RareGAN: Generating Samples for Rare Classes

Zinan Lin, Hao Liang, Giulia Fanti, Vyas Sekar

Carnegie Mellon University
zinanl@andrew.cmu.edu, hl106@rice.edu, gfanti@andrew.cmu.edu, vsekar@andrew.cmu.edu

Abstract

We study the problem of learning generative adversarial networks (GANs) for a rare class of an unlabeled dataset subject to a labeling budget. This problem is motivated from practical applications in domains including security (e.g., synthesizing packets for DNS amplification attacks), systems and networking (e.g., synthesizing workloads that trigger high resource usage), and machine learning (e.g., generating images from a rare class). Existing approaches are unsuitable, either requiring fully-labeled datasets or sacrificing the fidelity of the rare class for that of the common classes. We propose RareGAN, a novel synthesis of three key ideas: (1) extending conditional GANs to use labelled *and* unlabelled data for better generalization; (2) an active learning approach that requests the most useful labels; and (3) a weighted loss function to favor learning the rare class. We show that RareGAN achieves a better fidelity-diversity tradeoff on the rare class than prior work across different applications, budgets, rare class fractions, GAN losses, and architectures[1].

1 Introduction

Many practitioners in diverse domains such as security, networking, and systems require samples from *rare* classes. For example, operators often want to generate queries that force servers to send undesirable responses (Moon et al. 2021), or generate packets that trigger high CPU/memory usage or processing delays for performance evaluation (Petsios et al. 2017).

Prior domain-specific solutions to these problems rely heavily on prior knowledge (e.g., source code) of the systems, which is often unavailable (Lin et al. 2019). Indeed, in response to a recent executive order *Improving the Nation's Cybersecurity*[2], the U.S. National Institute of Standards and Technology published guidance highlighting the importance of creating 'black box' tests for device and software security that do not rely on the implementation or source code of systems (Black, Guttman, and Okun 2021).

Given the success of generative adversarial networks (GANs) (Goodfellow et al. 2014) on data generation, we ask if we can use GANs to generate samples from a rare class (e.g., attack packets, packets that trigger high CPU usage) without requiring prior knowledge about the systems. Note that there are two unique characteristics in our problem:

C1. *High labeling cost.* Labels (whether a sample belongs to the rare class) are often not available a priori, and getting labels is often resource intensive. For example, for a new system, we often do not know a priori which packets will trigger high CPU usage, and evaluating the CPU usage of a packet (for labeling it) can be time consuming.

C2. *Rare class only.* We only need samples from the *rare* class (e.g., attack packets); system operators are often less concerned about *common* class samples (e.g., benign packets).

To the best of our knowledge, no prior GAN paper considers both constraints. Prior related work (see § 3.2) often assumes that the labels are available (failing C1), or tries to generate both rare *and* common samples, which sacrifices the fidelity on the rare class (failing C2). We will see in § 4 that these new characteristics bring unique challenges.

Contributions. We propose RareGAN, a generative model for rare data classes, given an unlabeled dataset and a labeling budget. It combines three ideas: (1) It modifies existing conditional GANs (Odena, Olah, and Shlens 2017) to use both labelled *and* unlabelled data for better generalization. (2) It uses active learning to label samples; we show theoretically that unlike prior work (Xie and Huang 2019), our implementation does not bias the learned rare class distribution. (3) It uses a weighted loss function that favors learning the rare class over the common class; we propose efficient optimization techniques for realizing this reweighting.

We show that RareGAN achieves a better fidelity-diversity tradeoff on the rare class than baselines across different use cases, budgets, rare class fractions, GAN losses, and architectures. Table 1 shows that RareGAN achieves better fidelity

	Budget↓	Fidelity↓	Diversity↑
AmpMAP	14,788,089	16.60	1.68%
RareGAN (ours)	**200,000**	**4.16**	**98.07%**

Table 1: RareGAN achieves better fidelity and diversity with lower budget on DNS amplification attacks than domain-specific techniques. See § 5.1 for the definition of metrics.

[1] This work builds on our previous workshop paper (Section 4.2 of (Lin et al. 2019)).

[2] https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

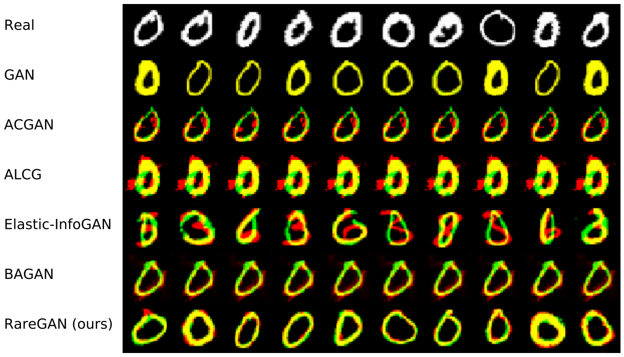


Figure 1: Random generated samples (no cherry-picking) on MNIST class '0' with $B = 1{,}000$ and $\alpha = 1\%$. The red channel plots a generated image, and the green channel the nearest real image from the training set. Yellow pixels show where the two overlap. RareGAN achieves the high sample quality and diversity without memorizing training data.

and diversity (with a smaller labeling budget) when generating DNS amplification attack packets, compared to a state-of-the-art domain-specific technique (Moon et al. 2021). Although RareGAN is primarily motivated from the applications in security, networking, and systems, we also consider image generation, both as a useful tool in its own right and to visualize the improvements. Fig. 1 shows generated samples trained on a modified MNIST handwritten digit dataset (LeCun et al. 1998) where we artificially forcing '0' digit as the rare class (1% of the training data). ACGAN (Odena, Olah, and Shlens 2017), ALCG (Xie and Huang 2019), and BAGAN (Mariani et al. 2018) produces severely mode-collapsed samples. Elastic-InfoGAN (Ojha et al. 2019) produces samples from the wrong class. GAN memorizes the training dataset. RareGAN (bottom) produces high-quality, diverse samples from the correct class without memorizing the training data.

2 Problem Formulation and Use Cases

Problem formulation. We focus on learning a generative model for a "rare" (under-represented) class, subject to two constraints: (1) We assume a limited budget for labeling training data. (2) We only want to learn the rare class distribution, not the common class. More precisely, we are given i.i.d samples $\mathcal{D} = \{x_1, ..., x_n\}$ from a mixture distribution $p = \alpha p_{\mathrm{r}} + (1-\alpha)p_{\mathrm{c}}$, where p_{r} denotes the *rare* class distribution, p_{c} is the *common* class distribution, $\alpha \ll 1$ denotes the weight of the rare class, and we have $\mathrm{Support}\left(p_{\mathrm{r}}\right) \cap \mathrm{Support}\left(p_{\mathrm{c}}\right) = \emptyset$. Each sample x_i has a label $y_i \in \{\mathrm{rare}, \mathrm{common}\}$. The training dataset $\mathcal{D}$ does not include these labels $y_1, ..., y_n$ beforehand, but we can request to label up to B samples during training. Given this budget B, we want to learn a generative model $\hat{p}_{\mathrm{r}}$ that *faithfully* reproduces the rare class distribution, i.e., to minimize $d\left(\hat{p}_{\mathrm{r}}, p_{\mathrm{r}}\right)$, where $d\left(\cdot,\cdot\right)$ is a distance metric between distributions.

This formulation is motivated from the following use cases.

Motivating scenario 1: amplification attacks (security). Many widely deployed public servers and protocols like DNS, NTP, and Memcached are vulnerable to amplification attacks (Moon et al. 2021; Rossow 2014), where the attacker send requests (network packets) to public servers with spoofed source IP addresses, so the response goes to the specified victims. These requests are designed to maximize response size, thus exhausting victims' bandwidth. Server operators want to know which requests trigger these *amplification attacks* to e.g., drop attack requests. Prior solutions require detailed information about the server, such as source code (Rossow 2014), which may be unavailable (Lin et al. 2019).

Our problem formulation. As in (Moon et al. 2021), we treat the *rare class* as all requests with an amplification factor $\text{(size of response packet)}/\text{(size of request packet)} \geq T$, a pre-defined threshold. All other requests belong to the *common class*. To label a request (i.e., check its amplification factor), we send the request through the server, which can be costly. Hence, we want to limit the number of label queries. We want to learn a uniform distribution over high-amplification requests to maximize coverage of the input space.

Motivating scenario 2: performance stress testing (systems & networking). Many deployed systems and networks today rely on black-box components (e.g., lacking source code, detailed specifications). System operators may therefore want to understand worst-case system performance (e.g., CPU/memory usage or delay in the presence of congestion) and optimize for such scenarios (Pedrosa et al. 2018). However, current tools for generating such workloads often rely on a system's source code (Caballero et al. 2007; Petsios et al. 2017; Pedrosa et al. 2018), which may be unavailable (Lin et al. 2019; Black, Guttman, and Okun 2021).

Our problem formulation. We treat the *rare class* as packets with resource usage (e.g., CPU/memory/processing delay) $\geq T$, a pre-defined threshold. The operator can use the trained $\hat{p}_{\mathrm{r}}$ to synthesize such workloads. For the same reasons as the previous case, we want to limit the number of label queries and learn the rare class faithfully.

Motivating scenario 3: inspecting rare class images (ML). Prior GANs on unbalanced image datasets focus on generating samples from *both* rare and common classes for improving downstream classification accuracy (§ 3.2). However, in some other applications, we may *only* need rare class samples. For example, in federated learning, we may want to inspect the samples from specific client/class slices that have bad accuracy for debugging (Augenstein et al. 2019).

We will see in § 5.2 that RareGAN outperforms baselines across all these very different use cases and data types.

3 Background and Related Work

3.1 Background

Generative Adversarial Networks (GANs). GANs (Goodfellow et al. 2014) are a class of deep generative models that have spurred significant interest in recent years. GANs involve two neural networks: a generator G for mapping a random vector z to a random sample $G(z)$, and a discriminator D for guessing whether the input image is generated or from the real distribution p. The vanilla GAN loss (Goodfellow et al. 2014) is $\min_G \max_D \mathcal{L}^{\text{JS}}_{\text{GAN}}\left(D, \hat{p}; p\right)$, where

$$\mathcal{L}^{\text{JS}}_{\text{GAN}}\left(D, \hat{p}; p\right) = \mathbb{E}_{x \sim p}\left[\log D(x)\right] + \mathbb{E}_{x \sim \hat{p}}\left[\log\left(1 - D(x)\right)\right], \quad (1)$$

and $\hat{p}$ denotes the generated distribution induced by $G(z)$ where z is sampled from a fixed prior distribution p_z (e.g., Gaussian or uniform). It has been shown that under some assumptions, Eq. (1) is equivalent to $\min_G d_{\text{JS}}\left(p, \hat{p}\right)$, where $d_{\text{JS}}\left(\cdot, \cdot\right)$ denotes Jensen-Shannon divergence between the two distributions. Several other distance metrics have later been proposed to improve the stability of training (Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017; Nowozin, Cseke, and Tomioka 2016; Mao et al. 2017). Wasserstein distance $d_{\text{W}}\left(\cdot, \cdot\right)$ is one of the most widely used metrics (Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017). The loss of Wasserstein GAN is $\min_G d_{\text{W}}\left(p, \hat{p}\right) = \min_G \max_{\|D\|_L \leq 1} \mathcal{L}^{\text{W}}_{\text{GAN}}\left(D, \hat{p}; p\right)$, where

$$\mathcal{L}^{\text{W}}_{\text{GAN}}\left(D, \hat{p}; p\right) = \mathbb{E}_{x \sim p}\left[D(x)\right] - \mathbb{E}_{x \sim \hat{p}}\left[D(x)\right] \quad (2)$$

and $\|D\|_L$ denotes the Lipschitz constant of D. RareGAN works well with both of these losses.

Auxiliary Classifier GANs (ACGAN). Conditional GANs (CGANs) (Mirza and Osindero 2014; Odena, Olah, and Shlens 2017) are a variant of GANs that support conditional sampling. Besides z, the generators in CGANs have an additional input c which controls the properties (e.g., category) of the generated sample. For example, in a face image dataset (with male/female labeled), instead of only sampling from the entire face distribution, generators in CGANs could allow us to control whether to generate a male or female by specifying c. Several different techniques have been proposed to train such a conditional generator (Mirza and Osindero 2014; Odena, Olah, and Shlens 2017; Salimans et al. 2016; Mariani et al. 2018). ACGAN (Odena, Olah, and Shlens 2017) is one such widely-used variant (Kong et al. 2019; Xie and Huang 2019; Choi et al. 2018). ACGAN adds a classifier C which discriminates the labels for both generated data and real data. The ACGAN loss function is:

$$\min_C \min_G \max_D \mathcal{L}_{\text{GAN}}\left(D, \hat{p}; p\right) + \mathcal{L}_{\text{classification}}\left(C, \hat{p}_{xl}; p_{xl}\right), \quad (3)$$

where $\mathcal{L}_{\text{GAN}}(D, \hat{p}; p)$ is the regular GAN loss (e.g., Eq. (1) (Kong et al. 2019; Xie and Huang 2019; Choi et al. 2018) or Eq. (2) (hwalsuklee 2018)), except that $\hat{p}$ is induced by $G(z, c)$, where $z \sim p_z$ and $c \sim p_l$, where p_l is the ground truth label distribution. $\mathcal{L}_{\text{classification}}$ is defined by

$$\begin{aligned} &\mathcal{L}_{\text{classification}}\left(C, \hat{p}_{xl}; p_{xl}\right) = \\ &-\mathbb{E}_{(x,c) \sim p_{xl}}\left[\log C(x, c)\right] - \mathbb{E}_{(x,c) \sim \hat{p}_{xl}}\left[\log C(x, c)\right], \quad (4) \end{aligned}$$

where $C(x, c)$ denotes classifier C's probability prediction for class c on input sample x, p_{xl} denotes the real joint distribution of samples and labels, and $\hat{p}_{xl}$ denotes the joint distribution over labels and generated samples in $\hat{p}$. In practice, D and C usually share some layers. Note that in Eq. (4) the classifier is trained to match not only the real data, but also the generated data, as in (Kong et al. 2019; hwalsuklee 2018; Odena, Olah, and Shlens 2017). However, in some other implementations, the second part of loss only applies on G, so that the classifier will not be misled by errors in the generator (Xie and Huang 2019; Choi et al. 2018).

3.2 Related work

Depending on the availability of the labels, prior related works can be classified into fully-labeled, unsupervised, semi-supervised, and self-supervised GANs.

Fully supervised GANs. Prior work has studied how to use GANs (particularly ACGAN) to augment imbalanced, *labeled* datasets, e.g., for downstream classification tasks (Mullick, Datta, and Das 2019; Douzas and Bacao 2018; Ren, Liu, and Liu 2019; Ali-Gombe and Elyan 2019; Mariani et al. 2018; Rangwani, Mopuri, and Radhakrishnan 2021; Asokan and Seelamantula 2020; Yang and Zhou 2021). For example, EWGAN (Ren, Liu, and Liu 2019), MFC-GAN (Ali-Gombe and Elyan 2019), Douzas and Bacao (2018), Wei et al. (2019), and BAGAN (Mariani et al. 2018) all augment the original dataset by generating samples from the minority class with a conditional GAN. Wei et al. (2019) utilizes known *mappings* between images in different classes (e.g., mapping an image of normal colon tissue to precancerous colon tissue). BAGAN (Mariani et al. 2018) instead trains an autoencoder on the entire dataset, learns a Gaussian latent distribution for each class, and uses that as the input noise for each class to the GAN generator. We cannot utilize these approaches because we lack labels.

Unsupervised GANs. Unsupervised GANs (Chen et al. 2016; Ojha et al. 2019; Lin et al. 2020) do not control which factors to learn. Hence, there is no guarantee that they will learn to separate samples along the desired factor and threshold (e.g., classification time of the generated packets).

Semi-supervised GANs. Our proposed approach in § 4.1 is one instance of semi-supervised GANs (Odena 2016; Salimans et al. 2016; Dai et al. 2017; Kumar, Sattigeri, and Fletcher 2017; Haque 2020; Zhou et al. 2018). Other semi-supervised GANs could also be used, like the seminal one (Salimans et al. 2016), which uses a single modified discriminator both to separate fake from real samples (as in classical GANs) and to classify the labels of real data. We choose to use separated classifier (as in ACGAN) as the classifier is not influenced by real/fake objective and therefore provides cleaner signal for our active learning technique in § 4.2. The closest prior work, **ALCG (Xie and Huang 2019)**, is also an instance of semi-supervised GANs. Like us, they are training conditional GANs in an active learning setting. However, their goal is to synthesize high-quality samples from *all* classes, whereas we want to faithfully reproduce only the rare class. We show how this distinction requires different algorithmic designs (§ 4), and leads to poor performance by ALCG on our problems (§ 5).

Self-supervised GANs. Self-supervision has been used in both unsupervised and semi-supervised GANs (Sun, Bhattarai, and Kim 2020; Chen et al. 2019; Ojha et al. 2019). It is also unclear how to apply self-supervised GANs in our problem, as they are most useful when we have some prior understanding of the physical or semantic of a system, but in our problem we are given an arbitrary system whose internal structure is unknown. For example, for disentangling digit types (e.g., 0 v.s. 1), Elastic-InfoGAN (Ojha et al. 2019) applies operations like rotation on images to construct positive pairs for self-supervised loss, as we know these operations do not change digit types. However, it is unclear what corresponding operations should be in our problem due to the 'black box' nature of the systems. For example, in motivating scenario 2 (§ 2), it is unclear what operations on packets would keep the CPU/memory usage of the system.

Our work is also related to prior work on weighted loss and data augmentation for GANs, which we discuss below.

Weighted loss. Our proposed approach involves a weighted loss for GANs (§ 4.3). Although weighted loss for GANs has been proposed in prior work (Zadorozhnyy, Cheng, and Ye 2021), the weighting schemes and the goals are completely different to ours. For example, awGAN (Zadorozhnyy, Cheng, and Ye 2021) adjusts the weights on the real/fake losses, with the goal of balancing the gradient directions of these two losses and making training more stable. Instead, as we see in § 4.3, the proposed RareGAN uses different weights for samples in the rare and the common classes (regardless of whether they are fake or real), with the goal of balancing the learning of the rare and the common classes.

Data augmentation. Data augmentation techniques have been proposed for improving GANs' performance in limited data regimes (Karras et al. 2020; Zhao et al. 2020). However, these techniques usually focus on image datasets and the augmentation operations (e.g., random cropping) do not extend to other domains (e.g., our networking datasets).

4 Approach

As mentioned in § 2 and § 3.2, our problem is distinguished by two factors: (1) We want to learn *only* the rare class distribution; (2) The rare/common labels are *not* available in advance, and we have a fixed labeling budget (can be used online during training).

The most obvious straw-man solution to our problem is to randomly and uniformly draw B samples from the dataset and request their labels. Then, we train a *vanilla GAN* on the packets with label "rare". However, since the rare class could have a very low fraction, the number of training samples will be small and the GAN is likely to overfit to the training dataset and generalize poorly. As in prior work (Xie and Huang 2019; Ren, Liu, and Liu 2019; Ali-Gombe and Elyan 2019; Mariani et al. 2018), we can use conditional GANs like ACGAN (§ 3.1) to incorporate common class samples into training, because they could actually be useful for learning the rare class. For example, in face image datasets, the rare class (e.g., men with long hair) and common classes share same characteristics (i.e., faces).

However, due to the small number of rare samples, ACGAN still has bad fidelity and diversity (§ 5.2). In the following, we progressively discuss the design choices we make in RareGAN to address the challenges, and highlight the differences to ALCG (§ 3.2), the most closely-related work.

4.1 Better distribution learning with unlabeled samples

In the above process, the majority of samples from $\mathcal{D}$ are unlabeled because of the labeling budget. Those samples are not used in training (as in ALCG). However, they contain information about the mixture distribution of rare and common classes, and could therefore help learn the rare class.

Our proposed approach relies on carefully altering the ACGAN loss. Recall that the loss has two parts: a classification loss separating rare and common samples, and a GAN loss evaluating their mixture distribution (§ 3.1). Note that the GAN loss does *not* require labels. We propose a modified ACGAN training that uses labeled samples for the classification loss, and *all* samples for the GAN loss. However, when training the GAN loss, we need to know the *fraction* of rare/common classes in order to feed the condition input to the generator. This can be estimated from the labeled samples The maximum likelihood estimate $\hat{\alpha} = \frac{x}{n}$ has variance $\frac{\alpha(1-\alpha)}{n}$, which is small for reasonable n.

4.2 Improving classifier performance with active learning

Because the rare and common classes are highly imbalanced, the classifier in ACGAN could have a bad accuracy. In classification literature, confidence-based active learning has been widely used for solving this challenge (Li and Sethi 2006; Joshi, Porikli, and Papanikolopoulos 2009; Sivaraman and Trivedi 2010), which expends the labeling budget on samples about which the classifier is least confident.

Inspired by these works, our approach is to divide the training into S stages. At the beginning of each stage, we pass all unlabeled samples through the classifier, and request the labels for the ${}^{B}/_{S}$ samples that have the lowest $\max\{C(x, \text{rare}), C(x, \text{common})\}$, where $C(x, c)$ denotes the classifier's (normalized) output.[3] This sample selection criterion is called "least confidence sampling" in prior literature (Lewis and Catlett 1994). There are other criteria like margin of confidence sampling and entropy-based sampling (Vlachos 2008; Kong et al. 2019; Xie and Huang 2019). Since we have only two classes, they are in fact equivalent.

While ALCG also uses confidence-based active learning, it does so in a diametrically opposite way: they request the labels for the *most certain* samples. These two completely different designs result from having different goals: ALCG aims to generate high-quality images, whereas we aim to faithfully reproduce the rare class distribution. The most certain samples usually have better image quality, and therefore ALCG wants to include them in the training. The least certain samples are more informative for learning distribution boundaries, and therefore we label them.

Relation to § 4.1. Naively using active learning is actually counterproductive; if we only use labeled samples in the training (as is done in ALCG), the learned rare distribution will be biased, which partially explains ALCG's poor performance in our setting (§ 5). As discussed in § 4.1, we instead use all unlabeled samples for training the GAN loss. The following proposition explains how this circumvents the problem (proof in App. A).

Proposition 1 *The optimization*

$$p^* \in \arg\min_{\hat{p}} \min_{C\in\mathcal{C}} d\left(\hat{p}, p\right) + \mathcal{L}_{\mathit{classification}}\left(C, \hat{p}_{xl}; p'_{xl}\right)$$

satisfies $d\left(p^*_r, p_r\right) = 0$*, where: (a)* p^*_r *is* p^* *under condition "rare", (b)* p'_{xl} *is* ***any*** *joint (sample, label) distribution where*

[3] In the first stage, the samples for labeling are randomly chosen from the dataset.

the support of p'_{xl} covers the entire sample space, (c) $\hat{p}_{xl}$ is the generated joint distribution of samples and labels, and (d) $\mathcal{C}$ is the set of measurable functions.

The above optimization is a generalization of the ACGAN loss function, where $d(\cdot,\cdot)$ denotes an appropriately-chosen distance function for the GAN loss (e.g., Wasserstein-1). Note that the classification loss is computed over the (possibly biased) distribution induced by active learning, p', whereas the GAN loss is computed with respect to the true distribution p, which uses all samples, labeled or not. The proposition is saying that even though we use the biased p' for the classification loss, we can still learn p_r. On the other hand, if we had used p' for the GAN loss (as is done in ALCG), we would recover a biased version of p_r.

4.3 Better rare class learning with weighted loss

Because the rare class has low mass, errors in the rare distribution have only bounded effect on the GAN loss in Eq. (3). Next, we propose a re-weighting technique for reducing this effect at the expense of learning the common class. Let $\hat{p}$ be the learned sample mixture distribution (without labels). Let $\hat{\alpha} = \int_{\text{Support}(p_r)} \hat{p}(x)dx$ be the fraction of rare samples under $\hat{p}$, and let $\hat{p}'_{\text{r}}$ be $\hat{p}$ restricted to and normalized over $\text{Support}\left(p_{\text{r}}\right)$. (Recall that $\hat{p}_{\text{r}}$ is the generated distribution under condition y =“rare”; it need not be the case that $\text{Support}\left(p_{\text{r}}\right) = \text{Support}\left(\hat{p}_{\text{r}}\right)$.) Similarly, let $\hat{p}'_{\text{c}}$ be $\hat{p}$ over $\mathcal{X} - \text{Support}\left(p_{\text{r}}\right)$ where $\mathcal{X}$ is the entire sample space. Recall that the original GAN loss in Eq. (3) tries to minimize

$$d\left(p,\hat{p}\right) = d\left(\alpha\cdot p_{\text{r}} + (1-\alpha)\cdot p_{\text{c}},\ \hat{\alpha}\cdot\hat{p}'_{\text{r}} + (1-\hat{\alpha})\cdot\hat{p}'_{\text{c}}\right)$$

where d is d_{JS} or d_{W}. We propose to modify this objective function to instead minimize

$$d\left(w\alpha\cdot p_{\text{r}} + \cdot(1-w\alpha)\cdot p_{\text{c}},\ \tfrac{w\hat{\alpha}}{s}\cdot\hat{p}'_{\text{r}} + \tfrac{(1-w\alpha)\cdot(1-\hat{\alpha})}{s\cdot(1-\alpha)}\cdot\hat{p}'_{\text{c}}\right), \quad (5)$$

where $w \in (1, 1/\alpha)$ is the additional multiplicative weight to put on the rare class; and $s = w\hat{\alpha} + \frac{1-w\alpha}{1-\alpha}\left(1-\hat{\alpha}\right)$ is the normalization constant, which is 1 when $\alpha = \hat{\alpha}$. It is straightforward to see that Eq. (5)$= 0 \Leftrightarrow d\left(p,\hat{p}\right) = 0$. However, these two objective functions have different effects in training: this modified loss will more heavily penalize errors in the rare distribution. Consider two extremes: (1) When $w = 1$, Eq. (5) is reduced to the original $d\left(p,\hat{p}\right)$, placing no additional emphasis on the rare class; (2) When $w = {}^{1}\!/\!{}_{\alpha}$, Eq. (5) is reduced to $d\left(p_{\text{r}},\hat{p}'_{\text{r}}\right)$, focusing only on the rare class and placing no constraint on the common class. This completely loses the benefit of learning the classes jointly (§ 4). For a $w \in (1, 1/\alpha)$, we can achieve a better trade-off between the information from the common class and the penalty on the error of rare class.

To implement the above idea, we propose to add a multiplicative weight to the loss of both real and generated samples according to their label, i.e. changing Eqs. (1) and (2) to

$$\mathcal{L}^{\text{JS}'}_{\text{GAN}}\left(D,\hat{p};p\right) =$$
$$\mathbb{E}_{x\sim p}\left[W(x)\cdot\log D(x)\right] + \tfrac{1}{s}\cdot\mathbb{E}_{x\sim\hat{p}}\left[W(x)\cdot\log\left(1-D(x)\right)\right] \quad (6)$$

for Jensen-Shannon divergence and

$$\mathcal{L}^{\text{W}'}_{\text{GAN}}\left(D,\hat{p};p\right) =$$
$$\mathbb{E}_{x\sim p}\left[W(x)\cdot D(x)\right] - \tfrac{1}{s}\cdot\mathbb{E}_{x\sim\hat{p}}\left[W(x)\cdot D(x)\right] \quad (7)$$

for Wasserstein distance, where

$$W(x) = \begin{cases} w & \left(x\in\text{Support}\left(p_r\right)\right) \\ \frac{1-w\alpha}{1-\alpha} & \left(x\notin\text{Support}\left(p_r\right)\right) \end{cases}. \quad (8)$$

Using Eq. (6) and Eq. (7) is equivalent to minimizing Eq. (5) for d_{JS} and d_{W} respectively (proof in App. B):

Proposition 2 *For any D, p, and $\hat{p}$, we have*

$$\mathcal{L}^{JS'}_{GAN}\left(D,\hat{p};p\right) = \mathcal{L}^{JS}_{GAN}\left(D,\hat{q};q\right) \quad (9)$$
$$\mathcal{L}^{W'}_{GAN}\left(D,\hat{p};p\right) = \mathcal{L}^{W}_{GAN}\left(D,\hat{q};q\right) \quad (10)$$

where $\hat{q} = \frac{w\hat{\alpha}}{s}\cdot\hat{p}'_{\text{r}} + \frac{(1-w\alpha)\cdot(1-\hat{\alpha})}{s\cdot(1-\alpha)}\cdot\hat{p}'_{\text{c}}$, and $q = w\alpha\cdot p_{\text{r}} + \cdot(1-w\alpha)\cdot p_{\text{c}}$.

Implementing this weighting is nontrivial, however: (1) The above implementation requires the ground truth labels of all real and generated samples for evaluating Eq. (8), and we do not want to waste labeling budget on weight estimation. For this, we use the ACGAN rare/common classifier as a surrogate labeler. Although this classifier is imperfect, weighting real and generated samples according to the same labeler is sufficient to ensure that the optimum is still $d\left(p,\hat{p}\right) = 0$. (2) Evaluating the normalization constant s in Eqs. (6) and (7) requires estimating $\hat{\alpha}$, which is inefficient as $\hat{\alpha}$ changes during training. Empirically, we found that setting $s = 1$ gave good and stable results.

Note that weighted loss has been used in prior classification and GAN literatures (Cui et al. 2019) for learning *all classes* in a balanced way. But here the goal is for learning only the *rare class* better. This difference leads to different weighting formulations.

5 Experiments

We conduct experiments on all three applications in § 1. The code can be found at https://github.com/fjxmlzn/RareGAN.

Use case 1: DNS amplification attacks. DNS is one of the most widely-used protocols in amplification attacks (Rossow 2014). DNS requests that trigger high amplification have been extensively analyzed in the security community (Kambourakis et al. 2007; Anagnostopoulos et al. 2013; Rossow 2014), though most of the those analyses are manual or use tools specifically designed for (DNS) amplification attacks. We show that RareGAN, though designed for a more general set of problems, can also be effectively used for finding amplification attack requests. In this setting, we define the rare class as DNS requests that have $\frac{\text{size of response}}{\text{size of request}} \geq T$, where T is a threshold specified by users. For the request space, we follow the configuration of (Lin et al. 2019; Moon et al. 2021): we let GANs generate 17 fields in the DNS request; for 5 fields among them (*qr*, *opcode*, *rdatatype*, *rdataclass*, and *url*), we provide candidate values; for all other 12 fields, we let GANs explore all possible bits. The entire search space is 3.6×10^{17}. Unlike image datasets where samples from the mixture distribution p are given, here we need to define p. Since our goal is to find all DNS requests with amplification $\geq T$, we define p as a uniform distribution over the search space. More details are in App. C.

Note on ethics: For this experiment, we needed to make many DNS queries. To avoid harming the public DNS resolvers, we set up our own DNS resolvers on Cloudlab (Duplyakin et al. 2019) for the experiments. These experiments did not involve collecting any sensitive data. Such "penetration testing" of services is common practice in the security literature and we followed best practices (Matwyshyn et al. 2010).[4]

Use case 2: packet classification. Network packet classification is a fundamental building block of modern networks. Switches or routers classify incoming packets to determine what action to take (e.g., forward, drop) (Liang et al. 2019). An active research area in networking is to propose classifiers with low inference latency (Chiu et al. 2018; Liang et al. 2019; Soylu, Erdem, and Carus 2020; Rashelbach, Rottenstreich, and Silberstein 2020). We take a recently-proposed packet classifier (Liang et al. 2019) for example, which was designed to optimize classification time and memory footprint. We define the rare class as network packets that have classification time $\geq T$, a threshold specified by users. GANs generate the bits of 5 fields: *source/destination IP*, *source/destination port*, and *protocol*. The search space is 1.0×10^{31}. As before, p is a uniform distribution over the entire search space. To avoid harming network users, we ran all measurements on our own infrastructure rather than active switches. More details are in App. C.

Use case 3: inspecting rare images. Although RareGAN is primarily designed for the above use cases, we also use images for visualizing the improvements. Following the settings of related work (Mariani et al. 2018), we simulate the imbalanced dataset with widely-used datasets: MNIST (LeCun et al. 1998) and CIFAR10 (Krizhevsky 2009). For MNIST, we treat digit 0 as the rare class, and all other digits as the common class. For CIFAR10, we treat airplane as the rare class, and all other images as the common class. In both cases, the default class fraction is 10%. To simulate a smaller rare class, we randomly drop images from the rare class.

5.1 Evaluation Setup

Baselines. To demonstrate the effect of each design choice, we compare all intermediate versions of RareGAN: vanilla GAN (§ 4), ACGAN (§ 4), ACGAN trained with all unlabled samples (§ 4.1), plus active learning (§ 4.2), plus weighted loss (§ 4.3). In all figures and tables, they are called: "GAN", "ACGAN", "RareGAN (no AL)" , "RareGAN" annotated with 1.0, and "RareGAN" annotated with weight (> 1.0), respectively.

We also evaluate representative prior work on three directions: (1) *GANs with active learning*: ALCG (using only labeled samples in training and using the most certain samples for labeling) (Xie and Huang 2019); (2) *GANs for imbalanced datasets*: BAGAN (Mariani et al. 2018); (3) *Unsupervised/Self-supervised GANs*: Elastic-InfoGAN (Ojha et al. 2019). We only evaluate the last two on MNIST, as they only released codes for that. As discussed in § 3.2, we cannot directly apply the last two in our problem; we make minimal modifications to make them suitable (see App. C).

Architectures. All baselines and RareGAN use the same network architectures. For the first two applications, the generators and discriminators are MLPs. The GAN loss is Wasserstein distance (Eq. (2)), as it is known to be more stable than Jensen-Shannon divergence on categorical variables (Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017). For the image datasets, we follow the popular public ACGAN implementation (hwalsuklee 2018), where the generator and discriminator are CNNs, and the GAN loss is Jensen-Shannon divergence (Eq. (1)).

Metrics. We aim to minimize the distance between real and generated rare class $d\left(\hat{p}_{\mathrm{r}}, p_{\mathrm{r}}\right)$. In practice, generative models are often evaluated along two axes: *fidelity* and *diversity* (Naeem et al. 2020). Because the data types differ across applications, we use different ways to quantify them:

Network packets (use cases 1, 2). *(1) Fidelity.* Network packets are a high-dimensional list of categorical variables. We lack the ground truth p_{r} to estimate fidelity. Instead, we estimate the true distribution of "scores" in p_{r} (i.e., $\frac{\text{size of response}}{\text{size of request}}$ in DNS amplification attacks, and classification time in packet classifiers). This surrogate distribution is operationally meaningful, e.g., for quantifying mean or maximum security risk. We define h_{r} as the ground truth distribution of this number over the rare class (estimated by drawing random samples from the entire search space, computing their scores, and then filtering out the scores that belong to the rare class), and $\hat{h}_{\mathrm{r}}$ as its corresponding generated distribution. We use $d_{\mathrm{W1}}\left(h_{\mathrm{r}}, \hat{h}_{\mathrm{r}}\right)$ as the fidelity metric, where $d_{\mathrm{W1}}\left(\cdot, \cdot\right)$ denotes Wasserstein-1 distance, as it has a simple, interpretable geometric meaning (integrated absolute error between the 2 CDFs). *(2) Diversity.* When GANs overfit, many generated packets are duplicates. Therefore, we count the fraction of unique rare packets (i.e., those with a threshold score $\geq T$) in a set of 500,000 generated samples as the diversity metric.

Images (use case 3). *(1) Fidelity.* We use widely-used Fréchet Inception Distance (FID) (Heusel et al. 2017) between generated data and real rare data to measure fidelity. *(2) Diversity.* The previous diversity metric is not applicable here, as duplicate images are very rare. Instead, we take a widely used heuristic (Wang, Zhang, and Van De Weijer 2016) to check if GAN overfits to the training data: for each generated image, we find its nearest neighbor (in L2 pixel distance) in the training dataset. We then compute the average of nearest distances among a set of generated samples. Note that these two metrics are not completely decoupled: when GAN overfits severely, FID also detects that. Nonetheless, these metrics are widely used in the literature (Wang, Zhang, and Van De Weijer 2016; Heusel et al. 2017; Arora and Zhang 2017; Shmelkov, Schmid, and Alahari 2018).

5.2 Results

Unless otherwise specified, the default configurations are: the number of stages $S = 2$ (for RareGAN and ALCG), weight $w = 3$ (for RareGAN); in DNS, labeling budget $B =$

[4] Two leading guidelines are *responsible disclosure* and *avoid unintentional harm.* We avoided harming the public Internet by running our experiments in sandboxed environments. Since we only reproduced synthetic (known) attack modes (Moon et al. 2021), we did not need to disclose new vulnerabilities.

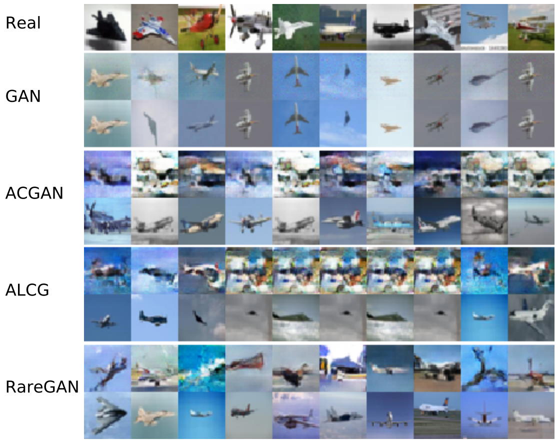


Figure 2: Generated samples (no cherry-picking) on CIFAR10 'airplanes' with B = 10,000 and $\alpha = 10\%$. Each baseline's upper row is generated samples; lower row is the closest real sample.

200,000, rare class fraction $\alpha = 0.776\%$ (corresponding to $T = 10$); in packet classification, B = 200,000, $\alpha = 1.150\%$ (corresponding to $T = 0.055$); in MNIST, $B = 5,000$, $\alpha = 1\%$; in CIFAR10, B = 10,000, $\alpha = 10\%$. Note that the choice of these default configurations do not influence the ranking of different algorithms too much, as we will show in the studies later. All experiments are run over 5 random seeds. More details are in App. C.

Robustness across applications. We start with a qualitative comparison between baselines. We show randomly-generated samples on MNIST and CIFAR10.[5] GAN produces high-quality MNIST images in Fig. 1 by memorizing the labeled rare data (9 samples). Other baselines do not memorize, but either produce mode-collapsed samples (e.g., BAGAN (Mariani et al. 2018)), low-quality *and* mode-collapsed samples (e.g., ACGAN (Odena, Olah, and Shlens 2017), ALCG (Xie and Huang 2019)) or samples from wrong classes (e.g., Elastic-InfoGAN (Ojha et al. 2019)). RareGAN produces samples that are of the same quality as GANs, but with better diversity. On CIFAR10, Fig. 2 shows for each baseline randomly-generated samples (top row) and the closest real samples (bottom row). Again, GAN memorizes the training data, ACGAN and ALCG have poor image quality, and RareGAN trades off between the two (its sample quality is slightly worse than GAN, but its diversity is much better).

Quantitatively, Fig. 3 plots the fidelity-diversity tradeoff of each baseline on our datasets. Lower fidelity scores (left) and higher diversity scores (upwards) are better. The main takeaway of Fig. 3 is that RareGAN has the best tradeoff in our experiments. We discuss each method. *(a) GAN.* In all cases, GANs have poor diversity due to memorization. In network applications, GAN fidelity is good due to overfitting. In the image datasets, FID is bad, as FID scores capture overfitting. *(b) ALCG and BAGAN.* Generally, ALCG and BAGAN has much worse fidelity than other methods, consistent to the qualitative results. *(c) Elastic-InfoGAN.* Elastic-InfoGAN has higher diversity but much worse fidelity (Fig. 3c). Note that the higher diversity metric here is an artifact of Elastic-InfoGAN incorrectly generating digits that are mostly not 0 (Fig. 1), as it is not able to learn the boundary betwween rare and common classes well. *(d) ACGAN.* ACGAN has better diversity metrics and less overfitting than GAN, at the cost of sample fidelity. *(e) Using unlabeled data.* Comparing "RareGAN (no AL)" with "ACGAN", we see that unlabeled data significantly helps the image datasets, but not the network datasets. This may be because of problem dimensionality: the dimension of the images are much larger than the other two cases, so additional data gives a more prominent benefit. *(f) Active learning and weighted loss.* Comparing RareGAN (3.0 and 1.0) with RareGAN (no AL), we see that weighted loss benefits the network packet datasets, but not the image datasets. This could be due to the complexity of the rare class boundary, which is nonsmooth in network applications (Moon et al. 2021).

Due to space limitations, the following parametric studies show plots for a single dataset; we defer the results on the other datasets to the appendices, where we see similar trends.

Robustness to labeling budget B. We decrease B to show how different approaches react in the low-budget setting. The results on MNIST are shown in Fig. 4. All three RareGAN versions are insensitive to budget. For the baselines, the sample qualities of ACGAN, ALCG, and BAGAN degrade significantly, as evidenced by the bad FIDs for small budgets. Elastic-InfoGAN has higher diversity again due to the incorrect generated digits. GANs always have the worst diversity, no matter the budget. Results on other datasets are in App. D; RareGAN generally has the best robustness across budgets.

Robustness to rare class fraction α. In Fig. 5, we vary α to measure the effect of class imbalance. All algorithms exhibit worse sample quality when the rare class fraction is decreased. However, for all α, RareGAN has a better fidelity-diversity tradeoff than ACGAN, ALCG, and BAGAN (achieving much better fidelity and similar diversity). Elastic-InfoGAN still has worse fidelity than RareGAN due to wrong generated digits. GANs always have the worst diversity. Results on the other datasets are in App. D, where we see that RareGAN generally has the best robustness to α.

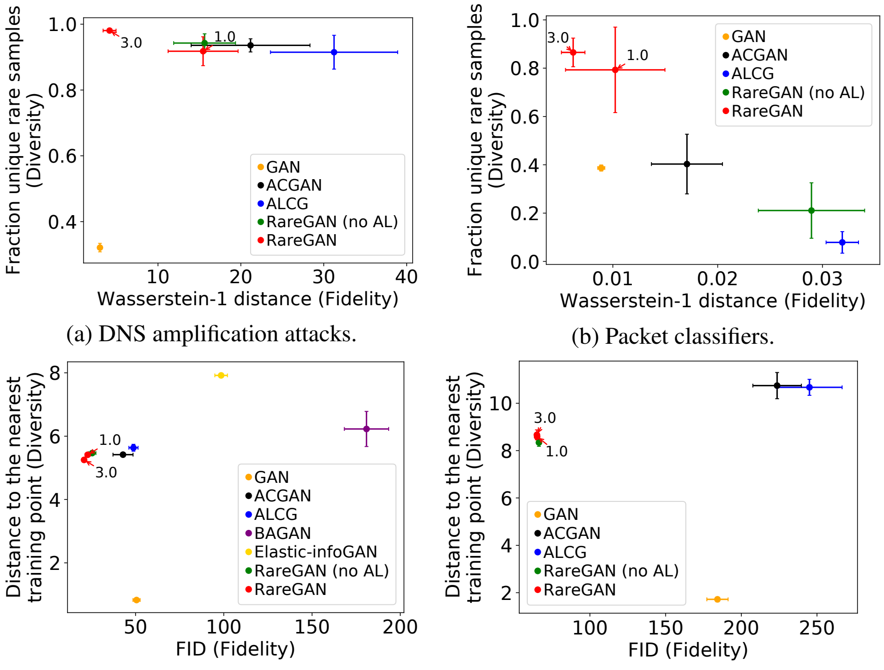
Variance across trials. The standard error bars in Fig. 3 show that ACGAN, ALCG, and BAGAN have high variance across trials, and RareGAN with weighted loss has lower variance. This is because the weighted loss penalizes errors in the rare class, thus providing better stability.

The following ablation studies give additional insights into each tunable component of RareGAN.

Influence of the number of stages S (§ 4.2) and the loss weight w (§ 4.3). We have seen that active learning and weighted loss do not influence the image dataset results much (Figs. 3c, 3d, 4 and 5). Therefore, we focus on DNS in Fig. 14 (App. E). As we increase the weight from $w = 1$ to $w = 5$, both metrics improve, saturating at $w \geq 3$. At the default weight $w = 3.0$, choosing $S = 2$ or $S = 4$ stages makes little difference. Comparing Fig. 14 with Fig. 3a RareGAN improves upon ACGAN and ALCG for almost all S and w.

Ablation study on RareGAN components. RareGAN has

[5] All samples are drawn from the model with the median FID score over 5 runs.



(a) DNS amplification attacks.

(b) Packet classifiers.

(c) MNIST.

(d) CIFAR10.

Figure 3: RareGAN achieves the best fidelity-diversity tradeoff on all applications. Horizontal axis is fidelity (lower is better). Vertical axis is diversity (higher is better). Fidelity/diversity metrics are explained in § 5.1. Bars show standard error over 5 runs.

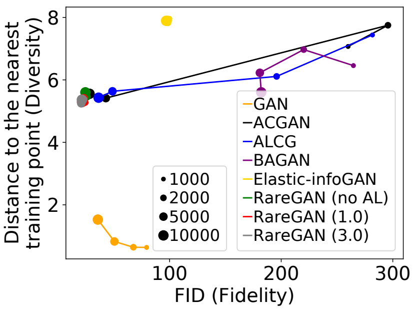


Figure 4: MNIST with different labeling budget B. RareGAN is insensitive to B.

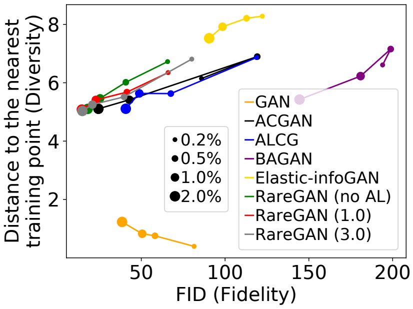


Figure 5: MNIST with varying rare class fraction. RareGAN has the best tradeoff.

three parts: (1) using unlabeled samples (§ 4.1), (2) active learning (§ 4.2), and (3) weighted loss (§ 4.3). Active learning only makes sense with unlabeled samples, so there are 6 possible combinations. Fig. 15 (App. F) shows each variant on the DNS application. Including all components, RareGAN yields the best diversity-fidelity tradeoff and low variance.

Comparison to domain-specific techniques. We compare to AmpMAP, the state-of-the-art work on (DNS) amplification attacks in the security community (Moon et al. 2021), in Table 1. AmpMAP finds high amplification packets by drawing random packets and requesting their amplification factors, and then doing random field perturbation on high amplification packets. AmpMAP uses amplification threshold 10, and the same packet space as ours. Note that AmpMAP is specically designed for amplication attacks, not applicable for other applications we did. Even in that case, our proposed RareGAN still achieves much better fidelity and diversity with a fraction of the budget.

6 Conclusions

We propose RareGAN for generating samples from a rare class subject to a limited labeling budget. We show that RareGAN has good, stable diversity and fidelity in experiments covering different loss functions (e.g., Jensen-Shannon divergence (Goodfellow et al. 2014), Wasserstein distance (Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017)), architectures (e.g., CNN, MLP), data types (e.g., network packets, images), budgets, and rare class fractions.

Acknowledgements

We thank Yucheng Yin for the help on baseline comparison, and Sekar Kulandaivel, Wenyu Wang, Bryan Phee, and Shruti Datta Gupta for their help with earlier versions of RareGAN. This work was supported in part by faculty research awards from Google, JP Morgan Chase, and the Sloan Foundation, as well as gift grants from Cisco and Siemens AG. This research was sponsored in part by National Science Foundation Convergence Accelerator award 2040675 and the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. Zinan Lin acknowledges the support of the Siemens FutureMakers Fellowship, the CMU Presidential Fellowship, and the Cylab Presidential Fellowship. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) (Towns et al. 2014), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges system (Nystrom et al. 2015), which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

References

Ali-Gombe, A.; and Elyan, E. 2019. MFC-GAN: class-imbalanced dataset classification using multiple fake class generative adversarial network. *Neurocomputing*, 361.

Anagnostopoulos, M.; Kambourakis, G.; Kopanos, P.; Louloudakis, G.; and Gritzalis, S. 2013. DNS amplification attack revisited. *Computers & Security*, 39: 475–485.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *ICML*, 214–223. PMLR.

Arora, S.; and Zhang, Y. 2017. Do gans actually learn the distribution? an empirical study. *arXiv preprint arXiv:1706.08224*.

Asokan, S.; and Seelamantula, C. S. 2020. Teaching a gan what not to learn. *arXiv preprint arXiv:2010.15639*.

Augenstein, S.; McMahan, H. B.; Ramage, D.; Ramaswamy, S.; Kairouz, P.; Chen, M.; Mathews, R.; et al. 2019. Generative models for effective ML on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*.

Black, P. E.; Guttman, B.; and Okun, V. 2021. Guidelines on Minimum Standards for Developer Verification of Software. arXiv:2107.12850.

Caballero, J.; Yin, H.; Liang, Z.; and Song, D. 2007. Polyglot: Automatic extraction of protocol message format using dynamic binary analysis. In *Proceedings of the 14th ACM conference on Computer and communications security*.

Chen, T.; Zhai, X.; Ritter, M.; Lucic, M.; and Houlsby, N. 2019. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 12154–12163.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2180–2188.

Chiu, Y.-K.; Ruan, S.-J.; Shen, C.-A.; and Hung, C.-C. 2018. The design and implementation of a latency-aware packet classification for OpenFlow protocol based on FPGA. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, 64–69.

Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; and Choo, J. 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 8789–8797.

Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; and Belongie, S. 2019. Class-balanced loss based on effective number of samples. In *CVPR*, 9268–9277.

Dai, Z.; Yang, Z.; Yang, F.; Cohen, W. W.; and Salakhutdinov, R. 2017. Good semi-supervised learning that requires a bad gan. *arXiv preprint arXiv:1705.09783*.

Douzas, G.; and Bacao, F. 2018. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with applications*, 91: 464–471.

Duplyakin, D.; Ricci, R.; Maricq, A.; Wong, G.; Duerig, J.; Eide, E.; Stoller, L.; Hibler, M.; Johnson, D.; Webb, K.; et al. 2019. The design and operation of CloudLab. In *2019 USENIX Annual Technical Conference*, 1–14.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved training of stein gans. *arXiv preprint arXiv:1704.00028*.

Haque, A. 2020. EC-GAN: Low-Sample Classification using Semi-Supervised Algorithms and GANs. *arXiv preprint arXiv:2012.15864*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*.

hwalsuklee. 2018. tensorflow-generative-model-collections. https://github.com/hwalsuklee/tensorflow-generative-model-collections.

Joshi, A. J.; Porikli, F.; and Papanikolopoulos, N. 2009. Multi-class active learning for image classification. In *CVPR*, 2372–2379. IEEE.

Kambourakis, G.; Moschos, T.; Geneiatakis, D.; and Gritzalis, S. 2007. A fair solution to dns amplification attacks. In *Second International Workshop on Digital Forensics and Incident Analysis (WDFIA 2007)*, 38–47. IEEE.

Karras, T.; Aittala, M.; Hellsten, J.; Laine, S.; Lehtinen, J.; and Aila, T. 2020. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*.

Kong, Q.; Tong, B.; Klinkigt, M.; Watanabe, Y.; Akira, N.; and Murakami, T. 2019. Active generative adversarial network for image classification. In *AAAI*, volume 33, 4090–4097.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images.

Kumar, A.; Sattigeri, P.; and Fletcher, T. 2017. Semi-supervised learning with gans: Manifold invariance with improved inference. *Advances in Neural Information Processing Systems*, 30.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lewis, D. D.; and Catlett, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, 148–156. Elsevier.

Li, M.; and Sethi, I. K. 2006. Confidence-based active learning. *IEEE transactions on pattern analysis and machine intelligence*, 28(8): 1251–1261.

Liang, E.; Zhu, H.; Jin, X.; and Stoica, I. 2019. Neural packet classification. In *Proceedings of the ACM Special Interest Group on Data Communication*, 256–269.

Lin, Z.; Moon, S.-J.; Zarate, C. M.; Mulagalapalli, R.; Kulandaivel, S.; Fanti, G.; and Sekar, V. 2019. Towards oblivious network analysis using generative adversarial networks. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, 43–51.

Lin, Z.; Thekumparampil, K.; Fanti, G.; and Oh, S. 2020. Infogan-cr and modelcentrality: Self-supervised model training and selection for disentangling gans. In *International Conference on Machine Learning*, 6127–6139. PMLR.

Mao, X.; Li, Q.; Xie, H.; Lau, R. Y.; Wang, Z.; and Paul Smolley, S. 2017. Least squares generative adversarial networks. In *ICCV*, 2794–2802.

Mariani, G.; Scheidegger, F.; Istrate, R.; Bekas, C.; and Malossi, C. 2018. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*.

Matwyshyn, A. M.; Cui, A.; Keromytis, A. D.; and Stolfo, S. J. 2010. Ethics in Security Vulnerability Research. In *IEEE Security & Privacy*.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Moon, S.-J.; Yin, Y.; Sharma, R. A.; Yuan, Y.; Spring, J. M.; and Sekar, V. 2021. Accurately measuring global risk of amplification attacks using ampmap. In *30th USENIX Security Symposium*.

Mullick, S. S.; Datta, S.; and Das, S. 2019. Generative adversarial minority oversampling. In *ICCV*, 1695–1704.

Naeem, M. F.; Oh, S. J.; Uh, Y.; Choi, Y.; and Yoo, J. 2020. Reliable fidelity and diversity metrics for generative models. In *ICML*, 7176–7185. PMLR.

Nowozin, S.; Cseke, B.; and Tomioka, R. 2016. f-gan: Training generative neural samplers using variational divergence minimization. In *NeurIPS*.

Nystrom, N. A.; Levine, M. J.; Roskies, R. Z.; and Scott, J. R. 2015. Bridges: A Uniquely Flexible HPC Resource for New Communities and Data Analytics. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, XSEDE '15, 30:1–30:8. New York, NY, USA: ACM. ISBN 978-1-4503-3720-5.

Odena, A. 2016. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*.

Odena, A.; Olah, C.; and Shlens, J. 2017. Conditional image synthesis with auxiliary classifier gans. In *ICML*. PMLR.

Ojha, U.; Singh, K. K.; Hsieh, C.-J.; and Lee, Y. J. 2019. Elastic-InfoGAN: Unsupervised Disentangled Representation Learning in Class-Imbalanced Data. *arXiv preprint arXiv:1910.01112*.

Pedrosa, L.; Iyer, R.; Zaostrovnykh, A.; Fietz, J.; and Argyraki, K. 2018. Automated synthesis of adversarial workloads for network functions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*.

Petsios, T.; Zhao, J.; Keromytis, A. D.; and Jana, S. 2017. Slowfuzz: Automated domain-independent detection of algorithmic complexity vulnerabilities. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2155–2168.

Rangwani, H.; Mopuri, K. R.; and Radhakrishnan, V. B. 2021. Class Balancing GAN with a Classifier in the Loop.

Rashelbach, A.; Rottenstreich, O.; and Silberstein, M. 2020. A Computational Approach to Packet Classification. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 542–556.

Ren, J.; Liu, Y.; and Liu, J. 2019. EWGAN: Entropy-based Wasserstein GAN for imbalanced learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 10011–10012.

Rossow, C. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *NDSS*.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*.

Shmelkov, K.; Schmid, C.; and Alahari, K. 2018. How good is my GAN? In *ECCV*, 213–229.

Sivaraman, S.; and Trivedi, M. M. 2010. A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 11(2): 267–276.

Soylu, T.; Erdem, O.; and Carus, A. 2020. Bit vector-coded simple CART structure for low latency traffic classification on FPGAs. *Computer Networks*, 167: 106977.

Sun, J.; Bhattarai, B.; and Kim, T.-K. 2020. MatchGAN: a self-supervised semi-supervised conditional generative adversarial network. In *Proceedings of the Asian Conference on Computer Vision*.

Towns, J.; Cockerill, T.; Dahan, M.; Foster, I.; Gaither, K.; Grimshaw, A.; Hazlewood, V.; Lathrop, S.; Lifka, D.; Peterson, G. D.; Roskies, R.; Scott, J. R.; and Wilkins-Diehr, N. 2014. XSEDE: Accelerating Scientific Discovery. *Computing in Science Engineering*, 16(5): 62–74.

Vlachos, A. 2008. A stopping criterion for active learning. *Computer Speech & Language*, 22(3): 295–312.

Wang, Y.; Zhang, L.; and Van De Weijer, J. 2016. Ensembles of generative adversarial networks. *arXiv preprint arXiv:1612.00991*.

Wei, J.; Suriawinata, A.; Vaickus, L.; Ren, B.; Liu, X.; Wei, J.; and Hassanpour, S. 2019. Generative image translation for data augmentation in colorectal histopathology images. *arXiv preprint arXiv:1910.05827*.

Xie, M.-K.; and Huang, S.-J. 2019. Learning class-conditional gans with active sampling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 998–1006.

Yang, H.; and Zhou, Y. 2021. IDA-GAN: A Novel Imbalanced Data Augmentation GAN. In *ICPR*. IEEE.

Zadorozhnyy, V.; Cheng, Q.; and Ye, Q. 2021. Adaptive Weighted Discriminator for Training Generative Adversarial Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4781–4790.

Zhao, S.; Liu, Z.; Lin, J.; Zhu, J.-Y.; and Han, S. 2020. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*.

Zhou, T.; Liu, W.; Zhou, C.; and Chen, L. 2018. Gan-based semi-supervised for imbalanced data classification. In *ICIM*. IEEE.

Appendix

A Proof of Prop. 1

Let p_c^* be p^* under condition "common". Note that the objective function in Prop. 1 is the ACGAN loss, which has two parts: the GAN loss and the classifier loss; recall the ACGAN classifier loss in Eq. (4).

First, suppose that $d(p_c^*, p_\mathrm{c}) = 0$ and $d(p_r^*, p_\mathrm{r}) = 0$. It is clear that in this case, p^* optimizes the GAN loss $d(p^*, p)$. Since $\mathcal{C}$ contains all measurable functions and since $\text{Support}\,(p_\mathrm{r}) \cap \text{Support}\,(p_\mathrm{c}) = \emptyset$, there exists a classifier $C \in \mathcal{C}$ that can separate the rare class p_r from the common one p_c. Since $p_r^* = p_\mathrm{r}$ and $p_c^* = p_\mathrm{c}$, the same C also has 0 loss on p^*, so the classifier loss is 0. Hence we have that p^* optimizes both parts of the AGCAN loss.

Now suppose that either $d(p_c^*, p_\mathrm{c}) > 0$ or $d(p_r^*, p_\mathrm{r}) > 0$. Then we have the following possibilities:

(Case 1) $\text{Support}\,(p_c^*) - \text{Support}\,(p_c) \neq \emptyset$ or $\text{Support}\,(p_r^*) - \text{Support}\,(p_r) \neq \emptyset$. In this case, we know the classification loss cannot achieve the optimal value (namely 0). Since either p_c^* contains elements not in p_c or p_r^* contains elements not in p_r, any classifier C must either have nonzero error on the rare or the common class. Hence, the ACGAN classifier loss will be nonzero.

(Case 2) $\text{Support}\,(p_c^*) \subseteq \text{Support}\,(p_c)$ and $\text{Support}\,(p_r^*) \subseteq \text{Support}\,(p_r)$. In this case, because $d(p_c^*, p_\mathrm{c}) > 0$ and/or $d(p_r^*, p_\mathrm{r}) > 0$, the GAN loss cannot achieve its optimal value of 0.

Therefore, we can conclude that for any $p^* \in \arg\min_{\hat{p}} \min_C d\,(\hat{p}, p) + \mathcal{L}_{\text{classification}}\,(C, \hat{p}_{xl}; p'_{xl})$, we have $d\,(p_r^*, p_r) = 0$.

Comments. The key here is the assumption $\text{Support}\,(p_\mathrm{r}) \cap \text{Support}\,(p_\mathrm{c}) = \emptyset$ (stated in problem statement § 1). If this is not satisfied, the bias in the labeled dataset (because of active learning) will influence the final solution through the classification loss. To be more specific, even if p^* satisfies $d(p_c^*, p_\mathrm{c}) = 0$ and $d(p_r^*, p_\mathrm{r}) = 0$, it does not necessarily optimize the classification loss. Therefore, the optimal solution could be moved away from this p^*.

B Proof of Prop. 2

It suffices to show that

$$\mathbb{E}_{x\sim p}\left[W(x)\cdot\log D(x)\right] = \mathbb{E}_{x\sim q}\left[\log D(x)\right] \tag{11}$$

$$\frac{1}{s}\cdot\mathbb{E}_{x\sim\hat{p}}\left[W(x)\cdot\log\left(1-D(x)\right)\right] = \mathbb{E}_{x\sim\hat{q}}\left[\log\left(1-D(x)\right)\right] \tag{12}$$

$$\mathbb{E}_{x\sim p}\left[W(x)\cdot D(x)\right] = \mathbb{E}_{x\sim q}\left[D(x)\right] \tag{13}$$

$$\frac{1}{s}\cdot\mathbb{E}_{x\sim\hat{p}}\left[W(x)\cdot D(x)\right] = \mathbb{E}_{x\sim\hat{q}}\left[D(x)\right] \tag{14}$$

We start from Eq. (13).

$$\begin{aligned}
&\mathbb{E}_{x\sim p}\left[W(x)\cdot D(x)\right]\\
=&\int p(x)W(x)D(x)dx\\
=&\int_{\text{Support}(p_\mathrm{r})}\alpha w\cdot p_\mathrm{r}(x)D(x)dx + \int_{\text{Support}(p_\mathrm{c})}(1-\alpha)\frac{1-w\alpha}{1-\alpha}\cdot p_\mathrm{c}\,(x)\,D(x)dx\\
=&\int_{\text{Support}(p_\mathrm{r})}\alpha w\cdot p_\mathrm{r}(x)D(x)dx + \int_{\text{Support}(p_\mathrm{c})}(1-w\alpha)\cdot p_\mathrm{c}\,(x)\,D(x)dx\\
=&\quad\mathbb{E}_{x\sim q}\left[W(x)\cdot D(x)\right]
\end{aligned}$$

The proof for Eq. (11) is similar except changing $D(x)$ to $\log D(x)$.

Now we look at Eq. (14).

$$\begin{aligned}
&\frac{1}{s}\cdot\mathbb{E}_{x\sim\hat{p}}\left[W(x)\cdot D(x)\right]\\
=&\,\frac{1}{s}\left(\int\hat{p}(x)W(x)D(x)dx\right)\\
=&\,\frac{1}{s}\left(\int_{\text{Support}(p_\mathrm{r})}\hat{\alpha}w\cdot\hat{p}_\mathrm{r}(x)D(x)dx + \int_{X-\text{Support}(p_\mathrm{r})}(1-\hat{\alpha})\frac{1-w\alpha}{1-\alpha}\cdot\hat{p}_\mathrm{c}(x)D(x)dx\right)\\
=&\,\mathbb{E}_{x\sim\hat{q}}\left[D(x)\right]
\end{aligned}$$

The proof for Eq. (12) is similar except changing $D(x)$ to $\log\left(1-D(x)\right)$.

C Experiment details

C.1 DNS amplification attacks

The details of the packet fields are as follows (following (Moon et al. 2021; Lin et al. 2019)).

- id: 16 bits, modeled using 16 2-dim softmax
- opcode: choosing from [0,1,2,3,4,5], modeled using a 5-dim softmax
- aa: 1 bit, modeled using one 2-dim softmax
- tc: 1 bit, modeled using one 2-dim softmax
- rd: 1 bit, modeled using one 2-dim softmax
- ra: 1 bit, modeled using one 2-dim softmax
- z: 1 bit, modeled using one 2-dim softmax
- ad: 1 bit, modeled using one 2-dim softmax
- cd: 1 bit, modeled using one 2-dim softmax
- rcode: 4 bits, modeled using 4 2-dim softmax
- rdatatype: choosing from [1, 28, 18, 42, 257, 60, 59, 37, 5, 49, 32769, 39, 48, 43, 55, 45, 25, 36, 29, 15, 35, 2, 47, 50, 51, 61, 12, 46, 17, 24, 6, 33, 44, 32768, 249, 52, 250, 16, 256, 255, 252, 251, 41], modeled using one 43-dim softmax
- Rdataclass: choosing from [1,3,4,255], modeled using one 4-dim softmax
- edns: 1 bit, modeled using one 2-dim softmax
- dnssec: 1 bit, modeled using one 2-dim softmax
- payload: 16 bits, modeled using 16 2-dim softmax
- url: choosing from ['berkeley.edu', 'energy.gov', 'chase.com', 'aetna.com', 'google.com', 'Nairaland.com', 'Alibaba.com', 'Cambridge.org', 'Alarabiya.net', 'Bnamericas.com'], modeled using one 10-dim softmax

C.2 Packet classification

The details of the packet fields are as follows.

- Source IP: 32 bits, modeled using 32 2-dim softmax
- Destination IP: 32 bits, modeled using 32 2-dim softmax
- Source port: 16 bits, modeled using 16 2-dim softmax
- Destination port: 16 bits, modeled using 16 2-dim softmax
- Protocol: 7 bits, modeled using 7 2-dim softmax

C.3 BAGAN evaluation

To make a fair comparison with BAGAN, we have taken the following approach, which is similar to how we compared with GAN/ACGAN in the paper. First, we request the labels of randomly (uniformly) selected samples up to the given budget (B), and then use those labeled samples to train BAGAN. We used the official BAGAN for MNIST GitHub repo https://github.com/IBM/BAGAN, and the only modifications we made are: (1) instead of having 10 classes, we treat the digit 0 as the rare class, and all other digits as the common class (consistent to how we compared other baselines in paper), and (2) we trained on the labeled samples described above. We keep all other hyper-parameters the same as the original code.

C.4 Elastic-InfoGAN evaluation

We take the official Elastic-infoGAN for MNIST code from https://github.com/utkarshojha/elastic-infogan, and do the following modifications to make it suitable for our problem:

- InfoGAN loss: We keep the InfoGAN loss (which uses all unlabeled data), and on top of it, we use the labeled data to train Q (InfoGAN encoder) in a supervised way.
- Contrastive loss: For a randomly selected labeled sample, we randomly pick a sample with the same label as the positive sample, and a sample with the different label as the negative sample, and apply the same contrastive loss (treating the cosine similarity divided by a temperature parameter as logits) on top of the positive/negative pairs.
- Instead of having 10 classes, we treat the digit 0 as the rare class, and all other digits as the common class (consistent to how we compared other baselines in paper).

We keep all other hyper-parameters the same as the original code.

C.5 Computation resources

All the experiments were run on a public cluster: Bridges-2 system at the Pittsburgh Supercomputing Center (PSC) with NVIDIA Tesla V100 GPUs. All the experiments took around 10k GPU hours. To evaluate DNS amplifications, we set up our own DNS resolvers on Cloudlab (Duplyakin et al. 2019). The server parameters are:

```
CPU: Eight 64-bit ARMv8 (Atlas/A57) cores at 2.4 GHz (APM X-GENE)
RAM: 64GB ECC Memory (8x 8 GB DDR3-1600 SO-DIMMs)
Disk: 120 GB of flash (SATA3 / M.2, Micron M500)
NIC: Dual-port Mellanox ConnectX-3 10 GB NIC (PCIe v3.0, 8 lanes)
```

To evaluate the packet classification time, we set up our own evaluation nodes on Cloudlab (Duplyakin et al. 2019). The server parameters are:

```
CPU: 64-bit Intel Quad Core Xeon E5530
RAM: 12GB
```

D Additional Results: Robustness w.r.t. Labeling Budget and Rare Class Fraction

The results with different labeling budget on MNIST, CIFAR10, DNS amplification attacks, and packet classifiers are in Figs. 4 and 6 to 8 respectively. The results with different rare class fractions are on MNIST, CIFAR10, DNS amplification attacks, and packet classifiers are in Figs. 5 and 9 to 11 respectively. In all cases, we can see that RareGAN with active learning and weighted loss improves upon the baselines. Two other observations: (1) We see that the curves in Figs. 7 and 10 do not have a clear pattern. The reason is that the random trials in DNS tend to have large variances. To illustrate this, we pick the smallest budget in Fig. 7 and the smallest rare class fraction in Fig. 10 and plot their error bars in Figs. 12 and 13. We can see that ACGAN, ALCG, and RareGAN without weighted loss all have large variances. However, weighted loss is able to reduce the variance a lot (as we already demonstrated in the main text). (2) We can see that in CIFAR10 all algorithms tend to have bad FIDs when the budget or rare class fraction are small, as this dataset is challenging. However, even in that case, RareGAN versions still have better FIDs than the baselines.

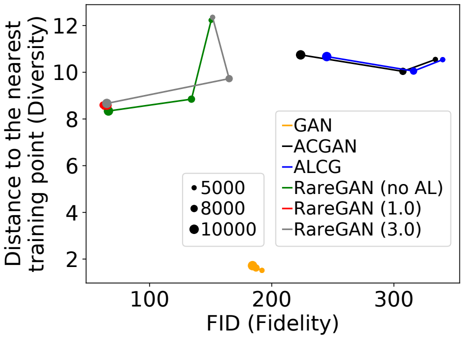


Figure 6: CIFAR10 with different budgets.

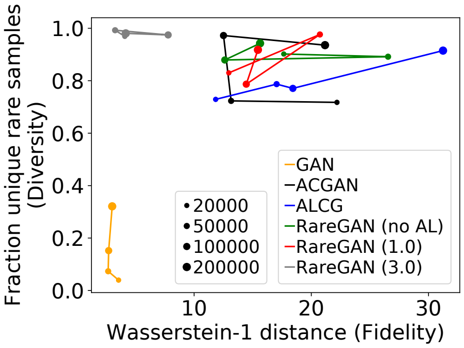


Figure 7: DNS amplification attacks with different budgets.

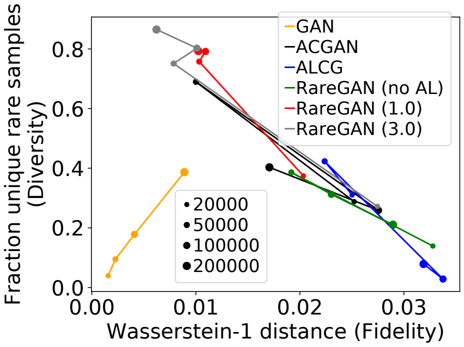


Figure 8: Packet classifiers with different budgets.

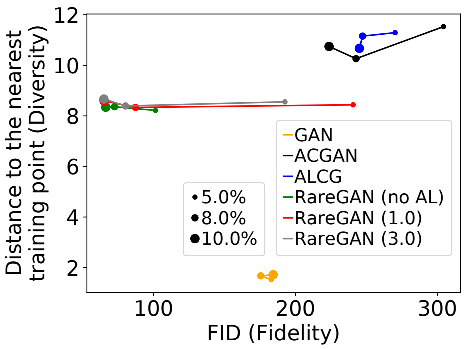


Figure 9: CIFAR10 with different rare class fractions.

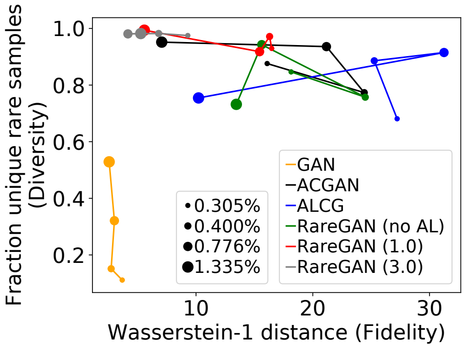


Figure 10: DNS amplification attacks with different rare class fractions. The fractions correspond to $T = 20, 15, 10, 5$ respectively.

E Additional Results: Effect of stages S and loss weight w

Figure 14 illustrates the effect of the two main hyperparameters of RareGAN: the number of stages S and the weight w.

F Additional Results: Ablation Study

Figure 15 illustrates the effect of using various sub-components of RareGANon the DNS dataset. We observe that the full RareGAN has the best utility-diversity tradeoff.

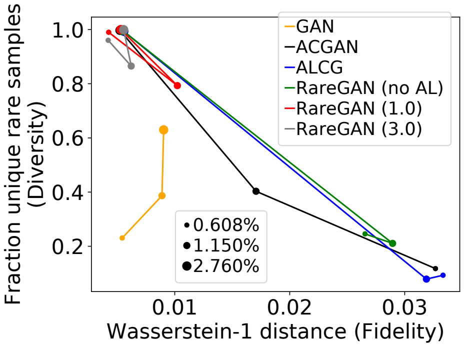


Figure 11: Packet classifiers with different rare class fractions. The fractions correspond to $T = 0.06, 0.055, 0.05$ respectively.

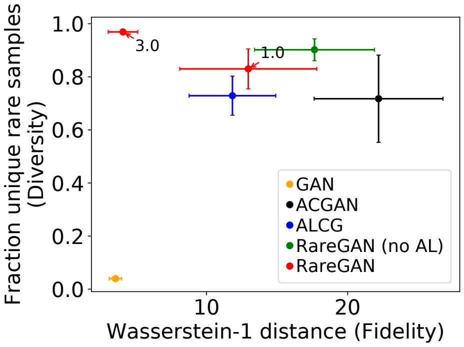


Figure 12: DNS amplification attacks with $B = 20{,}000$.

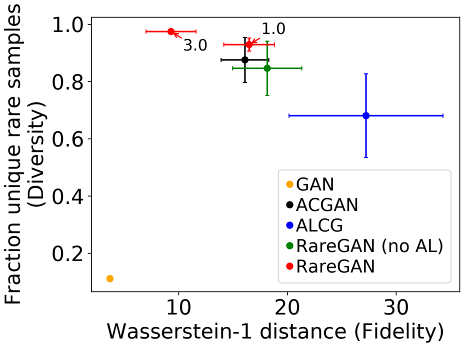


Figure 13: DNS amplication attacks with $\alpha = 0.305\%$ (corresponding to $T = 20$).

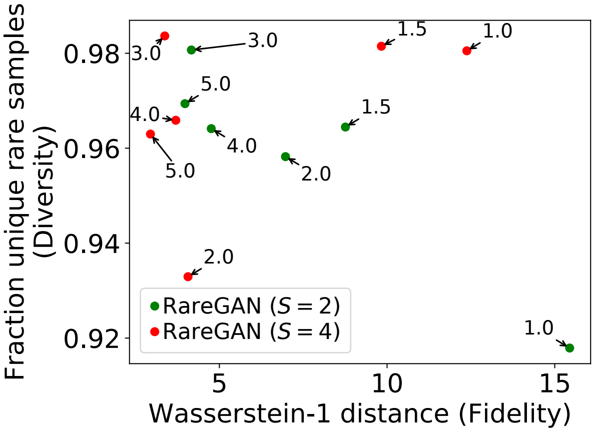


Figure 14: RareGAN with different weights w and numbers of stages S on DNS amplification attacks.

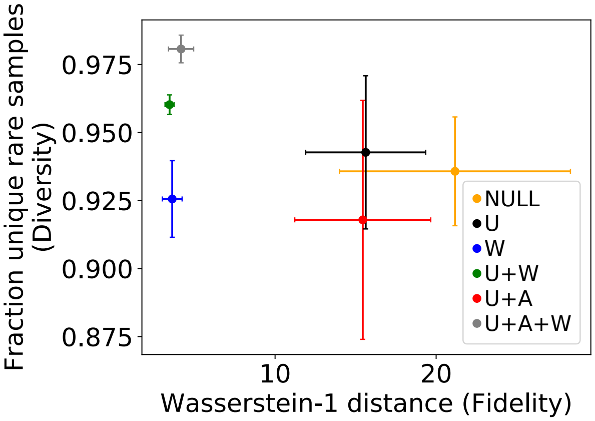


Figure 15: Different combinations of RareGAN components on DNS amplification attacks. U, A, W, NULL refer to using unlabeled samples, active learning, weighted loss, and none of them respectively. Bars show standard error over 5 runs.