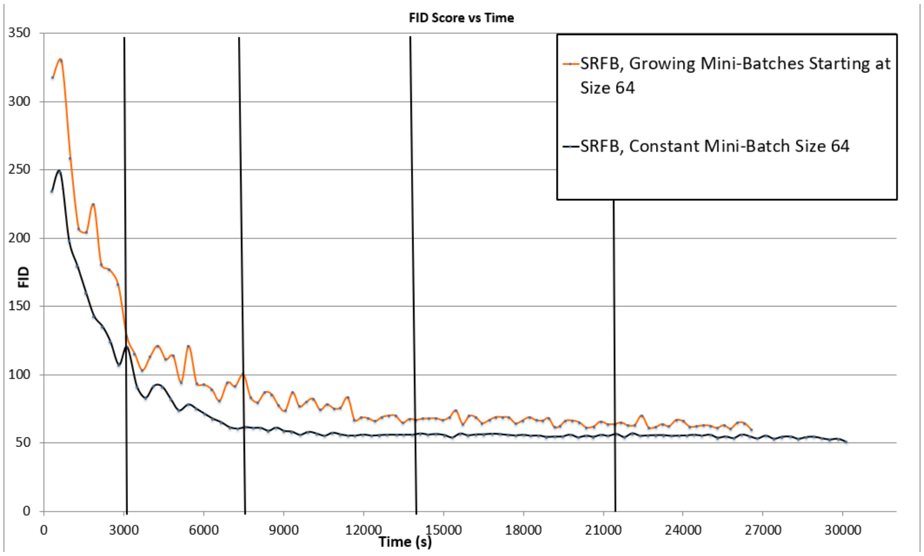


Figure 4.9: A plot of the results of GAN training using the SRFB algorithm, starting with a mini-batch size of 64. Both trials are trained for 640 epochs, with the mini-batch size growing from 64 to 128 after 64 epochs, 128 to 256 after 160 epochs, from 256 to 512 after 320 epochs, and 512 to 1024 after 512 epochs.

Chapter 5

Discussion

This chapter interprets and discusses the results with respect to the research questions posed in Chapter 1, addresses the limitations of the work, and describes suggestions for future work. Summarizing the results, it has been found that the SRFB method is notably faster to converge compared to the SVRE method, that adjusting the full batch interval for the SVRE method does not lead to significant improvements with regards to training time or FID score. It has also been found that growing mini-batches during training will lead to quicker covering of the number of epochs specified for training, but that performance over time is not improved compared to an approach using fixed mini-batch sizes. These results are discussed in detail below.

5.1 Research Question 1

The first research question involves comparing the SRFB and SVRE strategies in terms of their computational expense for GAN training. Compared in terms of FID score obtained over time, a GAN using the SRFB approach is a lot quicker to converge than a GAN using the SVRE strategy, achieving an average FID score of 58 in less than 2.5 hours, which the SVRE approach only achieves after almost 7 hours of training. It finally obtains a relatively low average FID score value of 50 after $8 hours$, 3 hours earlier than when using SVRE.
Investigations into optimizing SVRE for computational complexity by changing the full batch interval shows that changing this interval is not likely to provide an advantage in this regard. The SVRE GAN does not yield significantly improved results (in terms of FID) in a shorter time, nor does it allow for significantly better results (in terms of FID) when allowed to train for a longer training period.

5.1.1 Comparison with Respect to Iteration Complexity

Purely based on iteration complexity, where a single interval corresponds to a single updating of the network parameters, SVRE obtains lower FID scores in fewer iterations than the SRFB method. For example, it achieves an FID score of 48 after 249.6 epochs, which is lower than the best average FID score obtained by the SRFB. This is under the assumption that both the look-ahead step from the extragradient component of SVRE, and the periodic full batch evaluation occur within a single iteration of training.

5.2 Research Question 2

The second research question explores whether progressive growing of mini-batches might lead to a reduction in the computational expenses involved in GAN training using the SRFB method. Despite the parameters initially being identical for both constant mini-batch sizes and growing mini-batch trials (until the growing step occurs), each plot shows the growing mini-batch approach to perform poorly compared to the reference fixed batch approach in terms of FID over time, although it should be noted that this is the case even before the first growing of the mini-batch. A second trial of the growing mini-batch approach (starting at 32 samples per batch, run until 16 epochs) using identical settings yields results which outperform the constant size mini-batch approach for the first 16 epochs, indicating that this cannot be attributed to the growing mini-batch approach itself. This is also the case for a second trial of the fixed size mini-batch approach under identical conditions. Thus, while the constant and growing approaches should on average behave identically until the first doubling of the mini-batch size, this is consistently not the case. As this phenomenon cannot be explained by the existing literature, the results are inconclusive and require further investigation.

5.3 Research Question 3

The third research question investigates the impact of growing mini-batch size using the SRFB method in terms of iteration complexity and as a function of time. Increasing the mini-batch size throughout training yields more training data to be processed per training iteration. Thus, increasing the mini-batch size during training leads to less parameter updates per epoch. While the results suggest that there may be an advantage to using growing mini-batches for more time-efficient GAN training, potentially at the cost of quality of the samples generated, the anomalies encountered in testing means no definitive answer can be given at this time.

5.4 Limitations and Future Work

5.4.1 Fluctuating Stochastic Behavior in Results

The most important limitation of this work is the fluctuating behavior occurring in the results of model quality over time. Prior to the growing of the mini-batch size for GAN training using the SRFB approach, the results of the fixed mini-batch size trials consistently performed better than their growing mini-batch counterparts despite the setup of these models being identical at this stage of training. The most likely cause for this is the stochasticity involved in the training process. It is possible that the random shuffling of the training dataset may have provided the discriminator with mini-batches which gave the generator an advantage in improving its learning performance. It is also possible that the random noise vectors used as inputs to the generator may have provided it with an advantage in sample generation over other noise vectors. Both of these sources of randomness can play a large part in the performance of GANs, and the effects of both can only be mitigated through many repeated trials of the same experimental setup.

A second possible cause of the fluctuations visible in the results when comparing FID score to time is the scheduling of the metrics and the metrics themselves. The FID score is scheduled to occur at every $5,000$ iterations. Since calculating the FID score is a costly operation, with each calculation of the FID score typically requiring several seconds, this is done relatively infrequently. Thus, it is possible that the FID score is calculated for a random input vector which disadvantages the generator, together with an unfavorable mini-batch of samples which, when used for the FID score calculation, leads to a higher FID score indicating worse generated sample quality. The impact of this source of error can also be reduced through more repeated trials.

5.4.2 Mini-Batch Growth

A mini-batch size of 64 is commonly used in the literature, which is why it was chosen for initial testing. When to grow the mini-batches was chosen heuristically in order to observe the differences between mini-batch sizes throughout the training process, and was based on the number of epochs covered and the improvements in FID scores obtained through visual inspection of the fixed size mini-batch results. A future endeavor would be exploring other factors determining when to increase the size of the mini-batch, such as FID score or Inception Score.

5.4.3 Choice of Network Parameters and Architectures

In this thesis, learning rate was not explored explicitly in the investigation of the growing mini-batch sizes due to time constraints. Smith, Kindermans, and Le [25] suggest that increasing the mini-batch size during training while systematically reducing the learning rate can lead to faster training of convolutional neural networks while maintaining adequate performance guarantees. Whether this may also be the case for GANs warrants further investigation.

Chapter 6

Conclusion

Reducing the computational expenses of GAN training is an ongoing challenge. In this thesis, efficient mini-batch evaluation approaches were investigated for GANs employing variance reduction techniques, these being the stochastic relaxed forward backward (SRFB) algorithm and the stochastic variance reduced extragradient (SVRE) algorithm. Comparing the two approaches, it was found that a GAN using SRFB is notably faster to converge compared to SVRE, likely due to SVRE requiring computationally expensive periodic evaluations of the full batch. Compared in terms of iteration complexity, the SVRE manages to generate a higher quality image in less iterations than the SRFB. Adjusting the frequency of the SVRE's full batch evaluation did not lead to significant improvements when compared by the quality of images generated and computing time.

For the SRFB method, the progressive growing of the mini-batch size throughout training was investigated. While typically trials employing growing mini-batch sizes were comparatively quicker to cover the number of epochs defined as the halting condition, they were not faster to converge to a minimum, nor did they manage to achieve a lower (and thus, better) FID score than their fixed-size counterparts. While these patterns are consistently apparent in the findings described in Chapter 4, random behavior visible throughout the training process render this result to be as of yet inconclusive and warranting further investigation.

References

[1] Haskell B Curry. “The method of steepest descent for non-linear minimization problems”. en. In: *Quart. Appl. Math.* 2.3 (1944), pp. 258–261.

[2] Sebastian Ruder. *An overview of gradient descent optimization algorithms.* 2016. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* http://www.deeplearningbook.org. MIT Press, 2016.

[4] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

[5] Mu Li et al. “Efficient Mini-Batch Training for Stochastic Optimization”. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD ’14. New York, New York, USA: Association for Computing Machinery, 2014, pp. 661–670. ISBN: 9781450329569. DOI: 10.1145/2623330.2623612. URL: https://doi.org/10.1145/2623330.2623612.

[6] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).

[7] Tatjana Chavdarova. “Deep Generative Models and Applications”. In: (2020), p. 169. DOI: 10.5075/epfl-thesis-10257. URL: http://infoscience.epfl.ch/record/278463.

[8] Tatjana Chavdarova et al. “Reducing Noise in GAN Training with Variance Reduced Extragradient”. In: *Advances in Neural Information Processing Systems.* Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/58a2fc6ed39fd083f55d4182bf88826d-Paper.pdf.

[9] Tero Karras et al. “Progressive growing of GANs for improved quality, stability, and variation”. In: *arXiv preprint arXiv:1710.10196* (2017).

[10] Chris Donahue, Julian McAuley, and Miller Puckette. *Adversarial Audio Synthesis.* 2018. DOI: 10.48550/ARXIV.1802.04208. URL: https://arxiv.org/abs/1802.04208.

[11] Jiajun Wu et al. “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 82–90.

[12] Tim Salimans et al. “Improved Techniques for Training GANs”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: `https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf`.

[13] Barbara Franci and Sergio Grammatico. “A game–theoretic approach for Generative Adversarial Networks”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 1646–1651. DOI: `10.1109/CDC42340.2020.9304183`.

[14] Lorenzo Rosasco, Silvia Villa, and Bang Công Vũ. “Stochastic forward–backward splitting for monotone inclusions”. In: *Journal of Optimization Theory and Applications* 169.2 (2016), pp. 388–406.

[15] Balamurugan Palaniappan and Francis Bach. “Stochastic Variance Reduction Methods for Saddle-Point Problems”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: `https://proceedings.neurips.cc/paper/2016/file/1aa48fc4880bb0c9b8a3bf979d3b917e-Paper.pdf`.

[16] G.M. Korpelevich. *The extragradient method for finding saddle points and other problems.* 1976.

[17] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks.* 2017. DOI: `10.48550/ARXIV.1701.00160`. URL: `https://arxiv.org/abs/1701.00160`.

[18] Paul Tseng. “A Modified Forward-Backward Splitting Method for Maximal Monotone Mappings”. In: *SIAM Journal on Control and Optimization* 38.2 (2000), pp. 431–446. DOI: `10.1137/S0363012998338806`. eprint: `https://doi.org/10.1137/S0363012998338806`. URL: `https://doi.org/10.1137/S0363012998338806`.

[19] Radu Ioan Boţ et al. “Minibatch Forward-Backward-Forward Methods for Solving Stochastic Variational Inequalities”. In: *Stochastic Systems* 11.2 (2021), pp. 112–139. DOI: `10.1287/stsy.2019.0064`. eprint: `https://doi.org/10.1287/stsy.2019.0064`. URL: `https://doi.org/10.1287/stsy.2019.0064`.

[20] Gauthier Gidel et al. *A Variational Inequality Perspective on Generative Adversarial Networks.* 2019. DOI: `10.48550/ARXIV.1802.10551`. URL: `https://arxiv.org/abs/1802.10551`.

[21] Han Zhang et al. *Self-Attention Generative Adversarial Networks.* 2018. DOI: `10.48550/ARXIV.1805.08318`. URL: `https://arxiv.org/abs/1805.08318`.

[22] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[23] Alex Krizhevsky, Geoffrey Hinton, and Vinod Nair. "Learning Multiple Layers of Features from Tiny Images". MA thesis. University of Toronto, 2009.

[24] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems.* Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

[25] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. "Don't Decay the Learning Rate, Increase the Batch Size". In: *International Conference on Learning Representations.* 2018. URL: https://openreview.net/forum?id=B1Yy1BxCZ.

A Examples of Images Generated Using SRFB with a Fixed Mini-Batch Size of 64

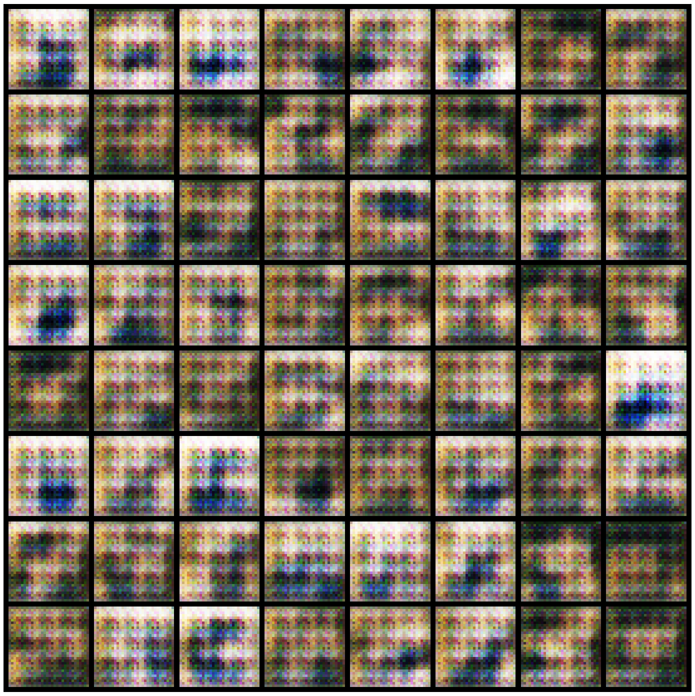


Figure A.1: An image generated after $1,000$ training iterations, 1.28 epochs, 62 seconds of training. These images are very noisy and basically indistinguishable from one another.

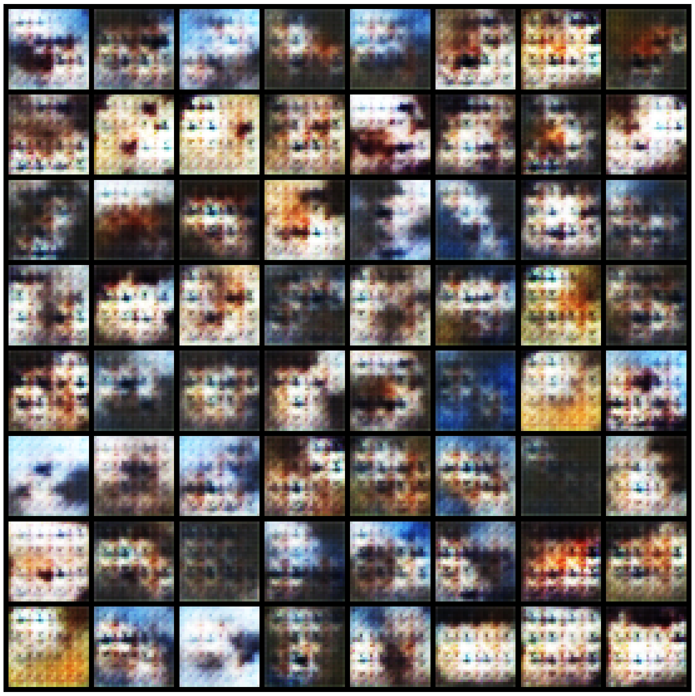


Figure A.2: An image generated after 10, 000 training iterations, 12.8 epochs, 11 minutes of training, obtaining an FID score of 227.508 and an IS of 2.751. The images are still very noisy, but variety is starting to emerge amongst the generated images in terms of basic shades.

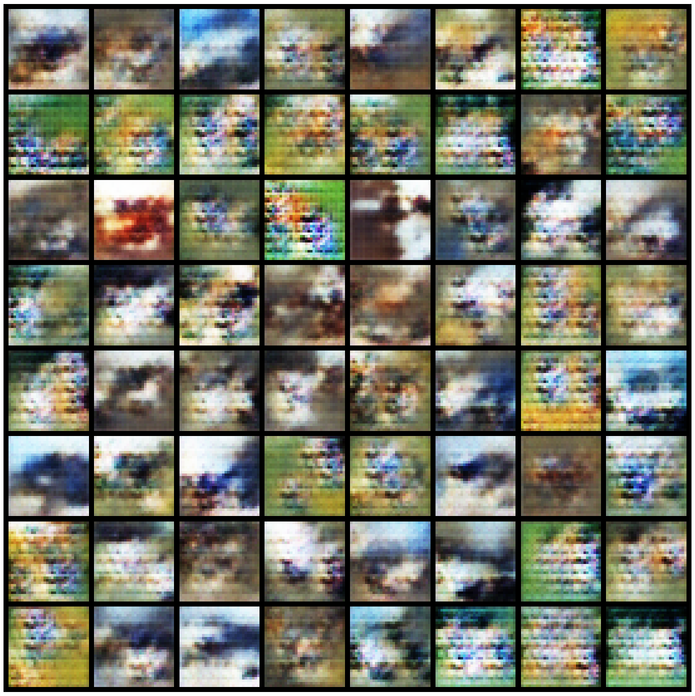


Figure A.3: An image generated after 15, 000 training iterations, 19.2 epochs, 18 minutes of training, obtaining an FID score of 198.308 and an IS of 2.328. Similarly to the image above, the images are still very noisy, but there is some variety. Notice that the FID score has decreased, but so has the IS.

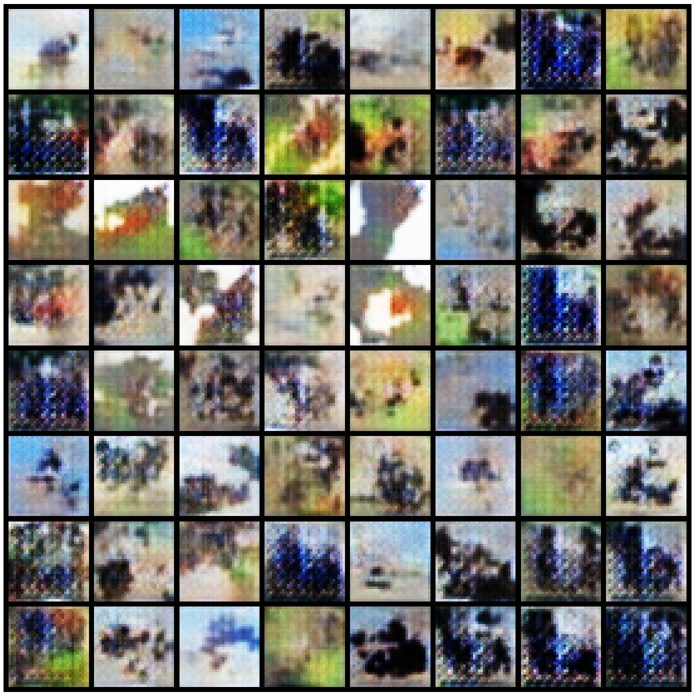


Figure A.4: An image generated after 25, 000 training iterations, 32 epochs, 30 minutes of training, obtaining an FID score of 142.557 and an IS of 3.072. Clearly, the images are still somewhat noisy, but the noise is reduced considerably compared to previous iterations.

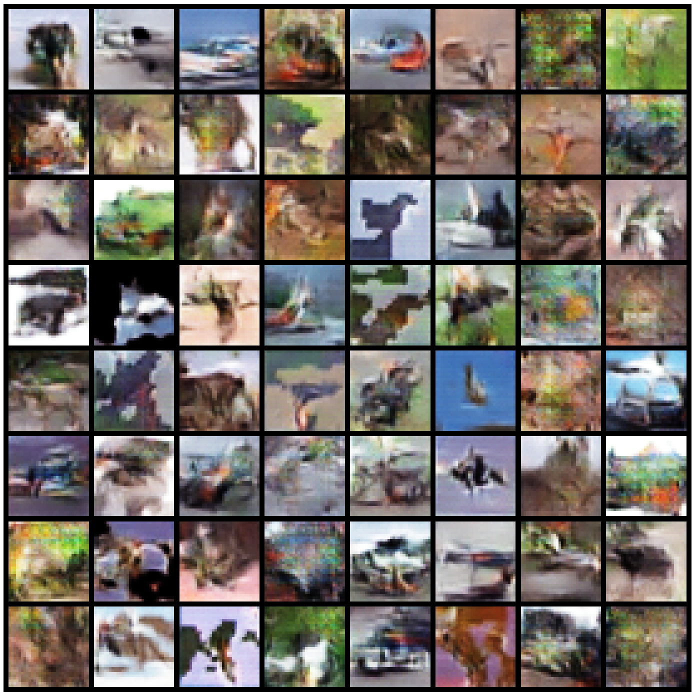


Figure A.5: An image generated after 75, 000 training iterations, 96 epochs, 87 minutes of training, obtaining an FID score of 76.536 and an IS of 4.768. There is still some noise present, but images are starting to become clearer.

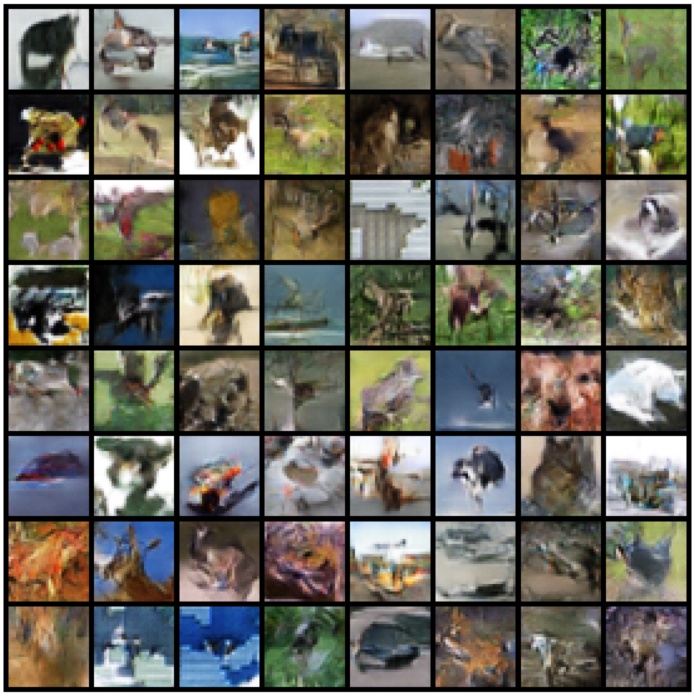


Figure A.6: An image generated after 200, 000 training iterations, 256 epochs, 3 hours and 41 minutes of training, obtaining an FID score of 57.697 and an IS of 5.565. The clearly visible noise is mostly gone from the images, and while the objects in the images are not yet fully formed yet (appearing blurry), it is clear that they are becoming more realistic.

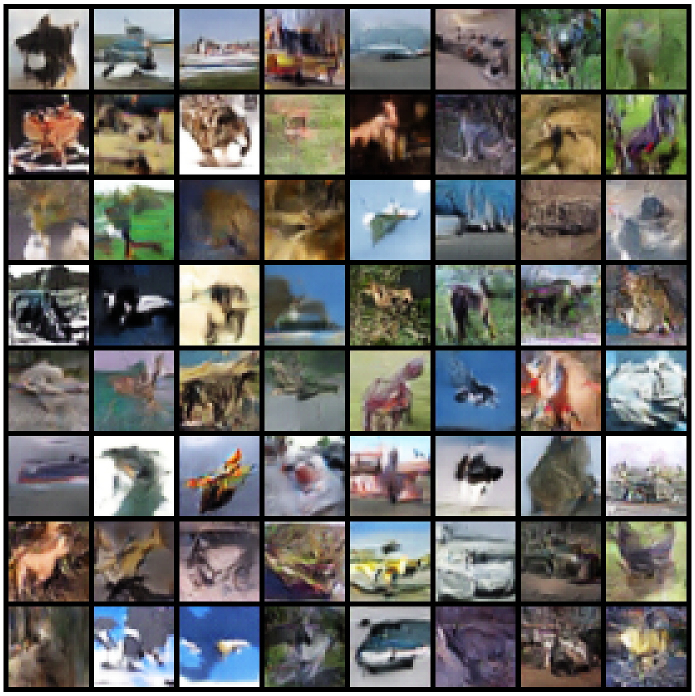


Figure A.7: An image generated after 500,000 training iterations, 640 epochs, 8 hours and 51 minutes of training, obtaining an FID score of 48.007 and an IS of 6.070. At the end of training, the images generated depict objects which, though not clearly distinguishable as specific objects in and of themselves, are clearly distinguishable as objects and from each other. For example, the image in the left top corner depicts an animal with pointy ears. Whether this generated image is a depiction of a horse, a dog, or a cat is not necessarily clear from the image, but it is clearly not an airplane, an automobile, a truck, a ship, or a frog.