CERIAS Tech Report 2024-4
Practicality in Generative Modeling & Synthetic Data
by Daniel Antonio Cardona
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

PRACTICALITY IN GENERATIVE MODELING & SYNTHETIC DATA

by

Daniel Antonio Cardona

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Statistics

West Lafayette, Indiana

August 2024

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Arman Sabbaghi, Co-Chair

Department of Statistics

Dr. Vinayak A.P. Rao, Co-Chair

Department of Statistics

Dr. Bruce A. Craig

Department of Statistics

Dr. Robert Cole

Sandia National Laboratories

Dr. Timothy J. Keaton

Department of Statistics

Approved by:

Dr. Jun Xie

To my mother for being the catalyst for my graduate studies
so that I could become the first doctor in my family

ACKNOWLEDGEMENTS

I would like to thank all the committee members for their advice on research and writing. At some point along the way, each member in the committee has provided significant help with overcoming barriers in my Ph.D. education.

I would like to thank Dr. Arman Sabbaghi for inviting me to work with him on research with Sandia National Laboratories and for serving as my advisor, even after leaving Purdue. I would like to thank Dr. Vinayak Rao for agreeing to serve as a co-advisor after Dr. Sabbaghi's departure, and enabling me to continue my Ph.D. program by granting me research credits over the last few semesters. I would like to thank both Dr. Bruce Craig and Dr. Rao for their support and insightful research advice. Lastly, I would like to thank Dr. Timothy Keaton for his invaluable flexibility in joining my committee within short notice and for his help with my writing.

I would like to thank Dr. Robert Cole and Carter Bullard for their roles in forming the motivational backbone behind this dissertation's research. Dr. Cole, my mentor from Sandia National Laboratories, has made a tremendous impact on my education. Through his curiosity and drive, Dr. Cole provided much of the inspiration behind the novel work in this dissertation. Similarly, Carter Bullard was instrumental in navigating a satellite traffic emulator and in providing cyber-security perspectives. I am lucky he was involved and thank him for his hard work and advice.

Lastly, I would like to thank Felice Brokaw, Hyeong Jin Hyun, and Patti Foster for their persistent and kind outreach towards me. Felice Brokaw has always been there for me and has made my everyday life easier throughout the later stages of my Ph.D. program. Hyeong Jin Hyun has been an amazing friend to me and I credit him with being a significant contributor to reviving my motivation towards the end of this program. With her open door policy, Patti Foster has been a reliable and valuable person for any and every graduate student. Without her support, it would have been incredibly difficult for me to complete this graduate program. Without these three people, I am not sure if I would have gotten this far.

TABLE OF CONTENTS

LIST OF TABLES 7

LIST OF FIGURES 8

ABSTRACT 10

1 LIGHTWEIGHT CHAINED UNIVERSAL SYNTHESIZERS (LiCUS) 11

1.1 Introduction 11

1.2 Background 12

1.2.1 Monte Carlo Sampling for Synthetic Data 12

1.2.2 Extreme Learning Machines 13

1.2.3 Related Work 14

1.3 Experimental Studies 14

1.3.1 Neural Network Generator Simulation Study 18

1.3.2 Imaging Synthesis Study 19

1.3.3 Train Synthetic, Test Real (TSTR) Study 22

1.4 Discussion 24

1.5 Summary 25

2 FEASIBILITY OF SYNTHETIC DATA SUPPLEMENTATION VIA LICUS . . . 27

2.1 Introduction 27

2.2 Background 29

2.2.1 Related Work 29

2.3 Experimental Study 30

2.3.1 Experimental Factors 31

2.3.2 Datasets 35

2.3.3 Experimental Results 36

2.4 Discussion 41

2.5 Summary 42

3 LEVERAGING EXPERT KNOWLEDGE FOR GENERATION OF SATELLITE NETWORK TRAFFIC FLOWS 43
3.1 Introduction 43
3.2 Background 44
3.2.1 Generative Adversarial Networks (GANs) 44
3.2.2 Related Work 46
3.3 Insertion of Expert Knowledge in WGAN-GP 48
3.3.1 NASA Operational Simulation for Small Satellites (NOS3) 48
3.3.2 Expert Knowledge via Argus 49
3.3.3 Model Architecture 50
3.4 Discussion 51
3.5 Summary 55
3.6 Acknowledgements 55

REFERENCES 56

VITA 63

LIST OF TABLES

1.1 Mean and standard error of the FD between the training dataset and LiCUS after 5 replications. 22

1.2 The number of parameters per generative method as defined in the TSTR experiment. 23

2.1 A visual representation of the confusion matrix. The central cells contain counts that form the basis of many statistics. 34

2.2 A summary of class counts for each dataset. 36

3.1 Five randomly generated network flows from the expert-informed WGAN-GP. The top table displays the categorical flow signatures while the bottom table displays the quantitative network flow features. It should be noted that destination port 12001 occurs 60% of the time. 54

LIST OF FIGURES

1.1 Summary of the 3-phase generative process of an M-dimensional random variable for C cycles: (i) initialization (blue), (ii) feature-wise generation (orange), (iii) propagated Gibbs-like synthesis (green) 18

1.2 Empirical CDF plots of both observed and five simulated datasets synthesized by our algorithm. Despite minor deviation along the observed empirical CDF and synthesis pushing the extremal support limits, the synthesized data exhibit empirical CDFs that closely resemble that of the observed data. 20

1.3 A sample of synthetic images produced by LiCUS. 21

1.4 TSTR results on the MNIST dataset and ElasticNet multinomial classifier. Note that training on synthetic data from LiCUS leads to greater testing accuracy when compared to synthetic data from a VAE or GAN. 24

2.1 Lattice plots summarizing experimental results concerning mean testing TPR vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation. 37

2.2 Lattice plots summarizing experimental results concerning mean testing PPV vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation. 38

2.3 Lattice plots summarizing experimental results concerning the mean testing F1 score vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation. 39

2.4 Lattice plots summarizing experimental results concerning mean testing AUC vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation. 40

3.1 An example of NOS3 [57] traffic captured and organized by Argus [60]. 48

3.2 An example of a network flow baseline. Note how the rows list unique categorical flow signatures. 49

3.3 The training process is indexed by batches, or samples, of $\sim 1,000$ flows. Vertical lines denote transitions between epochs, a marker of having trained on all available data ($\sim 300,000$ flows). 52

3.4 Illustration of critic scoring of observed and generated flows. An ideal critic would maximize observed flow scores and minimize generated flow scores. On the other hand, an ideal generator would maximize generated flow scores. The training process is indexed by batches, or samples, of $\sim$1,000 flows. Vertical lines denote transitions between epochs, a marker of having trained on all available data ($\sim$300,000 flows). 53

ABSTRACT

As machine learning continues to grow and surprise us, its complexity grows as well. Indeed, many machine learning models have become black boxes. Yet, there is a prevailing need for practicality. This dissertation offers some practicality on generative modeling and synthetic data, a recently popular application of generative models. First, Lightweight Chained Universal Approximators (LiCUS) is proposed. Motivated by statistical sampling principles, LiCUS tackles a simplified generative task with its universal approximation property while having a minimal computational bottleneck. When compared to a generative adversarial network (GAN) and variational auto-encoder (VAE), LiCUS empirically yields synthetic data with greater utility for a classifier on the Modified National Institute of Standards and Technology (MNIST) dataset. Second, following on its potential for informative synthetic data, LiCUS undergoes an extensive synthetic data supplementation experiment. The experiment largely serves as an informative starting point for practical use of synthetic data via LiCUS. In addition, by proposing a gold standard of reserved data, the experimental results suggest that additional data collection may generally outperform models supplemented with synthetic data, at least when using LiCUS. Given that the experiment was conducted on two datasets, future research could involve further experimentation on a greater number and variety of datasets, such as images. Lastly, generative machine learning generally demands large datasets, which is not guaranteed in practice. To alleviate this demand, one could offer expert knowledge. This is demonstrated by applying an expert-informed Wasserstein GAN with gradient penalty (WGAN-GP) on network flow traffic from NASA's Operational Simulation for Small Satellites (NOS3). If one were to directly apply a WGAN-GP, it would fail to respect the physical limitations between satellite components and permissible communications amongst them. By arming a WGAN-GP with cyber-security software Argus, the informed WGAN-GP could produce permissible satellite network flows when given as little as 10,000 flows. In all, this dissertation illustrates how machine learning processes could be modified under a more practical lens and incorporate pre-existing statistical principles and expert knowledge.

1. LIGHTWEIGHT CHAINED UNIVERSAL SYNTHESIZERS (LiCUS)

1.1 Introduction

Machine learning (ML) algorithms automate modeling and prediction tasks through complex models at the cost of large data requirements. Accordingly, ML has greatly facilitated the use of modeling and understandably grown in popularity. For instance, generative ML methods have garnered plenty of attention in part due to the realism of generated data. Common generative approaches target generating all features all at once [1]–[3]. Other approaches incrementally generate individual features, such as in [4].

Despite all the advantages that ML algorithms possess, they also have disadvantages that limit their application in practice. The major disadvantage in ML lies in the source of its power: scalability of vast quantities of parameters. While platoons of parameters are readily trained in ML algorithms, the training of these parameters collectively demand large datasets. In short, ML procedures are data-hungry and data collection may be laborious or expensive - or both!

Recently, a natural remedy to this data burden has surfaced: supplying ML classifiers with generated data through generative ML models, as in [5], [6]. Nevertheless, barring attempts at leveraging similar datasets, one would expect ML generative methods to be data-hungry due to the complexity of their model structures. In other words, this approach may simply redirect the data burden from the classifier to the generative model.

Furthermore, perhaps due in part to this data burden, black box environments are commonly riddled with obstacles and rely primarily on ad hoc tuning of features, at times akin to knob-turning. For instance, a common example is mode collapse [2], in which the generative model is myopic and only generates a subset of targeted classes. Additionally, optimization of neural networks is troubled with unstable gradients, such as exploding or vanishing gradients. Thus, the training process further compounds the data requirement.

While we do not directly focus on solving the data burden, we seek a lightweight generative model to enhance data usage and avoid the perils of stochastic optimization algorithms altogether. We pursue this goal through lean universal approximators, such as extreme learn-

ing machine (ELM) [7]. We propose an efficient generative process, Lightweight Chained Universal Synthesizers (LiCUS).

As in Gibbs sampling, a popular Markov Chain Monte Carlo (MCMC) sampling method [8], [9], we consider data synthesis with the objective of approximately retaining observed conditional distributions in synthetic datasets. If synthetic data sufficiently maintain their conditional distributions, we hypothesize that synthetic data would inform classifiers of these conditional distributions and hence offer greater training fuel when compared to methods that pursue fidelity or variational inference. Indeed, experimental results suggest LiCUS offers synthetic data with greater training utility than a comparable generative adversarial network (GAN) or variational auto-encoder (VAE).

1.2 Background

1.2.1 Monte Carlo Sampling for Synthetic Data

To provide a statistical framework for data synthesis, it is imperative to review popular classical methodology; notably, Monte Carlo (MC) and Markov Chain Monte Carlo (MCMC). MC sampling refers to a set of simulation methods to approximate probability distributions through random number generators. These methods typically enable simple and convenient estimation of quantities involving probability distributions, such as means, standard deviations, and quantiles. However, MC sampling can become difficult to implement for complex random variables. To address these tasks, MCMC methods have been developed; for a greater review of MCMC, please refer to [9]. These MCMC methods drive Markov chains to asymptotically obtain simulations that ultimately follow the specified target distribution. One popular MCMC algorithm is the Gibbs sampler [10].

Instead of simulating from the entire joint distribution of a set of random variables, the Gibbs sampler simulates from a sequence of conditional distributions. Traditionally, Gibbs sampling is the iterative simulation of each random variable component via its conditional distribution when all remaining variables are held fixed at their previously drawn values. Furthermore, as these conditional distributions target fewer variables, Gibbs sampling exhibits dimension reduction and can greatly simplify many Monte Carlo problems in practice.

It is possible for Gibbs sampling to experience slow convergence. For instance, if two random variables are highly correlated, it would take relatively more Gibbs MCMC draws to fully explore the support of the underlying target joint distribution and thus induce poor mixing. A popular remedy is Hamiltonian Monte Carlo (HMC) [11] in which log-likelihood gradient information is employed to reduce the impact of correlations between samples. Specifically, by iteratively drawing random momentum, the support is better explored by numerically simulating a particle's path via Hamiltonian dynamics.

1.2.2 Extreme Learning Machines

Consider the task of modeling a univariate response y. A single-layer neural network f is then characterized by the number of nodes p, weight matrices W_1, W_2, and an element-wise activation function g:

$$f(z) = W_2 g(W_1 z) \tag{1.1}$$

where $z \in \mathbb{R}^m$, $W_1 \in \mathbb{R}^{p \times m}$, and $W_2 \in \mathbb{R}^{1 \times p}$. For brevity, bias or intercept terms were not explicitly included. Typically, a single-layer neural network would then iteratively optimize its weight matrices through stochastic gradient descent.

As an economical representation of a single-layer neural network, an ELM [12] only optimizes a single weight matrix:

$$h(z) = g(z^T \gamma)\beta \tag{1.2}$$

where $z \in \mathbb{R}^m$, $\gamma \in \mathbb{R}^{m \times p}$, and $\beta \in \mathbb{R}^p$. Again, for brevity, bias or intercept terms were not explicitly included. Similarly, for greater ease of communication, denote γ as the inner weights and β as the outer weights. An ELM would draw γ from some pre-defined weight distribution G. Consequently, only β is optimized through a single least-squares optimization that has a closed-form analytical expression. By construction, ELMs are greatly differentiated by its brief fitting procedure. Indeed, with a bottleneck of a single matrix inversion operation, ELMs require polynomial-time computation. In addition to the drastic simplicity in the training process, ELMs boast an universal approximation property [7]. Therefore,

an ELM retains the modeling capability of a single-layer neural network while reducing the learning process to least squares estimation.

1.2.3 Related Work

Most synthetic work involves two goals: fidelity and machine learning efficacy. Fidelity of generated data often relies on a large array including adversarial discriminators [2], divergences [13], likelihood valuations [1], [5], [6] , and several statistical scores [14], [15]. On the other hand, machine learning efficacy, such as [16], instead seeks to maximize performance of a task when given only synthetic data. The most common metrics involved accuracy measures (e.g., F1 scores) for classification tasks and coefficent of determination R^2 for regression tasks [5], [6].

It is worth noting that most data generation approaches rely on data-hungry neural networks. In contrast to these methods, this component of the dissertation focuses on lightweight data generation. Such lightweight approaches include data augmentation as in [17] and roughly hand-crafted synthetic data [18], [19]. These methods have been shown to ease ML training processes and suggest fidelity as being too costly of a goal for general purposes.

Our work is most similar to the Leave-One-Out (LOO) proposal [13], which also tackles conditional distributions. While LOO only relies on tree-based ML methods, it does not account for uncertainty. Instead, LOO only outputs conditional expectations as synthetic data. In contrast, LiCUS outputs draws from conditional distributions as synthetic data.

1.3 Experimental Studies

Similar to Gibbs sampling, LiCUS simplifies the simulation of all random variables by instead drawing from an iterative sequence of approximated Normal conditional distributions. This sequential sampling will be referred to as a cycle. One may increase the number of cycles by recursively calling our algorithm with the previous output as initialization. Though, only one cycle was considered as there was lack of significant improvement from additional cycles

in all works described in Section 1.3. To proactively compensate for iterative computation, the bulky model space of neural networks is reduced to a leaner space of ELMs.

To formally describe the ELM generative approach, consider an arbitrary iteration of the algorithm. For simplicity, consider the task of synthesizing a dataset of vectors $x_{\mathrm{i}} \in \mathbb{R}^m$ for $\mathrm{i} = 1, \ldots, n$. At iteration $\mathrm{j} \in \{1, .., m\}$, the algorithm targets sampling of the j^{th} random variable. Specifically, a new ELM is created to predict $y_{\mathrm{i}} \coloneqq x_{\mathrm{i,j}}$ given $x_{\mathrm{i},\neg\mathrm{j}} \in \mathbb{R}^{m-1}$ for $\mathrm{i} = 1, \ldots, n$.

With least squares as the fitting procedure for ELMs, it is natural and equivalent to assume an additive Normal error term on the output. Accordingly, we apply the ELM algorithm to model the conditional expectations of $\{y_{\mathrm{i}}|x_{\mathrm{i},\neg\mathrm{j}}\}_{\mathrm{i}=1,\ldots,n}$ and add an additive error term $\epsilon = (\epsilon_1, \ldots, \epsilon_n) \sim \mathcal{N}(0, \sigma_{\mathrm{j}}^2 I)$:

$$y_{\mathrm{i}} = h_{\mathrm{i}}\beta + \epsilon_{\mathrm{i}} \tag{1.3}$$

where $h_{\mathrm{i}} = (1, g(x_{\mathrm{i}}^T\gamma))$, $\gamma \in \mathbb{R}^{m\times p}$ is drawn from some pre-defined weight distribution G, g is the activation function, and $\beta \in \mathbb{R}^{p+1}$. To the best of our knowledge, there is not a generally accepted optimal weight distribution. As long as one selects a continuous weight distribution, the specific choice of the weight distribution does not affect the universal approximation property of the ELM algorithm [7]. While [7] defaulted to a uniform weight distribution for its simulations, this distribution confines each γ_{ij} in a pre-defined interval with equal probability throughout its support, regardless of magnitude. Instead of pre-defining the interval of support, we pre-define both the mean and variance of γ_{ij} and maximize entropy by choosing a Normal distribution [20]. With $\gamma_{\mathrm{ij}} \sim \mathcal{N}(\mu, \sigma^2)$, we defined $\mu = 0$ and considered $\sigma^2 \in \{0.1, 1, 10\}$ in the following sections. Though, there was not significant change with the varying values of σ^2 and settled with $\sigma^2 = 1$. Throughout this chapter, we define each γ_{ij} entry to follow a standard Normal distribution. In all, the j^{th} component will be modeled as:

$$x_{\mathrm{i,j}}|x_{\mathrm{i},\neg\mathrm{j}}, \gamma, \hat{\beta}, \hat{\sigma_{\mathrm{j}}^2} \sim \mathcal{N}((1, g(x_{\mathrm{i},\neg\mathrm{j}}^T\gamma))\hat{\beta}, \hat{\sigma_{\mathrm{j}}^2}) \tag{1.4}$$

Therefore, each conditional distribution will be modeled as a Normal distribution with the ELM output as the mean parameter and an estimated $\hat{\sigma_{\mathrm{j}}}^2$ via maximum likelihood estimation (MLE).

By construction, ELMs are exposed to ill-conditioned or singular $H^T H$ matrices, where the i^{th} row of H is h_{i}. To improve numerical stability during matrix inversions, we added a diagonal jitter or nugget term ηI to $H^T H$ when computing the output weights $\hat{\beta}$:

$$\hat{\beta} = (H^T H + \eta I)^{-1} H^T Y \tag{1.5}$$

where η was set to a low value of 10^{-9}. In nearly all cases in the following experiments in this chapter, singularities were not encountered. Indeed, in the experiments for this chapter, between 0%-0.5% of fitted extreme learning machines encountered singularities when performing a matrix inversion.

In the original work proposing the ELM algorithm [12], it was recommended to avoid singularities by solving least squares optimization problem via a Moore-Penrose generalized inverse. Unfortunately, the authors in [12] did not recommend a specific method to construct the Moore-Penrose generalized inverse in singular settings. When facing a singular $H^T H$, we construct the generalized inverse through singular value decomposition (SVD) and define the output weights as:

$$\hat{\beta} = H^{\dagger} Y = V D^{-1} U^T Y \tag{1.6}$$

where $H^{\dagger} = V D^{-1} U^T$ and $H = U D V^T$ via SVD. Note that this construction of the generalized inverse requires $O(n(p+1)^2)$ time [21].

Another option includes the use of ridge regression. This accessible and low-cost approach could resolve singularity issues while also aiding numerical stability in matrix inversions:

$$\min_{\beta} \sum_{\mathrm{i}=1}^{n} (y_{\mathrm{i}} - h_{\mathrm{i}}\beta)^2 + \sum_{\mathrm{j}=1}^{p} \beta_{\mathrm{j}}^2 \tag{1.7}$$

The regularized solution yields $\beta_{\text{ridge}} = (H^T H + \lambda I)^{-1} H^T Y$ where $\lambda \geq 0$ is the ridge tuning parameter, H and Y are defined such that the k^{th} row is h_k and y_k, respectively. One may choose the value of the tuning parameter λ via cross-validation.

Lastly, as in many MCMC methods, LiCUS requires an initialization. For simplicity, we draw initializations from a multivariate Normal distribution parametrized by maximum-likelihood estimates, given the observed data.

By construction of its fitting procedure, our algorithm is dramatically more computationally efficient than popular neural network algorithms that directly generate from joint distributions. Indeed, at each iteration, our algorithm exhibits computational complexity similar to sequential fitting of multiple standard linear regressions.

For simplicity, we restrict theoretical runtime analysis to scenarios with full-rank hidden matrices and $p < n$ to ensure strictly positive degrees of freedom. Additionally, we assume sampling of inner weights and activation can be executed with constant computational cost $O(1)$. In other words, the computational cost of the previous actions does not increase with respect to the dataset size n, dimensionality of the observed vectors m, and number of nodes p used in LiCUS.

To illustrate the computational cost, we consider simulating a dataset $Z \in \mathbb{R}^{n \times m}$, where n is the number of observations and m is the number of components to simulate. The first step of each iteration involves mapping the input by the sampled inner weights $\gamma \in \mathbb{R}^{m \times p}$, a cost of $O(mnp)$. As each iteration involves fitting a new full-rank p-node ELM, we examine the computational bottlenecks: a cross-product and its inversion. The cross-product of the hidden representation $H \in \mathbb{R}^{n \times p}$ requires $O(np^2)$ times while its inversion $(H^T H)^{-1} \in \mathbb{R}^{p \times p}$ requires $O(p^3)$. Lastly, computation of the outer weights, $(H^T H)^{-1} H^T Y$, are overshadowed by the aforementioned matrix operations. Additionally, if a singular $H^T H$ was encountered, the computation of a generalized inverse would be also be overshadowed by the previous matrix operations. As a result, computation of the outer weights requires $O(np^2 + p^3)$ time.

As a result, each iteration demands $O(mnp+np^2+p^3)$ time, whose summands respectively account for the linear mapping of the input, the cross-product of $H^T H$, and its subsequent inversion. With each cycle comprising of m total iterations, the computational complexity of each cycle is at most $O(m \times (mnp + np^2 + p^3))$. For C total cycles, the complexity is upper bounded by $O(C \times m \times (mnp + np^2 + p^3))$.

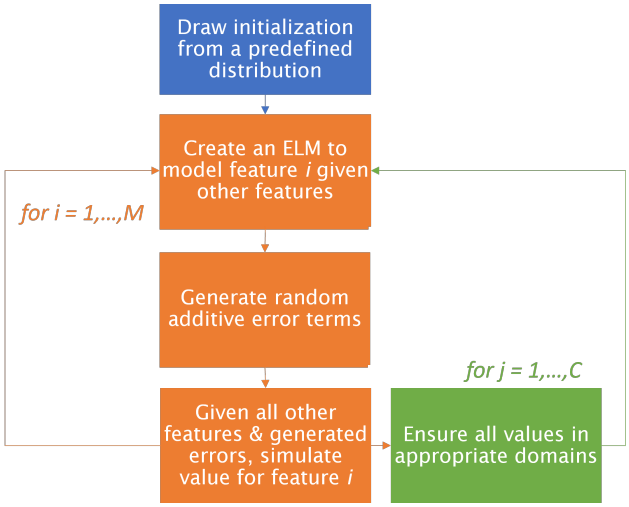


Figure 1.1. Summary of the 3-phase generative process of an M-dimensional random variable for C cycles: (i) initialization (blue), (ii) feature-wise generation (orange), (iii) propagated Gibbs-like synthesis (green)

To explore the practicality of the rapid generative approach, we conduct three studies: (i) a toy dataset with a known complex generator, (ii) synthesis of MNIST images, and (iii) inspection of its synthetic quality.

1.3.1 Neural Network Generator Simulation Study

To provide a proof of correctness, the first simulation study relies on data generated by a unique neural network generator. With 100-dimensional standard Normal random variables as input, our generator is configured with three layers, each armed with 100 nodes. Note that the generator consists of $31,310$ learnable parameters. The generator will rely on the traditional sigmoid activation function:

$$g(x) = \frac{\mathrm{e}^x}{1 + \mathrm{e}^x} \tag{1.8}$$

The generator is characterized by weights independently and identically drawn from the Normal distribution with mean 0 and standard deviation $1/3$, such that all weights are

typically less than 1. In all, the generator constructs a toy dataset of 1,000 independent and identically distributed 10-dimensional variables.

For this study, our approach relies on 10 ELMs, each activated by the sigmoid function (Equation 3.10). To match the number of nodes in the generator, we allocate 300 nodes across 10 ELMs; in other words, each ELM was restricted to exactly 30 nodes. Note that LiCUS will consist of $10 \times 31 = 310$ learnable parameters. So, LiCUS will only use approximately 1% of the number of parameters in the underlying generator.

Lastly, to demonstrate utility under a naive weight sampling distribution, each inner weight was identically and independently sampled from a standard Normal distribution. With these ELMs, five 1000-variable datasets were synthesized - all without singular hidden representations. Note that only one cycle ($C = 1$) was executed as marginal cycles led to minor improvements.

In this study, synthesis quality is measured by visual similarity of empirical cumulative distribution functions (CDFs) of synthetic datasets and the empirical CDF of the observed data obtained from the neural network generator. Figure 1.2 displays some of the empirical CDF plots.

For any generative process, it would be worthwhile to question whether the data are simply memorized. In Figure 1.2, one could observe some deviations between the observed and simulated empirical CDFs. In addition, the simulated data seems to extrapolate past the range of observed data. These observations suggest memorization is not at play, at least qualitatively. The next section provides a quantitative analysis to answer whether LiCUS would simply memorize a real-world training dataset.

1.3.2 Imaging Synthesis Study

The second study considers the application of the ELM generative approach on the popular benchmark dataset of images of handwritten digits: Modified National Institute of Standards and Technology (MNIST) [22]. MNIST consists of 70,000 images of handwritten digits and corresponding labels of the digit's numerical value. There are 10 labeled classes: $0, 1, \ldots, 8, 9$. Additionally, images have 28×28 pixel resolution, where each pixel has integer

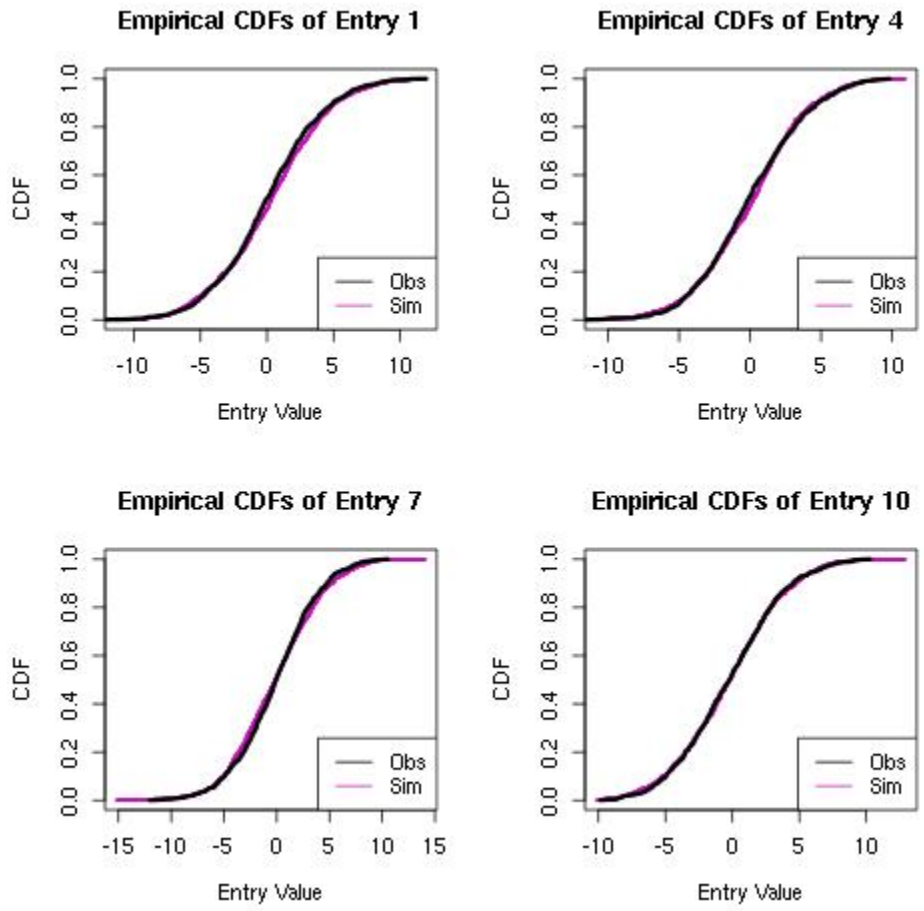


Figure 1.2. Empirical CDF plots of both observed and five simulated datasets synthesized by our algorithm. Despite minor deviation along the observed empirical CDF and synthesis pushing the extremal support limits, the synthesized data exhibit empirical CDFs that closely resemble that of the observed data.

value in $\{0, \ldots, 255\}$. In this study, we apply our approach on a class-wise basis. Per labeled class, we sample 5,000 real images as the training data to fit LiCUS.

For this experiment, synthesis is driven by $28 \times 28 = 784$ ELMs. Preliminary experimentation suggested each ELM should consist of 512 hidden neurons with the hyperbolic tangent function as the activation function:

$$g(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}} \tag{1.9}$$

Figure 1.3. A sample of synthetic images produced by LiCUS.

To evaluate utility despite a naive weight sampling distribution, the inner weights were identically and independently sampled from the $\mathcal{N}(0,1)$ distribution. Under this weight distribution, full-rank synthetic data matrices were obtained.

We acknowledge pixel domains and compromise by the following transformation: add half a pixel value and then apply log-odds to output continuous values. Since marginal cycles led to minor improvements in fidelity, only a single cycle was employed (e.g., $C = 1$). Figure 1.3 illustrates synthetic images curated by LiCUS.

Figure 1.3 offers a snapshot of randomly selected synthetic images generated by LiCUS. LiCUS reasonably models the general shape of all digits. As evidenced by some patchy synthetic digits, LiCUS does not exhibit pristine pixel resolution.

To examine whether LiCUS is simply memorizing the training dataset, we employ the popular method Fréchet Inception Distance (FID) [23]. FID computes the Fréchet distance (FD) between the feature representations of two datasets, say R and G, by assuming each sample follows a Gaussian distribution:

$$FD((\mu_R, \Sigma_R), (\mu_G, \Sigma_G)) = ||\mu_R - \mu_G||^2 + \text{tr}(\Sigma_R + \Sigma_G - 2(\Sigma_R\Sigma_G)^{1/2}) \tag{1.10}$$

where $\mu_R, \mu_G \in \mathbb{R}^p$ are the respective mean statistics of datasets R, G and $\Sigma_R, \Sigma_G \in \mathbb{R}^{p\times p}$ are the respective covariance statistics of datasets R, G.

Specifically, FID first extracts feature representations via the Inception-v3 model [24] for each dataset R, G and then computes the FD between the two sets of feature representations. However, FID was designed for ImageNet or similar datasets and exclusively relies on the

Table 1.1. Mean and standard error of the FD between the training dataset and LiCUS after 5 replications.

Mean	Standard Error
0.226	0.0095

Inception-v3 classifier, which was also trained on ImageNet. This is an obstacle as we focus on MNIST, a relatively small grayscale image dataset, whereas ImageNet is a relatively large image dataset with RGB color channels. In other words, MNIST images are not compatible with the Inception-v3 input framework.

Due to Inception-v3's incompatibility with the MNIST dataset, we perform feature extraction by training a MNIST classifier. We define our classifier as a single-layer neural network activated by hyberbolic tangent (Equation 1.9) and its optimization method as stochastic gradient descent (SGD) [25] with learning rate 0.01 and batch size of 5,000. After conducting a grid search on the number of nodes {4,8,16,32,64,128}, the testing accuracy on held-out data is highest at $\sim 93\%$ when nodes are 64 or 128. The difference between the testing accuracy of the 64-node and the 128-node architectures is less than 1%. Thus, we opt for the smaller architecture and proceed with training our 64-node neural network for 400 epochs.

After training the classifier, we computed the FD between the extracted feature representations of the training and LiCUS datasets. We repeat this procedure 5 times and present the mean and standard errors of the computed FDs in Table 1.1.

By definition of the FD, if LiCUS yields a FD of 0, there is perfect memorization by LiCUS. Since the mean FD between the training datasets and LiCUS synthetic datasets is >20 standard errors away from 0, it is reasonable to assert that LiCUS is not memorizing the training dataset.

1.3.3 Train Synthetic, Test Real (TSTR) Study

To measure quality of generated data, we turn to the "Train on Synthetic, Test on Real" (TSTR) method [16]. In short, TSTR reports the test metrics of a model trained exclusively

Table 1.2. The number of parameters per generative method as defined in the TSTR experiment.

Method	Number of Parameters
LiCUS	402,192
VAE	402,752
GAN	403,005

on synthetic data. TSTR offers an avenue to gauge the utility of synthetic data to prepare classifiers for unseen data.

In this experiment, we give a multinomial classifier generated data from an individual data generator and examine its testing accuracy on unseen real data. The multinomial classifier is regularized by ElasticNet [26] selected from a grid of tuning parameters via 5-fold cross-validation: 5 different α values and 100 different λ values for each α value. The classifier is trained on 1,000 images from varying sources: the real training dataset, LiCUS, vanilla GAN, and vanilla VAE. In addition, each generative method was trained on 5,000 real images. Lastly, the testing dataset consists of 1,000 unseen images. We repeat the experiment 5 times and visualize the mean TSTR outcomes per data source in Figure 1.4.

To better enable fair comparisons between models, we restrict each neural network model to single fully-connected layers. Similarly, following the LiCUS procedure, each class of data is given a dedicated single neural network. Lastly, we aim to provide approximately the same number of parameters to each generative method. For the LiCUS method, we fix the per-class nodes to 512. This forces the number of nodes for each neural network in VAE and GAN to 224 and 238, respectively. In addition, the VAE and GAN models will have 112 and 119 latent random variables, respectively. As shown in Table 1.2, these architectures ensure that each generative method is armed with roughly the same number of parameters.

As LiCUS has full access to its training dataset throughout the fitting process, we mirror this process in the optimization of the neural networks: the batch size is equal to the training dataset size. Under the stochastic gradient descent optimization framework [25], preliminary

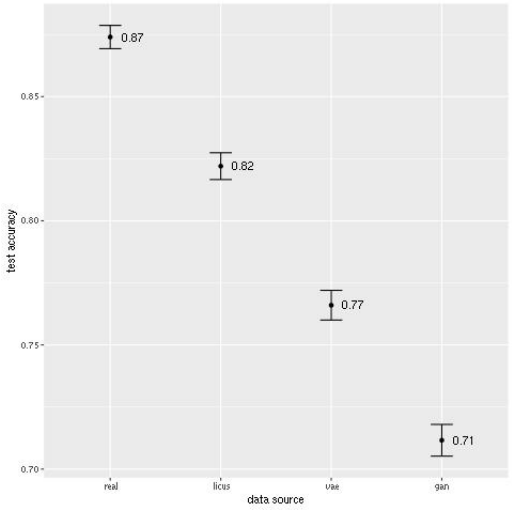


Figure 1.4. TSTR results on the MNIST dataset and ElasticNet multinomial classifier. Note that training on synthetic data from LiCUS leads to greater testing accuracy when compared to synthetic data from a VAE or GAN.

experimentation suggested a learning rate of 1 and 10^{-6} for the GAN and VAE, respectively, for $1,000$ epochs, or steps.

As seen in Figure 1.4, LiCUS provided synthetic data with greater training utility than the other generative methods. Though, the best performance arises when the classifier is trained directly with real data. Indeed, training with real data resulted in a mean accuracy of 87% compared that of 82% when training with LiCUS.

1.4 Discussion

The simulation study explored whether the proposal could emulate a hypothetical neural network generator. Figure 1.2 suggests that the ELM-based generative procedure successfully outputs representative synthetic data. After compiling five synthetic datasets, there are very few instances out of 5,000 replicates that escape the observed range. Hence, our algorithm appears to be an efficient substitute for a neural network generator, which is expensive to obtain under a GAN framework. In all, the proposed iterative scheme is capable of simulating

from a hypothetical neural network generator, at least in regards to marginal distributions. We delegate non-marginal investigation to the MNIST study.

The MNIST dataset easily enables quick inspection of whether the correlation structure of a synthetic image is adequate; e.g., is this image visually reasonable? As such, the MNIST study grants insight into whether LiCUS is capable of addressing complex correlation structures. Ultimately, it seems there is a trade-off for the simplicity of our ELM algorithm in terms of fuzzy or spotty resolution. Lastly, an analysis of the FD between the extracted feature representations of the training and LiCUS datasets suggests that LiCUS is not memorizing the training dataset.

Lastly, via the TSTR study, the approximation of conditional distributions evidently enables LiCUS to offer utility in providing training data. Indeed, training a classifier exclusively on synthetic data from LiCUS is nearly comparable to a classifier trained directly on real data. Interestingly, LiCUS offers synthetic data similar to real data in terms of training information. Additionally, while the GAN method solely pursues a fidelity objective, it offers the least training information out of all sources considered. Furthermore, perhaps due to its variational inference framework, the VAE outperforms the GAN in terms of creating useful training material.

1.5 Summary

Instead of tackling a joint distribution altogether, LiCUS embodies a generative strategy akin to Gibbs sampling. With a small cost in generation quality, our ELM-based approach only requires the computational cost of least-squares optimization. In short, we offer a lightweight synthesis approach with a foundation derived off statistical principles.

To verify the synthesis proposal, we offer two studies. First, we construct a simulation study to validate proof of correctness for data generated by a hypothetical neural network generator. Therefore, our efficient ELM approach is comparable to an otherwise expensive generator. Second, the application to the MNIST dataset demonstrates capability of addressing complex correlation structures. While fidelity is not a goal here, it is clear that LiCUS easily captures structure and variety in its synthetic output; alas, its constrained complexity

does not appear to afford pristine realism. Furthermore, in a very similar approach to FID, analysis of the FD between the extracted feature representations of the training and LiCUS datasets suggests that LiCUS is not memorizing the training dataset.

To validate the utility of the proposal, we consider the TSTR approach. By training on only synthetic data, we gain perspective on the quality of synthetic information for testing performance. As a result, LiCUS seems to offer substantial training information, nearly similar to real training data itself. Indeed, experimental results suggest LiCUS offers synthetic data with greater training utility than a comparable GAN or VAE. That being said, a future research direction could include studying how LiCUS would fare if the sizes of the training and synthetic datasets were varied.

2. FEASIBILITY OF SYNTHETIC DATA SUPPLEMENTATION VIA LICUS

2.1 Introduction

In practice, datasets are not collected with equal and balanced class representations. Such datasets are termed as imbalanced datasets. For instance, it is very common to encounter imbalanced datasets in fraud detection [27], [28] or disease detection [29], [30]. The *class imbalance* problem, or "curse of imbalanced datasets", increases the difficulty of classification tasks [31], [32]. In short, it is trivial to achieve a high accuracy on the dominant class, or the majority class, while performing poorly on the other small classes, or the minority classes. This problem is amplified when the minority class is the primary class of interest.

In general, there are three categories of machine learning approaches for imbalanced datasets [33]: data-level methods, algorithm-level methods, and hybrid methods. Data-level methods modify the dataset to remove the bias towards the majority class. Algorithm-level methods adjust the cost functions to rectify the imbalance bias. Lastly, hybrid methods merge both data-level and algorithm-level methods. With the overhead cost of synthetic data, algorithm-level methods are computationally cheaper and thus expected to be most popular in practice. However, if one were able to manufacture better training data via data-level methods, data-level methods would be the preferred method. This dissertation chapter will focus on data-level methods and label such supplemented data as synthetic data.

Specifically, consider the optimization of a binary classifier with some additive loss function C for some dataset $\{(x_{\mathrm{i}}, y_{\mathrm{i}})\}_{1\leq \mathrm{i} \leq n}$ where $x_{\mathrm{i}} \in \mathbb{R}^p$ and $y_{\mathrm{i}} \in \{0,1\} \forall \mathrm{i} = 1, \ldots, n$:

$$\min_{\theta} C(x, y, \theta) = \min_{\theta} \sum_{\mathrm{i}=1}^{n} C(x_{\mathrm{i}}, y_{\mathrm{i}}, \theta) \tag{2.1}$$

where $\theta \in \mathbb{R}^d$ represents the parameters of the binary classifier. Data-level methods insert synthetic data into the training dataset and optimize for θ given the expanded dataset:

$$\min_{\theta} C(x, y, \tilde{x}, \tilde{y}, \theta) = \min_{\theta} \sum_{\mathrm{i}}^{n} C(x_{\mathrm{i}}, y_{\mathrm{i}}, \theta) + \sum_{\mathrm{j}=1}^{L} C(\tilde{x}_{\mathrm{j}}, \tilde{y}_{\mathrm{j}}, \theta) \tag{2.2}$$

where $\{(\tilde{x}_j, \tilde{y}_j)\}_{1 \leq j \leq L}$ is the supplemental synthetic dataset. Of course, $\tilde{x}_j \in \mathbb{R}^p$ and $\tilde{y}_j \in \{0, 1\}$ for $j = 1, \ldots, L$. While Equation 2.2 describes a general data-level method, other variants will be covered later in the chapter.

Practitioners of synthetic data supplementation would benefit from reviewing previous works. It may be desirable to learn about the effects of class imbalance levels, which statistics have been shown to be boosted by synthetic data supplementation, and how much synthetic data one should supplement. Some of the most popular data-level methods have studied synthetic data supplementation given a single class imbalance level [34]–[36]. It should be noted that these works do not detail why only the observed class imbalance level was studied. Another work [37] studied the sensitivity of its parameter with respect to artificial class imbalance levels. In [37], the authors proposed the integration of perhaps the simplest data-level method, random over-sampling of minority instances, into the training process of a deep neural network. It should be noted that the datasets in [37] were not naturally imbalanced and hence had to be artificially imbalanced. Lastly, out of the papers listed, two of them focus on a single statistic while the other two consider multiple statistics. While these four papers do not constitute an audit of literature for data-level methods, we posit that it may be uncommon to find previous literature exploring the benefits of synthetic data supplementation over varying imbalance levels and over multiple statistics.

In order to create a starting point for future practitioners, we study synthetic data supplementation over varying imbalance levels for multiple target statistics. There also has not been much guidance on how much synthetic data one should add, especially over multiple imbalance levels and for specific target statistics. Such work could provide a valuable starting point for practitioners working on synthetic data for imbalanced data. Lastly, we consider why data-level methods tend to distort class representation in the training phase and hence bias classifiers. Furthermore, to the best of our knowledge, it has not been studied whether synthetic data would provide comparable value to simply collecting more real data. In practice, this would be an important question to answer.

Armed with Lightweight Chained Universal Synthesizers (LiCUS), this dissertation component primarily details our investigation into (i) how much synthetic data one should add, (ii) the effect of class imbalance on performance after synthetic data supplementation, and

(iii) which statistics are feasible for improvement. Additionally, we consider an unbiased synthetic data scheme such that all classes are supplemented and thus respect the observed class representation. Lastly, with our gold standard doubling as a surrogate for the collection of more real data, we shed light on whether one should create synthetic data or simply collect more real data.

2.2 Background

2.2.1 Related Work

Data synthesis could range from simple re-sampling from the observed dataset to a complex generative machine learning algorithm. The underlying goal is to reduce the class imbalance of the dataset. We begin with discussing foundational data-level methodology: over-sampling and under-sampling. Over-sampling consists of increasing the size of the minority class with the supplementation of synthetic data. On the other hand, under-sampling is the process of reducing the quantity of the majority class. The motivation is to reduce the volume of the majority class in an effort to address the class imbalance of the dataset.

While over-sampling or under-sampling appear to be the most direct and widespread tool for data-level methods, there is another popular synthetic data method: synthetic minority over-sampling technique (SMOTE) [34]. In order to create synthetic minority data, SMOTE linearly interpolates amongst some nearest neighbors. In comparison to over-sampling and under-sampling, SMOTE has been demonstrated to provide greater utility and undoubtedly led to its rise in popularity. Of course, there have been many extensions to SMOTE that typically include specific algorithms to eliminate individual members from the majority class, minority class, and the resulting synthetic dataset [35], [36].

Of course, there are relevant data-level methods involving neural networks [38]. For instance, there is the two-phase method [39] in which one limits the size of all classes via random under-sampling, trains on that modified dataset, and then fine-tunes on the original dataset. Another method is dynamic sampling [40] in which one creates a score to guide both over-sampling and under-sampling on all classes.

Previous work tends to compare against other synthetic data methods, focus on a single statistic, and consider a single imbalance level. However, there is not much work comparing synthetic data supplementation across multiple statistics or imbalance levels, let alone consideration of both. In addition, LiCUS has not yet been used in a synthetic data supplementation study. Lastly, there has not been any work to compare synthetic data supplementation with the collection of additional real data.

While we consider synthetic data for classification tasks, other works study synthetic data for generative tasks. Specifically, with the rise of large language models (LLMs), there is concern about the increasing artificial footprint when training future LLMs. For instance, [41] labeled the detrimental effect of iterative retraining of generative models as *model collapse*. Another work provides two theoretical conditions to ensure stable iterative retraining of generative models [42]: sufficient generative model complexity and size on real data relative to synthetic data. While [42] recommends synthetic data to be at most the volume of the real dataset in generative settings, our work identifies some benefits outside of that recommended range for classification settings.

2.3 Experimental Study

To shed light on feasible synthetic data supplementation, we conduct an experiment investigating the effects of varying supplementation schemes for different class imbalance levels and target statistics. Furthermore, we propose the use of a gold standard supplement, which is defined as the collection of more real data. Collection of additional real data will be simulated by a reserved data partition. Therefore, the observed dataset will be partitioned into 3 sets: training, reserved, and testing. In all, such a study would offer practitioners a starting point for future applications of synthetic data supplementation. Hence, we proceed with defining factors for the experimental study that would be considered by a practitioner: synthetic multiples, supplementation scheme, class imbalances, and targeted statistics.

2.3.1 Experimental Factors

Synthetic multiples are defined as the number of synthetic datasets that are supplemented, each with the same size as the expanded classes. For instance, a synthetic multiple of 2 declares that two synthetic datasets were provided as supplemental data. In a foundational work, only 1-5 synthetic multiples were deemed necessary [34]. We continue under this assertion throughout the experiment. Of course, higher values of synthetic multiples could have been considered. Nevertheless, the experiment's results reflect converging or monotonic trends with at most 5 synthetic multiples. In other words, the experimental results suggest that greater synthetic multiples would likely not lead to different conclusions. This was also confirmed with additional experimentation with higher values of synthetic multiples.

For our experiment, we choose a Bayesian Logistic Regression model with a weakly informative prior [43] to serve as our classifier. Given that each predictor is normalized to zero mean and unit variance, the Bayesian model is defined with the following structure:

$$y_{\mathrm{i}}|x_{\mathrm{i}},\theta \overset{\text{iid}}{\sim} \text{Bernoulli}(x_{\mathrm{i}}^T\theta) \forall \mathrm{i} \in \{1,\ldots,n\} \tag{2.3}$$

$$\theta_0 \sim \text{Cauchy}(0,10) \tag{2.4}$$

$$\theta_{\mathrm{j}} \overset{\text{iid}}{\sim} \text{Cauchy}(0,2.5) \forall \mathrm{j} \in \{1,\ldots,p\} \tag{2.5}$$

where n is the size of the dataset, p is the number of predictors, and the intercept is included in each $x_{\mathrm{i}} \in \mathbb{R}^{p+1}$. Hence, for given x, y, θ, the model has the following log-likelihood and log-prior functions:

$$l(x,y|\theta) = y(x^T\theta) - \log(1+\mathrm{e}^{x^T\theta}) \tag{2.6}$$

$$h(\theta) = -\log(10\boldsymbol{\pi}) - \log\left(1+\left(\frac{\theta_0}{10}\right)^2\right) - p\log(2.5\boldsymbol{\pi}) - \sum_{\mathrm{j}=1}^{p}\log\left(1+\left(\frac{\theta_{\mathrm{j}}}{2.5}\right)^2\right) \tag{2.7}$$

As it is common to only provide synthetic data to the minority class, it is natural to continue with this biased supplementation scheme, despite its distortion of class representation

during the training phase. The following equation describes the accompanying optimization problem:

$$\min_{\theta} L_{\text{biased}}(x, y, \tilde{x}^m, \tilde{y}^m|\theta) = \min_{\theta} \sum_{\text{i}=1}^{n} l(x_{\text{i}}, y_{\text{i}}|\theta) + \sum_{\text{j}=1}^{kn_m} l(\tilde{x}_{\text{j}}^m, \tilde{y}_{\text{j}}^m|\theta) + h(\theta) \tag{2.8}$$

where $(x_{\text{i}}, y_{\text{i}})$ for $\text{i} = 1, \ldots, n$ denotes the observed dataset, $(\tilde{x}_{\text{j}}^m, \tilde{y}_{\text{j}}^m)$ for $\text{j} = 1, \ldots, n_m$ is the supplemented synthetic data of the minority class, n_m is the number of observed minority members, and k is the synthetic multiplier. However, we explore the utility of an unbiased supplementation scheme. To be more specific, each class will be supplemented with synthetic data and results in the following optimization problem:

$$\min_{\theta} L_{\text{unbiased}}(x, y, \tilde{x}^m, \tilde{y}^m, \tilde{x}^M, \tilde{y}^M|\theta) = \min_{\theta} \sum_{\text{i}=1}^{n} l(x_{\text{i}}, y_{\text{i}}|\theta) + \sum_{\text{j}=1}^{kn_m} l(\tilde{x}_{\text{j}}^m, \tilde{y}_{\text{j}}^m|\theta) + \sum_{h=1}^{kn_M} l(\tilde{x}_h^M, \tilde{y}_h^M|\theta) + h(\theta) \tag{2.9}$$

with similar components as in Equation 2.8 and some new notation: $(\tilde{x}_{\text{j}}^M, \tilde{y}_{\text{j}}^M)$ for $\text{j} = 1, \ldots, n_M$ is the supplemented synthetic data of the majority class and n_M is the number of observed majority members.

Another experimental factor to consider is the artificially induced class imbalance level of the observed dataset. Unfortunately, there are no benchmark levels for class imbalance. Indeed, relevant works typically adopt the observed class imbalance and hold it as a static variable. Instead, we consider varying class imbalances to gain more insight compared to a singular class imbalance level. Unfortunately, there are no generally accepted representative class imbalance levels. Nevertheless, each finite dataset has limited expressibility in terms of artificial class imbalance levels. There is an additional constraint: by construction, this experiment requires 3 partitions. As in [34], we study datasets that have ≥ 20 minority class samples per partition. Hence, our modified datasets should have ≥ 60 total minority class instances. For our datasets, it is possible to achieve this with class imbalance levels as low as 5%. Naturally, each marginal class imbalance level is doubled and thus we have the following class imbalance levels: 5%, 10%, and 20%.

Before moving past class imbalance levels, it should be noted that inducing a specific artificial class imbalance level in a dataset must change the resulting modified dataset's

size. In other words, there are multiple modified dataset sizes across individual datasets and class imbalance levels. Therefore, while dataset size could be considered an additional relevant factor, dataset size was ultimately fixed at its lowest viable level in order to study environments in which data is not abundant.

Lastly, we select popular statistics that require the use of correct minority classification. The suite of target statistics are the true positive rate (TPR), positive predictive value (PPV), F1 score, and area-under-the-curve (AUC). It should be noted that PPV is also known as precision while TPR is also known as recall or sensitivity. While there are several other statistics available, these statistics were selected such that the experiment explores the effects of targeting coarse statistics (e.g., TPR) and combinations of multiple coarse statistics (e.g., F1 score). To compute any of these statistics, we extract expected coefficients by computing the mean of 10,000 draws from the posterior distribution of the Bayesian logistic regression model and choose a classification threshold via cross-validation.

Before explaining these statistics in greater detail, we denote the minority class as the positive class while the majority class is denoted as the negative class. It will also be helpful to define some terminology: TP denotes the number of predicted positives for labeled positives, FN denotes the number of predicted negatives for labeled positives, and FP denotes the number of predicted negatives for labeled positives. Table 2.1 serves as a visual aid to convey how these counts are combined to form several statistics.

$$\text{F1} = \frac{2 \times \text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} \tag{2.10}$$

Next, we must discuss the F1 score and AUC. First, the F1 score (Equation 2.10) is defined as the harmonic mean of the TPR and PPV. The F1 score serves as a common transformation of both TPR and PPV into a scalar value. Lastly, to discuss AUC, we must introduce the True Negative Rate (TNR), the counterpart of TPR, and the receiver operating curve (ROC) [44]. Given a classifier, the ROC summarizes the classifier's TPR given its FPR $= 1 - \text{TNR}$, where TNR is defined in Table 2.1. Specifically, many probability threshold values for classification are created to compute threshold-specific TPR and FPR values. In short, the ROC enables plotting a classifier's TPR vs FPR across many thresholds.

Table 2.1. A visual representation of the confusion matrix. The central cells contain counts that form the basis of many statistics.

		Actual Class		
		Positive Class	Negative Class	
Predicted Class	Predicted Positive	True positive (TP)	False positive (FP)	Positive Predictive Value (PPV) $\frac{TP}{TP+FP}$
	Predicted Negative	False negative (FN)	True negative (TN)	Negative Predictive Value (NPV) $\frac{TN}{TN+FN}$
		True Positive Rate (TPR) $\frac{TP}{TP+FN}$	True Negative Rate (TNR) $\frac{TN}{TN+FP}$	

ROC is a popular method to visually assess performance while AUC summarizes the ROC as a scalar value by computing the area under the ROC. Both AUC and F1 score are contained within $[0, 1]$ and higher values are desired.

2.3.2 Datasets

By construction, our experiment demands specific imbalanced datasets. For completeness, these datasets should naturally exhibit an imbalance between 2 classes. In addition, following common practice concerning the class imbalance problem, we ask for datasets with at least 20 samples in the minority class for data synthesis. Thus, after a three-way partitioning, a suitable dataset would contain at least 60 minority instances. In all, we would like a set of datasets that can be induced to shared artificial class imbalance levels while consisting of at least 60 minority instances. With these constraints in mind, we select 2 appropriate datasets from the many datasets listed in [34]: the Mammography and Phoneme datasets. Specifically, we could satisfy our conditions with artificial class imbalance levels as low as 5%. Note that artificially changing class imbalance levels changes the dataset size. Over all the considered artificial class imbalance levels, the lowest dataset size will be $1,300$ and so datasets are limited to this size for all class imbalance levels.

First, the Mammography dataset [45] enables cancer detection via mammogram image features. In this dataset, the minority class consists of the 260 images with calcifications while the majority class has 10,923 images without calcifications. In total, there are 11,183 samples, each with 6 hand-crafted continuous features selected from prior literature [45]:

- Average grey level of calcification
- Gradient strength of calcifications perimeter pixels
- Root mean square (RMS) noise fluctuation in the calcification
- RMS noise fluctuation in the 3.5mm x 3.5mm local background
- Contrast, defined as average grey level of the calcification minus the average of a two pixel wide border surrounding the calcification

- A low order moment based shape descriptor [46]

Second, the Phoneme dataset [47] offers data to classify between nasal and oral sounds. In this dataset, the oral sounds class is the minority class, with 1,586 samples. On the other hand, the nasal sounds class is the majority class with 3,818 samples. In total, 5,404 sounds comprise the Phoneme dataset. Each sample has 5 continuous features detailing the normalized amplitudes of the first 5 harmonics.

Table 2.2. A summary of class counts for each dataset.

Dataset	Majority Count	Minority Count
Mammography	10,923	260
Phoneme	3,818	1,586

2.3.3 Experimental Results

To recap, we conduct a synthetic data experiment with the following factors: synthetic multiples, supplementation scheme, class imbalance, and target statistics. We conduct a factorial experiment with 100 replicates. For each replicate, we randomly partition the entire observed dataset into 3 sections with the aptly named 'extra' reserved or validation dataset serving as our gold standard of additional real data collection. Our LiCUS synthesis procedure is a coordinated ensemble of 500-node extreme learning machines [7], each regularized by relaxed LASSO [48] with cross-validation. The accompanying initialization scheme is simply feature-wise re-sampling.

We provide several lattice plots (Figures 2.1,2.2,2.3, and 2.4) to visually summarize the experimental results. A train-only control method, denoted by *standard*, was included in which no synthetic data is added. All lattice plots detail performance with unseen testing data. Unlike the F1 and AUC statistics, these results suggest that synthetic data tends to be most beneficial for TPR and PPV in testing settings. In addition, the biased supplementation scheme often outperforms the unbiased supplementation scheme. Further discussion of these lattice plots is reserved for Section 2.4.

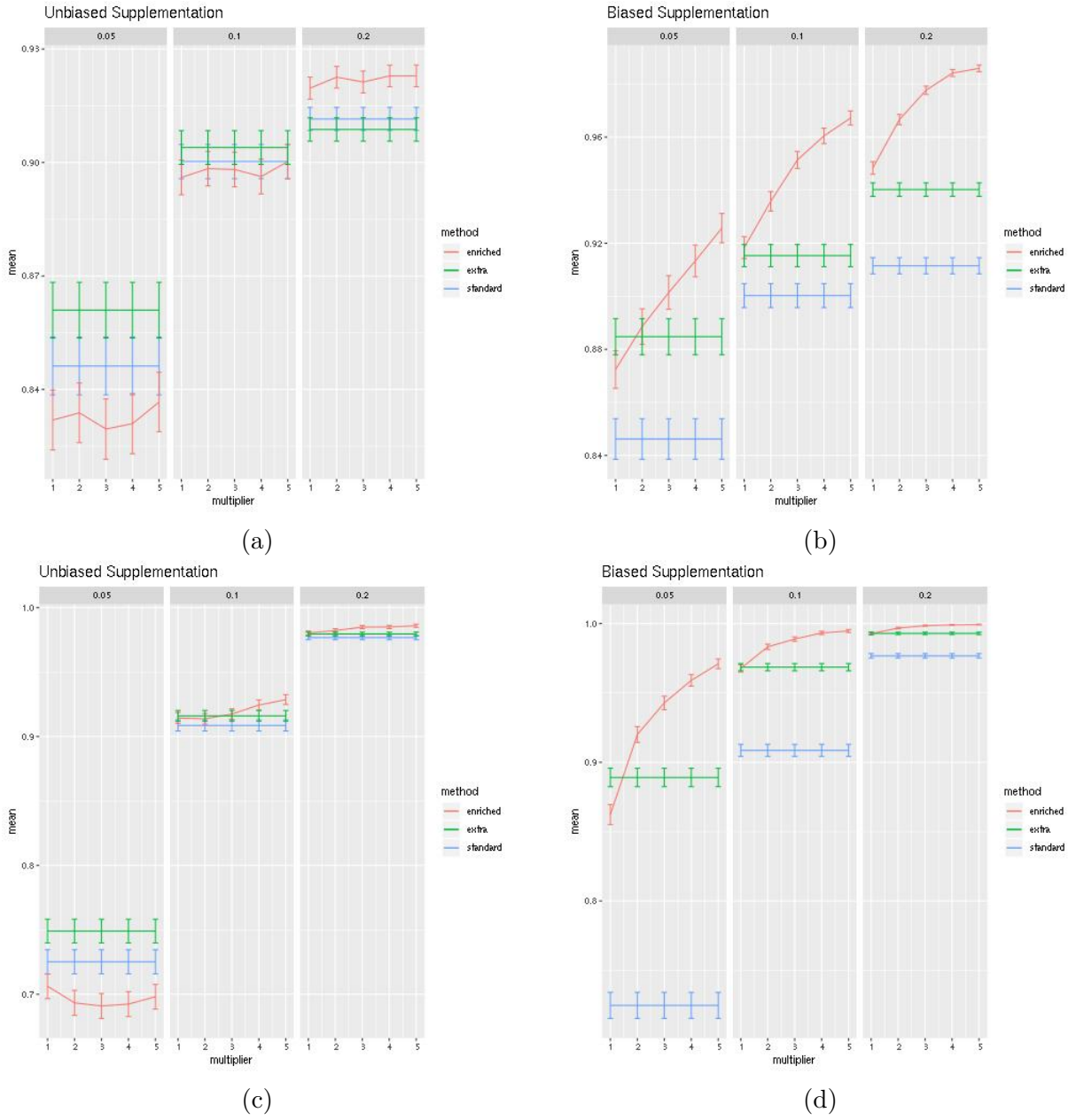


Figure 2.1. Lattice plots summarizing experimental results concerning mean testing TPR vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation.

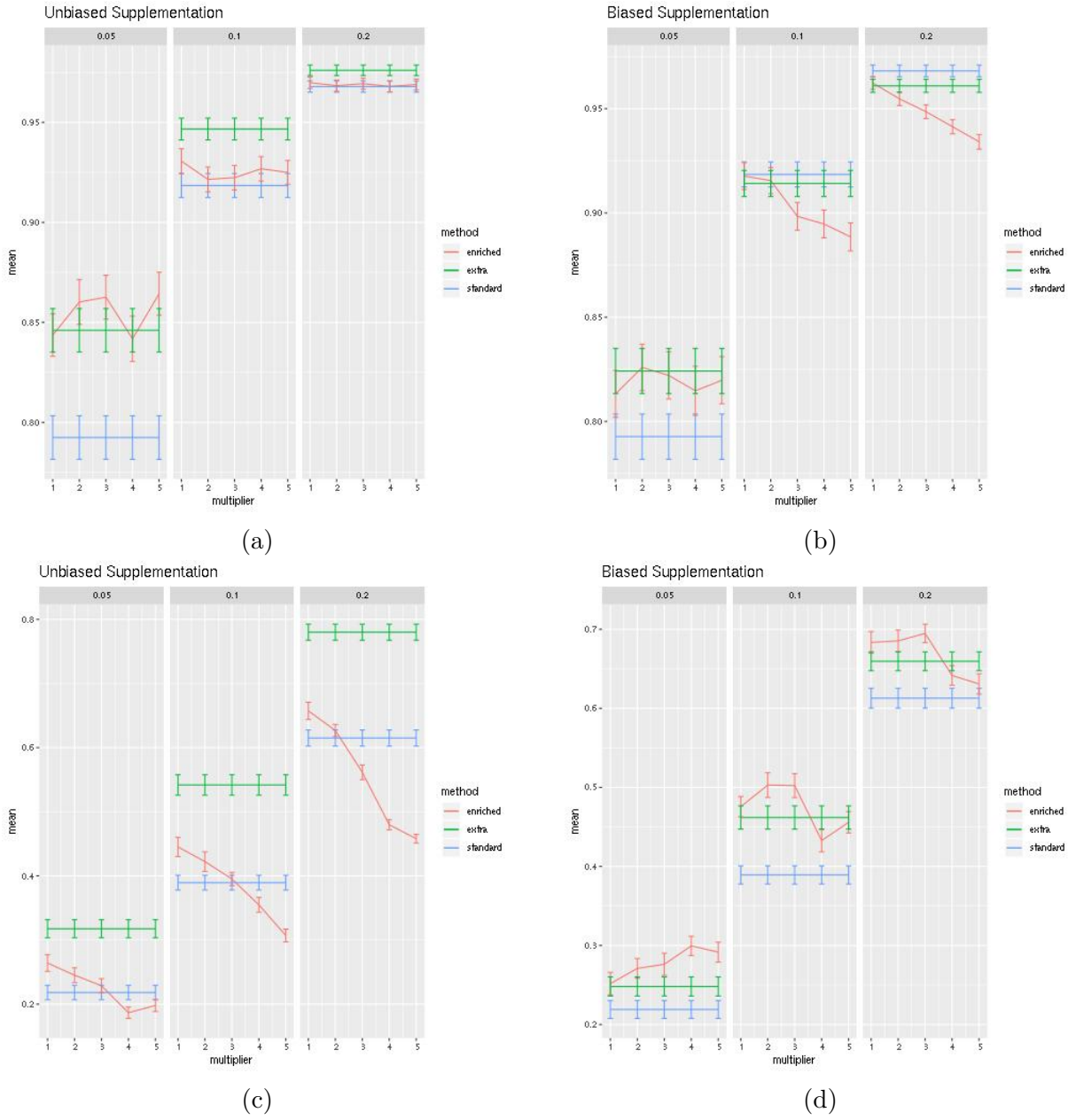


Figure 2.2. Lattice plots summarizing experimental results concerning mean testing PPV vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation.

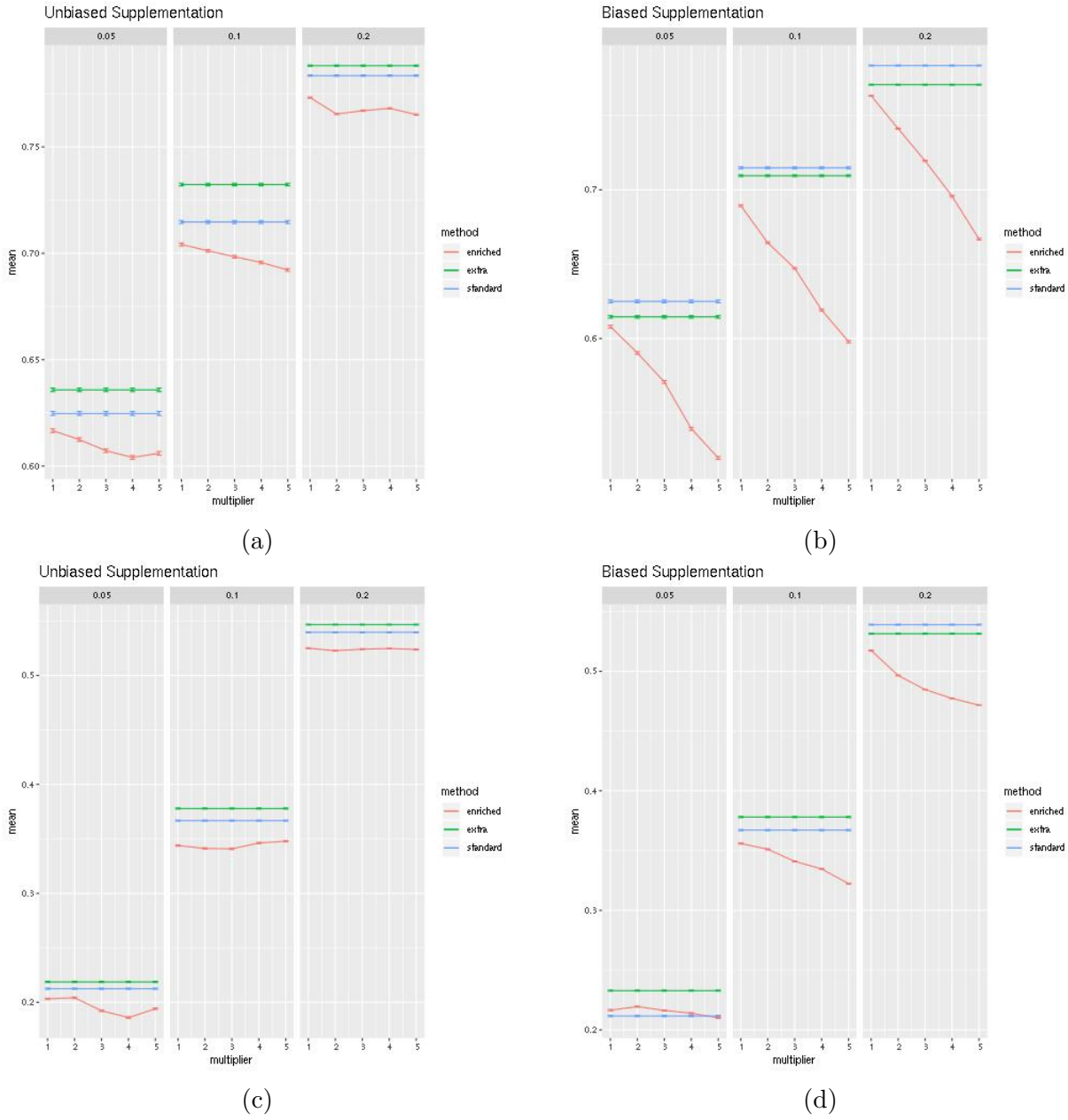


Figure 2.3. Lattice plots summarizing experimental results concerning the mean testing F1 score vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation.

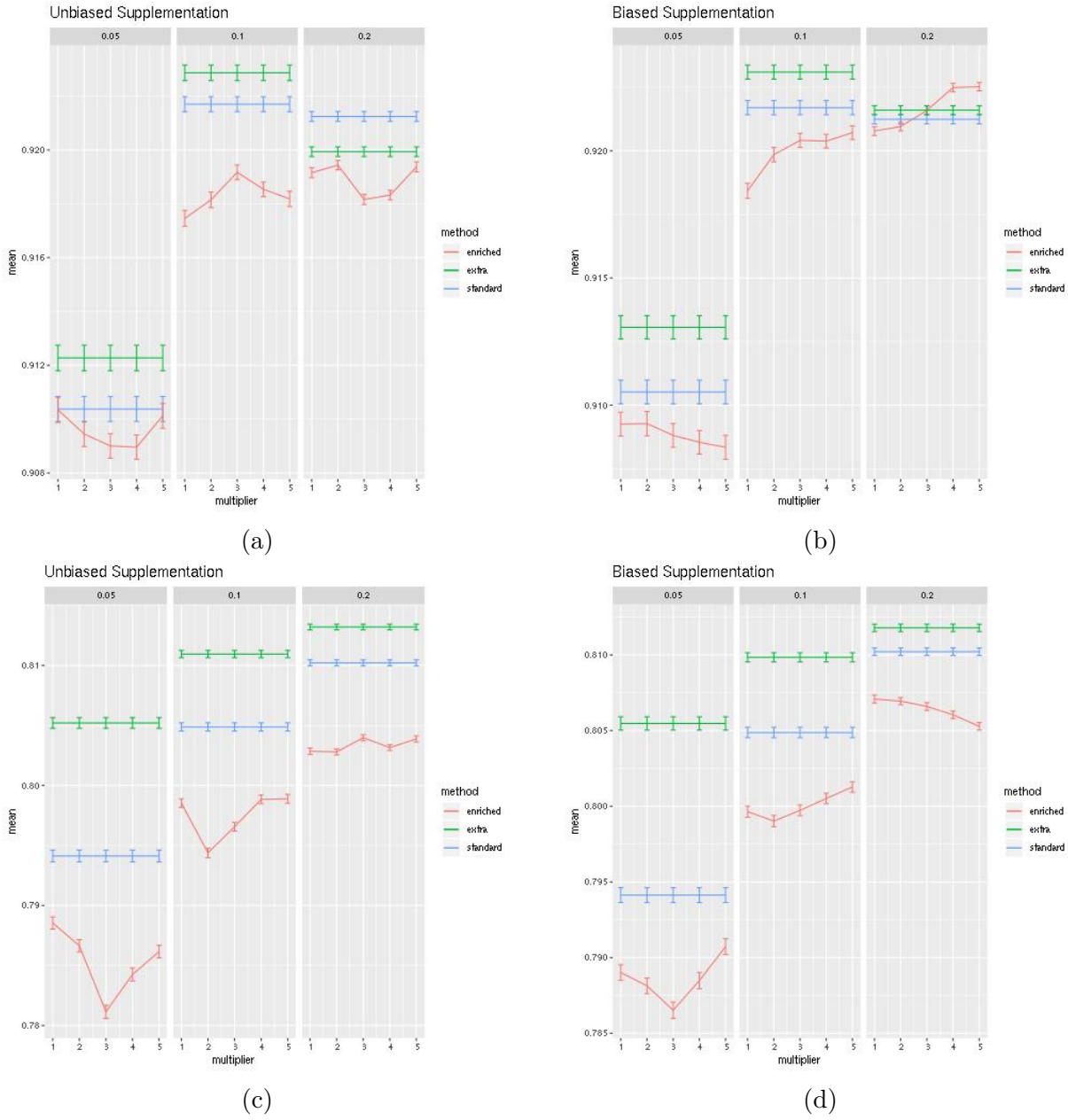


Figure 2.4. Lattice plots summarizing experimental results concerning mean testing AUC vs synthetic multiples for each class imbalance level. The top row displays results for the Mammography dataset while the bottom row displays results for the Phoneme dataset. Red denotes the control method in which synthetic data is not added. Green denotes the collection of *extra* real data. Blue denotes synthetic data supplementation.

2.4 Discussion

The experimental results offer some insight into the feasibility of boosting target statistics with synthetic data supplementation. First, this experimental study offers guidelines to practitioners on synthetic data. Second, the study highlights the feasibility of boosting specific statistics. Lastly, the study provides some evidence to answer whether one should supplement synthetic data instead of collecting more real data.

The most feasible target statistic for synthetic data supplementation is the TPR. While we see mixed effects under the unbiased supplementation scheme, there is dramatic improvement with the biased supplementation scheme. Indeed, 1 or 2 synthetic multiples is sufficient to at least match the benefits gained from collecting additional real data.

The other feasible target statistic, PPV, could be boosted with synthetic data supplementation. Although the biased supplementation scheme once again outperforms the unbiased scheme, the unbiased scheme still provides some benefit. Clearly, we observe that a single synthetic multiple is sufficient to yield some benefit, though not as much as the collection of more real data.

The rest of the results point to how the F1 score and AUC are not feasible for synthetic data supplementation. Evidently, it is difficult to boost these functions of multiple statistics. Accordingly, these results suggest one should seek out more real data to improve the test performance of more complex statistics, such as F1 or AUC.

In all, excluding the targeting of TPR and PPV, the results of the experimental study suggest it is advisable to forgo synthetic data supplementation and instead collect more real data - at least within the confines of the experimental environment and factors. Given that the experiment was conducted on two datasets, future research could involve further experimentation on a greater number and variety of datasets, such as images. Nevertheless, the experimental study suggests 1 or 2 synthetic multiples are sufficient to boost fundamental statistics. In addition, the biased supplementation scheme generally outperforms the unbiased scheme, perhaps explaining the popularity of the biased scheme.

Recall that the experiment defined its gold standard as collecting more real data to double the observed dataset. In other words, the dataset size was doubled. Future research should

consider varying data collection multiples, such as additional data collection to triple or even quintuple the size of the resulting training dataset size. Furthermore, including these varying ratios could then inspire another research question asking about data collection costs versus synthetic data costs. Lastly, it would be interesting to compare an algorithm-level method, such as class-wise weighting of the loss function, to an alternative action of additional data collection or LiCUS or both.

2.5 Summary

This dissertation component has provided an experimental study to examine the effects of class imbalance levels, choice of statistic, and the type of supplementation scheme. Its results have suggested simple statistics are feasible for synthetic data supplementation. On the other hand, functions of multiple statistics are difficult to improve upon. In addition, by proposing a gold standard of additional data collection, the experimental study suggests when one should opt for additional data collection rather than pursue synthetic data supplementation. The experimental study provides some evidence that more data collection is preferable to synthetic data supplementation. Indeed, as data collection becomes cheaper, the relative utility of synthetic data supplementation may very well diminish.

3. LEVERAGING EXPERT KNOWLEDGE FOR GENERATION OF SATELLITE NETWORK TRAFFIC FLOWS

3.1 Introduction

Generation of network traffic flows is instrumental for the development of network security products and systems. In the development and testing process of network security controls, generative methods allow practitioners to gauge credibility. The most common generative strategy involves replaying captured network packets or flows to develop, debug, and test the capabilities of a security control system. Indeed, there are commercial network traffic generation products available to the network security industry that are based on two techniques: (i) captured packet or flow replay and (ii) generation of synthetic data. The reliability of each of these techniques are naturally dependent on the availability of large repositories of previously captured data. Consequently, there is great demand for generation of synthetic network flows.

In response, generative deep learning has become a popular tool for automatic, realistic generation of network flows [49]. Generative adversarial networks (GANs) have become a popular tool for its practicality in network traffic environments [50]–[52]. GANs have demonstrated their versatility with applicability to multiple tasks: balancing imbalanced datasets [53]; evasion of censorship by a network [54]; signal spoofing, or impersonation of an agent or transmitter [54]; evading and development of intrusion detection systems (IDSs) [55]. Interestingly, the discriminator or critic of a GAN could be leveraged for detection, tracking, and classification of unmanned aerial vehicles (UAVs) [56].

While there are plenty of publicly available network traffic datasets, it is not guaranteed to resemble a practitioner's dataset at hand, let alone for the public datasets to pass certain quality controls [52]. Specific types of networks require fabricated packet data satisfying the appropriate network protocol types and headers. Furthermore, synthetic data must exhibit specific characteristics of the target network. A generative method must respect the permissible communications between agents in the network; e.g., specific addresses can only transmit to particular addresses according to precise ports and protocols. This may not be easily learned by a generative model. Indeed, applying a standard GAN on NASA's Opera-

tional Simulation for Small Satellites (NOS3) satellite emulator [57] fails to acknowledge the rules of the satellite network system.

Furthermore, in contrast to open systems, such as internet network systems, there are very few packet and flow generators available for industrial control systems network performance and threat evaluation within closed systems. Indeed, complex closed systems have unknown, if not obscured, network traffic characteristics, protocols, services and security vulnerabilities. As a result, network packet and flow generators are not expected to be useful for closed systems. Instead of relying on captured replay, we propose network traffic generation through an expert-informed Wasserstein GAN with Gradient Penalty (WGAN-GP) [58].

There are many adversarial deep learning approaches to supplement an intrusion detection system [59]. However, to the best of our knowledge, generative models have not been aided by existing intrusion detection systems. By leveraging cyber-security software, we inject expert knowledge into the generative process. Specifically, by exporting a foundational cyber-security method, one could translate expert knowledge as a probabilistic prior or regularization. This would theoretically enable the generative model to explicitly respect the rules of a network. We demonstrate the utility of this approach with application on the NOS3 satellite emulator with expert knowledge extracted from Argus, an open-source network flow data system [60].

3.2 Background

3.2.1 Generative Adversarial Networks (GANs)

Before discussing related work, it may be helpful to first introduce the popular generative method, generative adversarial networks (GANs) [2]. Based on game theory, a GAN consists of two competing, or adversarial, neural networks: the generator and the discriminator. For illustration, we consider some dataset $\{(x_{\mathrm{i}}, y_{\mathrm{i}})\}_{1\leq \mathrm{i}\leq n}$ where $x_{\mathrm{i}} \in \mathbb{R}^d \forall \mathrm{i} = 1, \ldots, n$. These neural networks are mathematically defined below with L layers with p nodes each and

respective element-wise activation functions g_G and g_D for the generator G and discriminator D.

$$z_1^G(x) = W_1^G x \tag{3.1}$$

$$z_{\mathrm{j}}^G(x) = W_{\mathrm{j}}^G g_G(z_{\mathrm{j}-1}^G(x)) \forall \mathrm{j} \in \{2, \ldots, L\} \tag{3.2}$$

$$G(x) = z_L^G(x) \tag{3.3}$$

where $W_1^G \in \mathbb{R}^{p \times d}$, $W_{\mathrm{j}}^G \in \mathbb{R}^{p \times p} \forall \mathrm{j} \in \{2, \ldots, L-1\}$, and $W_L^G \in \mathbb{R}^{d \times p}$.

$$z_1^D(x) = W_1^D x \tag{3.4}$$

$$z_{\mathrm{j}}^D(x) = W_{\mathrm{j}}^D g_D(z_{\mathrm{j}-1}^D(x)) \forall \mathrm{j} \in \{2, \ldots, L\} \tag{3.5}$$

$$D(x) = g^*(z_L^D(x)) \tag{3.6}$$

where $x \in \mathbb{R}^d$, $W_1^D \in \mathbb{R}^{p \times d}$, $W_{\mathrm{j}}^D \in \mathbb{R}^{p \times p} \forall \mathrm{j} \in \{2, \ldots, L-1\}$, $W_L^D \in \mathbb{R}^{1 \times p}$, and g^* is chosen such that $D(x) \in [0, 1] \forall x \in \mathbb{R}^d$.

The GAN structure is an excellent fit for cyber-security as practical applications involve at least two agents defending or attacking communication networks. For instance, the generator network could try to evade a defensive agent while the discriminator network could try to detect anomalous or intrusive network communication. As the discriminator tends to be a binary neural network classifier, the discriminator classifies fake data with a value of 1 and real data with a value of 0. On the other hand, the generator attempts to create data that would be classified as real via gradient back-propagation. In practice, each network is trained while the other network is fixed. This is summarized in the following optimization:

$$\min_G \max_D \mathbb{E}_{X \sim p_{\mathrm{real}}}[\log(D(X))] + \mathbb{E}_{Z \sim \boldsymbol{\pi}}[\log(1 - D(G(Z)))] \tag{3.7}$$

where G and D respectively denote the generator and discriminator networks, p_{real} denotes the theoretical distribution of real data, $\boldsymbol{\pi}$ denotes the prior distribution for the generator network, and X, Z are random variables respectively following $p_{\mathrm{real}}, \boldsymbol{\pi}$.

However, it is not uncommon to fail to train a GAN to convergence. Specifically, GANs may suffer from modal collapse [61], in which the generator exclusively restricts its attention to a subset of the dataset. For instance, if a GAN experienced modal collapse on images of animals, the GAN could only be generating cats and dogs while ignoring all the other classes of animals.

There have been many proposed solutions to avoid modal collapse. For instance, the bidirectional GAN (BiGAN) [62] modifies the discriminator network to jointly classify a datum along with its encoded representation. Another solution, the energy-based GAN (EBGAN) [63], involves replacing the discriminator's classification network with an auto-encoder. It is argued that the discriminator's reconstruction loss is more informative than a simple binary classification. Lastly, Wasserstein GAN (WGAN) [64] replaced the discriminator with a 1-Lipschitz critic neural network C such that:

$$|C(x) - C(y)| \leq |x - y| \forall x, y \in \mathbb{R}^d \tag{3.8}$$

Instead of a binary classification, the critic network outputs a continuous score. The authors provide theoretical support that WGAN should avoid modal collapse altogether.

Interestingly, while many GAN variants have been applied to communication networks [50], WGAN-GP [58] seems to be the most popular tool [51]. While WGAN assumes the critic network is 1-Lipschitz, WGAN-GP regularizes the critic network, C, such that the L2-norm of its gradient is expected to be 1. This results in the following optimizaiton:

$$\inf_{G} \sup_{||C|| \leq 1} \mathbb{E}_{X \sim p_{\text{real}}}[C(X)] - \mathbb{E}_{Z \sim \boldsymbol{\pi}}[C(G(Z))] + \lambda \mathbb{E}_{\tilde{X} \sim p_{\tilde{x}}}[(||\nabla_{\tilde{X}} C||_2 - 1)^2] \tag{3.9}$$

where C is the critic neural network, λ is the regularization hyper-parameter, and $\tilde{X} = qG(Z) + (1-q)X$ such that $q \sim \text{Unif}([0,1])$, $Z \sim \boldsymbol{\pi}$, and $X \sim p_{\text{real}}$.

3.2.2 Related Work

There are a handful of popular generative application categories concerning network activity [50]: mobile networks, network analysis, Internet of Things, physical layer commu-

nications, and cyber-security. However, this dissertation chapter focuses on closed networks. By definition, data on closed networks are not very accessible, possibly explaining why data generation of closed networks has not been widely published. Though, there is an instance of applying GAN on closed military network data [65]. Nevertheless, the aforementioned work translates network data into images while our work does not. In fact, to the best of our knowledge, satellite network traffic has not been studied as a potential application for a GAN.

There are a few works that assume a network is attacking or defending via a neural network classifier and thus include this classifier into a generative methodology [50], [55], [66], [67]. Perhaps the most relevant work to our work, a black-box IDS was probed to train a WGAN in real-time [68]. The trained WGAN managed to evade detection across varying types of IDSs. While these works include an external adversarial classifier, we include an internal cooperative classifier. Indeed, our proposal grants access to many tools by incorporating a transparent and accessible developed cyber-security software program. We select Argus [60], an open source network activity auditing tool. We will discuss our use of Argus more extensively in Section 3.3.2.

$$g(x) = \frac{\mathrm{e}^x}{1 + \mathrm{e}^x} \tag{3.10}$$

Lastly, a previous work [51] considered many GAN variants to generate synthetic data for network analytical tasks and found WGAN & WGAN-GP tended to be most advantaged in terms of training and convergence. Accordingly, we employ a WGAN-GP in our work. On a related note, it is common and natural to model categorical attributes of network traffic with sigmoid activation functions (Equation 3.10). Though, an interesting approach involves learning embeddings for categorical attributes and later employ those learned embeddings to train a WGAN-GP [69]. Instead of relying on sigmoid activation functions or learning embeddings to sample categorical features, we opt for seamless integration into a GAN framework via Gumbel Softmax categorical generation [70], which will be described in the next section.

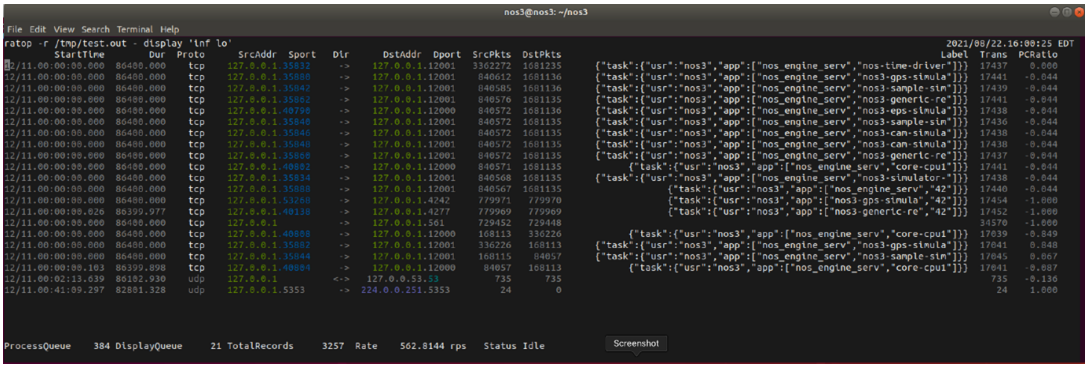


Figure 3.1. An example of NOS3 [57] traffic captured and organized by Argus [60].

3.3 Insertion of Expert Knowledge in WGAN-GP

3.3.1 NASA Operational Simulation for Small Satellites (NOS3)

Satellite network data consists of the observed communication across multiple satellite components. Each transmission represents a specific service, perhaps unique to an individual sender or recipient. Many transmissions occur in a few seconds with meaningful content across multiple satellite components.

Complex closed satellite systems are subject to compromise by external network-based threats, exploitation of embedded vulnerabilities, and insiders - all of which contribute to a difficult cyber-defense problem. As satellite systems are generally unavailable for experimentation, there is limited information available to assess threats against such assets. In practice, there is low or non-existent accessible volumes of unclassified data available for security development for satellite systems.

Without access to observed satellite data, we elected to use the NOS3 satellite emulator [57] as a surrogate. While the NOS3 emulator is not a perfect implementation of the network components of an actual satellite, it has relevance as a network testbed for security control development for a theoretical satellite platform.

To represent an idle satellite orbiting the Earth without any commands or tasks to process, we collected a network flow data every 5 seconds over a 24-hour simulation period. This generated a set of 353,810 network flow status records for 18 simulated network connections,

capturing over 200 network features for each network activity; however, only 4 categorical features and 10 quantitative features were retained. The categorical 4-tuple consists of the protocol type, source address, destination address, and destination port. On the other hand, the quantitative features include the transmission start time, duration, source packets, destination packets, source bytes, destination bytes, source application bytes, destination application bytes, source inter-arrival packet time, and destination inter-arrival packet time. These factors were retained due to their importance in anomalous classification of network flows.

3.3.2 Expert Knowledge via Argus

We emulate expert knowledge in the form of a well-known intrusion detection method known as network activity baselining [71]. Specifically, anomalous classification of a network flow occurs when the flow does not match entries in a defined baseline. For instance, we could construct a baseline consisting of unique categorical flow signatures, namely our 4-tuple definition. Such a network activity baseline for the NOS3 satellite emulator is depicted in Figure 3.2.

In other words, the network activity baseline serves as a whitelist. We will reference this method as baseline anomaly detection or baseline matching. Given a network flow, baseline anomaly detection checks whether the flow signature is on the baseline whitelist, a list of permissible agent-to-agent communications. If a network flow is observed and its flow signature is not found on the baseline whitelist, the network flow is declared to be anomalous.

Rank	StartTime	Dur	Proto	SrcAddr	Dir	DstAddr	Dport	SrcPkts	DstPkts	DstBytes	SAppBytes	DAppBytes	SIntPkt	DIntPkt
1	12/11.00:00:00.00	86400.00	tcp	127.0.0.1	->	127.0.0.1	12001	11431809	17063622	1996438824	913946645	870239748	90614.22	60682.78
2	12/11.00:00:00.00	86400.00	tcp	127.0.0.1	->	127.0.0.1	12000	1933313	3866610	534429796	160800551	279233536	178550.72	89266.67
3	12/11.00:00:00.00	86400.00	tcp	127.0.0.1	->	127.0.0.1	4242	779971	779970	1458610147	0	1407132127	110768.02	110767.50
4	12/11.00:00:00.02	86399.98	tcp	127.0.0.1	->	127.0.0.1	4277	779969	779969	1458608044	0	1407130090	110767.99	110767.51
5	12/11.00:00:00.00	86400.00	tcp	127.0.0.1	->	127.0.0.1	561	729452	729448	515101791	135	466958199	236949.57	236950.88
6	12/11.00:02:13.63	86102.93	udp	127.0.0.1	<->	127.0.0.53	53	735	735	84379	40659	53509		
7	12/11.00:41:09.29	82801.33	udp	127.0.0.1	->	224.0.0.251	5353	24	0	0	1080	0		

argus.nos3.training.features

Figure 3.2. An example of a network flow baseline. Note how the rows list unique categorical flow signatures.

Baseline anomaly detection is provided by Argus software [60] and leveraged via L2 regularization on the critics scores in WGAN-GP. If one were to train a WGAN-GP on the NOS3 dataset, the critic network, on average, scored real flows with a value of approximately +2 and -2 for fake flows. This is empirical behavior of a converged WGAN-GP with the selected architecture defined in the following section after being trained on the NOS3 dataset. We could use this empirical evidence as a starting point to define Argus output. Additionally, recall that WGAN-GP relies on a gradient penalty. As such, valuation of Argus output will certainly affect gradients and thus back-propagation of both neural networks. With this in mind, Argus was instructed to score flows with categorical signatures approved by the baseline anomaly detection with +2 and -2 otherwise. In all, our regularized WGAN-GP is summarized by the following optimization problem:

$$\inf_{G} \sup_{||C||\leq 1} \mathbb{E}_{X\sim p_{\text{real}}}[C(X)] - \mathbb{E}_{Z\sim \pi}[C(G(Z))] + \lambda \mathbb{E}_{\tilde{X}\sim p_{\tilde{x}}}[(||\nabla_{\tilde{X}} C||_2 - 1)^2] + \gamma R_{\text{Argus}}(C, G) \quad (3.11)$$

where

$$R_{\text{Argus}}(C, G) = \mathbb{E}_{Z\sim \pi}[(C(G(Z)) - \text{Argus}(G(Z)))^2] + \mathbb{E}_{X\sim p_{\text{real}}}[(C(X) - \text{Argus}(X))^2] \quad (3.12)$$

3.3.3 Model Architecture

Multiple candidate architectures were considered, ranging from single layer to four layer deep neural networks and number of nodes per layer$\{64, 128, 256, 512, 1024\}$. In addition, multiple values for λ, γ were considered: 0.1, 1, and 10. Lastly, to automate tuning of the learning rate and to include second-order gradient movement, neural networks were trained via the AdaDelta optimization framework [72] with a smoothing parameter with values 0.1, 0.5, and 0.9. All network architectures employed the hyperbolic tangent function (Equation 3.13) as the activation function.

A grid search was performed over all combinations of these tuning parameters. Then, network activity baselines were constructed on 1,000 generated network flows from each candidate model architecture. Although these WGAN-GP architectures failed to create physically possible network activity baselines, we identified the network architecture with the

most realistic network activity baseline and without modal collapse. Specifically, we continue by defining each neural network with four 256-node layers, regularization parameters λ, γ are both set to 10, and the AdaDelta optimization framework has smoothing parameter valued at 0.10.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.13}$$

Each categorical variable will be represented by a single one-hot vector consisting of indicator variables for each factor level. These one-hot vectors will be sampled via Gumbel Softmax categorical generation [70]. The categorical generator learns parameter μ_l to generate score $S_l \sim \text{Gumbel}(\mu_l, 1)$ for level l. By construction, once the scores are fed into the softmax activation function (Equation 3.14), we obtain a discrete probability distribution on the factor levels (Equation 3.15). We generate each categorical random variable according to this sampling scheme.

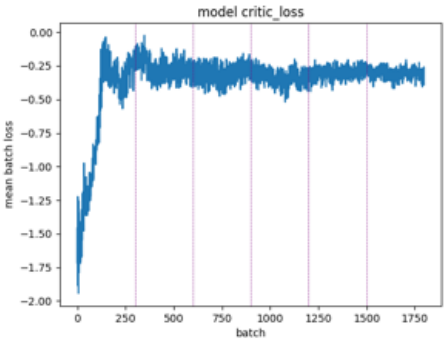
$$g(x_l; x_1, \ldots, x_L) = \frac{e^{x_l}}{\sum_{k}^{L} e^{x_k}} \tag{3.14}$$

$$P(S_l = \max_k S_k) = \frac{e^{S_l}}{\sum_{k}^{L} e^{S_k}} \tag{3.15}$$

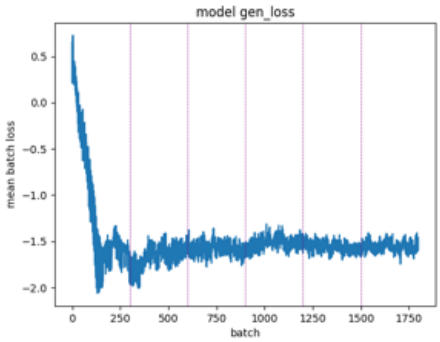
Our framework has a critic loss (Figure 3.3a), generator loss (Figure 3.3b), gradient penalty (Figure 3.3c), and L2 loss between the critic and Argus' baseline anomaly detection (Figure 3.3d). We also provide a plot detailing the critic's scores across fake and real network flows in Figure 3.4. Lastly, we provide a randomly generated sample of 5 network flows in Table 3.1.

3.4 Discussion

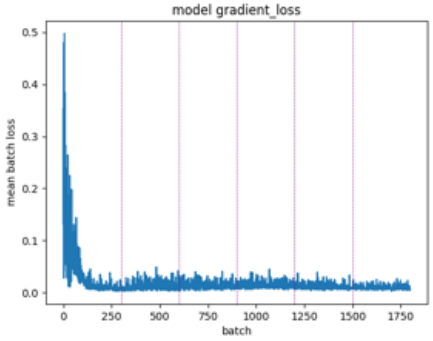
Initial implementations of WGAN-GP generated poor results as its generated flows were non-sensical in terms of its flow signatures. However, the expert-informed WGAN-GP learned how to generate permissible network flows. Hence, baseline anomaly detection



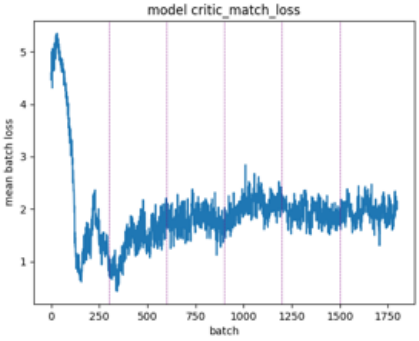
(a) Critic loss, namely fake critic scores minus real critic scores. An ideal critic minimizes the score while an ideal generator maximizes it.



(b) Generator loss, namely the negative value of fake critic scores. An ideal generator would minimize this loss.



(c) Gradient norm penalty on the critic. To remain in the dual model space, an ideal critic would exhibit controlled gradients.



(d) L2 loss between Argus output and critic scores. With the intention of minimizing discrepancies between anomaly baseline detection, an ideal critic would minimize this loss.

Figure 3.3. The training process is indexed by batches, or samples, of $\sim 1,000$ flows. Vertical lines denote transitions between epochs, a marker of having trained on all available data ($\sim 300,000$ flows).

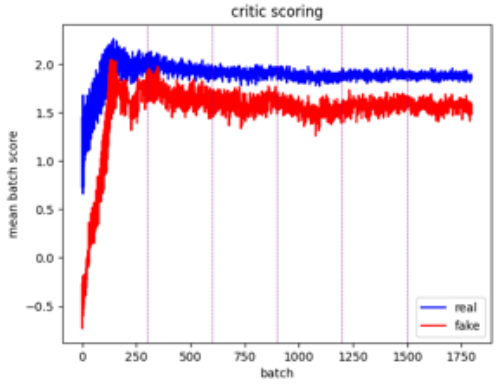


Figure 3.4. Illustration of critic scoring of observed and generated flows. An ideal critic would maximize observed flow scores and minimize generated flow scores. On the other hand, an ideal generator would maximize generated flow scores. The training process is indexed by batches, or samples, of ∼1,000 flows. Vertical lines denote transitions between epochs, a marker of having trained on all available data (∼300,000 flows).

Table 3.1. Five randomly generated network flows from the expert-informed WGAN-GP. The top table displays the categorical flow signatures while the bottom table displays the quantitative network flow features. It should be noted that destination port 12001 occurs 60% of the time.

Proto	SrcAddr	DstAddr	Dport
6	2130706433	2130706433	12001
6	2130706433	2130706433	12001
6	2130706433	2130706433	12001
6	2130706433	2130706433	12001
6	2130706433	2130706433	12001

StartTime	Dur	SrcPkts	DstPkts	SrcBytes	DstBytes	SAppBytes	DAppBytes	SIntPkt	DIntPkt
1607694279	4.952441185	47	96	7450	13678	4072	4130	106029.2709	78529.74905
1607692556	4.951366268	48	97	7035	13344	3877	5447	112117.2319	85453.27129
1607688345	4.918831774	50	94	7046	14270	4054	5266	112174.2422	96940.17833
1607685087	4.958401674	49	96	6971	12869	4139	6358	114013.1046	63265.784
1607675648	4.847237564	50	92	6873	13449	3791	6663	131123.9808	67681.5489

evidently delivered much needed expert knowledge. When the critic was regularized by the baseline anomaly detection output, the critic had to agree with the industry standard anomaly detection system, at least to some extent. For smaller amounts of real data (10k-100k thousand flows), this expert-informed WGAN-GP achieved successful training and did not generate any anomalous flows, at least according to the baseline matching anomaly detection system. With the generation of feasible and permissible flows, this Argus-driven guidance proves instrumental to training a WGAN-GP. Lastly, although the critic scores both real and fake network flows with similar scores, Figure 3.4 displays a visually noticeable gap between the two distributions. This implies that the WGAN-GP has not generated perfectly indistinguishable network flows. In an effort to close this gap, future research could employ expert knowledge on quantitative flow behavior.

3.5 Summary

Armed with expert knowledge from Argus, WGAN-GP became capable of modeling permissible network flows. Indeed, with as little exposure to about 10k observed network flows, our method did not generate any anomalous flows. Therefore, it is reasonable to claim our approach may be trained with minimal sets of data. As such, leveraging expert knowledge has increased the scope of network flow generation in less accessible or active networks. In short, this application implies that our method would provide realistic generated data for many aspects of satellite communications and richer volumes of traffic, such as Internet traffic.

3.6 Acknowledgements

The TCARSS technology was developed using Government support under a collaboration with National Technology & Engineering Solutions of Sandia, LLC, manager and operator of Sandia National Laboratories owned by the United States Department of Energy/National Nuclear Security Administration. The Government has certain rights in the TCARSS technology.

REFERENCES

[1] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3.06, 2014.

[3] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, PMLR, 2015, pp. 1530–1538.

[4] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *International Conference on Machine Learning*, PMLR, 2016, pp. 1747–1756.

[5] M. Esmaeilpour, N. Chaalia, A. Abusitta, F.-X. Devailly, W. Maazoun, and P. Cardinal, "Bi-discriminator GAN for tabular data synthesis," *Pattern Recognition Letters*, 2022.

[6] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] G.-B. Huang, L. Chen, C. K. Siew, *et al.*, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[8] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[9] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970. DOI: 10.1093.

[10] D. J. Geman and S. A. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984. DOI: 10.1109.

[11] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, IEEE, vol. 2, 2004, pp. 985–990.

[13] S. Ickin, K. Vandikas, F. Moradi, J. Taghia, and W. Hu, "Ensemble-based synthetic data synthesis for federated QoE modeling," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, IEEE, 2020, pp. 72–76.

[14] S. Bourou, A. El Saer, T.-H. Velivassaki, A. Voulkidis, and T. Zahariadis, "A review of tabular data synthesis using GANs on an IDS dataset," *Information*, vol. 12, no. 09, p. 375, 2021.

[15] I. B. Mustapha, S. Hasan, H. Nabus, and S. M. Shamsuddin, "Conditional deep convolutional generative adversarial networks for isolated handwritten Arabic character generation," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1309–1320, 2022.

[16] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," *arXiv preprint arXiv:1706.02633*, 2017.

[17] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 104–12 114, 2020.

[18] N. Mayer, E. Ilg, P. Fischer, *et al.*, "What makes good synthetic training data for learning disparity and optical flow estimation?" *International Journal of Computer Vision*, vol. 126, no. 9, pp. 942–960, 2018.

[19] J. Tremblay, A. Prakash, D. Acuna, *et al.*, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.

[20] S. Y. Park and A. K. Bera, "Maximum entropy autoregressive conditional heteroskedasticity model," *Journal of Econometrics*, vol. 150, no. 2, pp. 219–230, 2009.

[21] X. Li, S. Wang, and Y. Cai, "Tutorial: Complexity analysis of singular value decomposition and its variants," *arXiv preprint arXiv:1906.12085*, 2019.

[22] Y. LeCun and C. Cortes, *MNIST Handwritten Digit Database*, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[25] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, p. 386, 1958.

[26] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[27] C. Phua, D. Alahakoon, and V. Lee, "Minority report in fraud detection: Classification of skewed data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 50–59, 2004.

[28] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, "Effective detection of sophisticated online banking fraud on extremely imbalanced data," *World Wide Web*, vol. 16, pp. 449–475, 2013.

[29] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, and A. Geissbuhler, "Learning from imbalanced data in surveillance of nosocomial infection," *Artificial Intelligence in Medicine*, vol. 37, no. 1, pp. 7–18, 2006.

[30] R. B. Rao, S. Krishnan, and R. S. Niculescu, "Data mining for improved cardiac care," *Acm Sigkdd Explorations Newsletter*, vol. 8, no. 1, pp. 3–10, 2006.

[31] R. C. Prati, G. E. Batista, and M. C. Monard, "Data mining with imbalanced class distributions: Concepts and methods.," in *IICAI*, 2009, pp. 359–376.

[32] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[33] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[35] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, Springer, 2009, pp. 475–482.

[36] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *International Conference on Intelligent Computing*, Springer, 2005, pp. 878–887.

[37] S. Ando and C. Y. Huang, "Deep over-sampling framework for classifying imbalanced data," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 10*, Springer, 2017, pp. 770–785.

[38] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[39] H. Lee, M. Park, and J. Kim, "Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning," in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3713–3717.

[40] S. Pouyanfar, Y. Tao, A. Mohan, *et al.*, "Dynamic sampling in convolutional neural networks for imbalanced data classification," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, IEEE, 2018, pp. 112–117.

[41] I. Shumailov, Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot, and R. Anderson, "The curse of recursion: Training on generated data makes models forget," *arXiv preprint arXiv:2305.17493*, 2023.

[42] Q. Bertrand, A. J. Bose, A. Duplessis, M. Jiralerspong, and G. Gidel, "On the stability of iterative retraining of generative models on their own data," *arXiv preprint arXiv:2310.00429*, 2023.

[43] A. Gelman, A. Jakulin, M. G. Pittau, and Y.-S. Su, "A weakly informative default prior distribution for logistic and other regression models," *The Annals of Applied Statistics*, vol. 2, no. 4, pp. 1360–1383, 2008.

[44] D. M. Green, J. A. Swets, *et al.*, *Signal detection theory and psychophysics*. Wiley New York, 1966, vol. 1.

[45] K. S. Woods, C. C. Doss, K. W. Bowyer, J. L. Solka, C. E. Priebe, and W. P. Kegelmeyer Jr, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 06, pp. 1417–1436, 1993.

[46] L. Shen, R. M. Rangayyan, and J. L. Desautels, "Shape analysis of mammographic calcifications," in *Proceedings Fifth Annual IEEE Symposium on Computer-Based Medical Systems*, IEEE Computer Society, 1992, pp. 123–124.

[47] P. Alinat, "Periodic progress report 4, ROARS project ESPRIT II-Number 5516," *Technical Thomson Report TS ASM 93/S/EGS/NC*, vol. 79, 1993.

[48] T. Hastie, R. Tibshirani, and R. J. Tibshirani, "Extended comparisons of best subset selection, forward stepwise selection, and the lasso," *arXiv preprint arXiv:1707.08692*, 2017.

[49] M. R. Shahid, G. Blanc, H. Jmila, Z. Zhang, and H. Debar, "Generative deep learning for Internet of Things network traffic generation," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, IEEE, 2020, pp. 70–79.

[50] H. Navidan, P. F. Moshiri, M. Nabati, *et al.*, "Generative adversarial networks (GANs) in networking: A comprehensive survey & evaluation," *Computer Networks*, vol. 194, p. 108 149, 2021.

[51] J. J. Aho, A. W. Witt, C. B. Casey, N. Trivedi, and V. Ramaswamy, "Generating realistic data for network analytics," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, IEEE, 2018, pp. 401–406.

[52] T. J. Anande and M. S. Leeson, "Generative adversarial networks (GANs): A survey of network traffic generation," *International Journal of Machine Learning and Computing*, vol. 12, no. 6, pp. 333–343, 2022.

[53] Z. Wang, P. Wang, X. Zhou, S. Li, and M. Zhang, "FLOWGAN: Unbalanced network encrypted traffic identification method based on GAN," in *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, IEEE, 2019, pp. 975–983.

[54] J. Li, L. Zhou, H. Li, L. Yan, and H. Zhu, "Dynamic traffic feature camouflaging via generative adversarial networks," in *2019 IEEE Conference on Communications and Network Security (CNS)*, IEEE, 2019, pp. 268–276.

[55] M. Usama, M. Asim, S. Latif, J. Qadir, *et al.*, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2019, pp. 78–83.

[56] C. Zhao, C. Chen, Z. Cai, M. Shi, X. Du, and M. Guizani, "Classification of small UAVs based on auxiliary classifier Wasserstein GANs," in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 206–212.

[57] NASA, *Nasa Operational Simulation for Small Satellites (NOS3)*. [Online]. Available: https://github.com/nasa/nos3.

[58] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[59] G. Abdelmoumin, J. Whitaker, D. B. Rawat, and A. Rahman, "A survey on data-driven learning for intelligent network intrusion detection systems," *Electronics*, vol. 11, no. 2, p. 213, 2022.

[60] L. QoSient, *Argus, copyright (c) 2000 - 2011*. [Online]. Available: https://openargus.org/.

[61] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[62] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[63] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[64] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 214–223.

[65] Y.-b. Park, S.-u. Shin, and I.-s. Lee, "A study on evaluation method of NIDS datasets in closed military network," *Journal of Internet Computing and Services*, vol. 21, no. 2, pp. 121–130, 2020.

[66] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 1, pp. 2–14, 2018.

[67] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative adversarial network for wireless signal spoofing," in *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*, 2019, pp. 55–60.

[68] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2022, pp. 79–91.

[69] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, 2019.

[70] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[71] U.S. Department of Defense, *Advanced Cyber Industrial Control System Tactics, Techniques, and Procedures (ACI TTP) for Department of Defense (DoD) Industrial Control Systems (ICS)*. [Online]. Available: https://www.acq.osd.mil/eie/Downloads/IE/ACI%20TTP%20for%20DoD%20ICS_Rev_2_(Final).pdf.

[72] M. D. Zeiler, "AdaDelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

VITA

Daniel Antonio Cardona

Education

- 2016 - B.S. in Biometry & Biological Statistics, Cornell University
- 2019 - M.S. in Mathematical Statistics, Purdue University
- 2024 - Ph.D. in Statistics, Purdue University

Fields of Study

- Generative Modeling
- Reinforcement Learning
- Synthetic Data

With Applications In

- Computer Network Traffic
- Cyber-security
- Industrial Control Systems