INSIDER THREAT DETECTION DATA AUGMENTATION USING WCGAN-GP

by

Mack Preston

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2022

Table of Contents

List of Tables

List of Figures

Abstract

This thesis explores the application of Generative Adversarial Networks (GANs) in augmenting insider threat detection datasets to alleviate class imbalance. In addition, a machine learning based insider threat detection system is proposed that augments datasets to improve detection rates while maintaining precision. WCGAN-GP, a promising new GAN variant, is trained on a publicly available synthetic insider threat dataset and used to generate realistic samples for multiple insider scenarios. The generated samples are used to augment the dataset, which is then used to train supervised classifiers to detect insider threats. The WCGAN-GP based augmentation strategy outperforms the baseline (under-sampled) strategy on a large feature set, increasing the detection rate while preserving a low false-positive rate. The framework was further tested on two later versions of the dataset which contain modified behaviour and new insider scenarios. The results show that the proposed approach is robust and can generalize to novel insider threat scenarios.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Nur Zincir-Heywood, whose support and guidance was paramount to the completion of this work. I would also like to express my gratitude to Dr. Duc C Le, whose previous work on insider threat detection was foundational to the research laid out in this thesis. Finally, I would like to thank Dr. Malcolm Heywood and Dr. Xiao Luo for their timely review and constructive feedback on this work.

Chapter 1

Introduction

Insider threats are a growing cyber-security concern for organizations in both public and private sectors. These threats can range from disgruntled IT administrators sabotaging core infrastructure, to opportunistic defectors ex-filtrating customer leads before joining a rival firm. This is a challenging topic due to highly imbalanced data, limited ground truth and possible changes to user and organizational behavior over time. The behaviour of both regular users and insider threats can take many forms, making it difficult to build a detection solution that can generalize without flagging benign users. This prompted the application of numerous machine learning methods, including Generative Adversarial Networks (GANs).

GANs were originally proposed in the field of computer vision and quickly gained popularity due to their ability to generate hyper-realistic images. Since then, they have been applied to fields ranging from music generation, drug discovery and text-to-image synthesis [21]. GANs have also been used within the cyber security domain [10]. Some of these applications include stenography, password cracking, malware detection and even malware synthesis. GANs have a wide range of uses within cyber security and privacy because they effectively model the adversarial nature of the fields themselves. Cyber Security is often viewed as a game of cat and mouse, where an attacker is always looking for new ways to compromise a defender's systems. In response, the defender must find ways to curb the novel offences and create robust obstacles against future attacks. This paradigm is also mirrored by the adoption of red teams and blue teams within organizations. Red teams attempt to emulate realistic threats facing organizations, actively looking for exploitable gaps in their systems. On top of detecting legitimate threats to an organization, blue teams can evaluate their performance and adjust accordingly based on feedback from the red team's exercises. This has the added benefit of fine-tuning the allocation of finite security resources and funding to best address real threats. These manifestations of attack

and defense are perfect analogs for the generator and discriminator sub-networks of the GAN. The generator aims to generate samples that fool the discriminator, while the discriminator attempts to spot the fake samples from the real. Training these networks together as adversaries results in increasingly realistic generated samples and ever-more robust detection of fakes.

The primary research objective of this thesis is to explore the ability of GANs to augment insider threat datasets to improve supervised classification performance (particularly on minority insider classes). To this effect, recent advances in GAN architectures are applied to solving the class imbalance problem of insider threat detection. Namely, the WCGAN-GP GAN variant is leveraged to generate realistic samples for multiple insider scenarios that can be used to augment insider threat datasets. This work also proposes a high-level framework that detects insider threats by training supervised classifiers on augmented datasets. Existing preprocessing techniques are utilized to reduce complex log-based user behaviour and organizational structure to daily feature vectors [27]. The publicly available CERT insider threat datasets were used to train the proposed framework, as well as test its performance and robustness across different organizations and scenarios [28]. To the best of my knowledge, this is the first work to evaluate the ability of WCGAN-GP to augment insider threat detection datasets. Chattopadhyay et al. have recently evaluated multi-class augmentation of the CERT R4.2 dataset using CGAN, but I believe this thesis to be the first work to evaluate the robustness of CERT R4.2 trained GANs against the CERT R5.2 and R6.2 datasets [14]. In doing so, this work evaluates the robustness of the augmentation strategies against different organizational structures and novel insider threats.

The remainder of this thesis will consist of the following structure. Chapter 2 explores some background, introduces the datasets and discusses related works. Chapter 3 describes the methodology used to perform the experiments and Chapter 4 presents and discusses the results of those experiments. Finally, conclusions are drawn and directions for future work are summarized in Chapter 5.

Chapter 2

Background and Related Work

This chapter reviews the background for this thesis, as well as related works. Section 2.1 provides an overview of the insider threat datasets used in this work. The broader problem of class imbalance is discussed in Section 2.2. Section 2.3 goes over some other successful approaches to insider threat detection, including unsupervised and supervised methods. Section 2.4 gives a broad overview of how GANs have been applied to other problems within cyber security. Finally, Section 2.5 reviews applications of GANs in data augmentation, including a comparison between the contributions of this thesis and similar works.

2.1 CERT Insider Threat Dataset

One of the greatest challenges in insider threat detection is finding a rich dataset on which to train and test models. This is partially because of the legal and privacy concerns associated with collecting detailed user information in the real world. Furthermore, organizations are unlikely to publish this collected data as it would make it easier for attackers to evade their protections. Another challenge is just how rare insider events are in contrast to regular user events. For these reasons, synthetic datasets are often used to benchmark detection algorithms.

The CERT Insider threat dataset is a synthetic dataset consisting of HTTP traffic, emails, PC logons, USB file transfers and psychometric profiles [28], [15]. It also provides a corresponding LDAP dataset that captures the structure of an organization with 1000 to 4000 employees over the course of 18 months. To make the dataset more realistic, Glasser et al. introduced latent variables aimed to model real user behaviour. For example, they created a realistic social network graph that drives behaviour like who sends emails to whom. Red-team exercises and profiles of real-world insider events were used to define the behaviour of the synthetic users. The dataset was released in versions (R4.1-R6.2), as more information, features and insider

scenarios were added to the models. CERT R4.2, R5.2 and R6.2 contain three, four and five insider scenarios respectively. R4.2 and R5.2 contain a much higher ratio of insider to normal events than would be found in a real world dataset. R6.2 attempts to create a more realistic balance by drastically increasing the number of benign events while reducing the instances of insider events. In this thesis, R4.2 is used for training purposes while R5.2 and R6.2 are used to test the robustness of the proposed approach.

2.2 Class Imbalance

Class imbalance occurs when a dataset has a highly skewed distribution of samples, where one or more minority classes have considerably fewer samples than the majority classes. This is a core problem of insider threat detection as normal user activity is considerably more frequent than malicious insider activity. Some of the simplest methods for overcoming this problem are oversampling and undersampling. Oversampling involves introducing multiple copies of the minority class samples into the dataset in an attempt to balance the class counts prior to classification. Unfortunately, this can lead to classifiers overfitting to samples from the minority class. Inversely, undersampling involves only including a sub-set of samples from the majority class to balance the class counts. The downside of this approach is that the sampling of the majority class might not be representative of the real-world distribution, as key data points could be discarded. Take the insider threat detection problem for example; If normal user behaviour data is under-sampled to match the insider behaviour class counts, all samples for IT administrators could be missed. This could lead to a system that flags every every IT administrator as an insider threat for legitimately accessing multiple PCs. This work combines basic under-sampling of the normal class and over-sampling of the insider classes to create an augmented dataset to improve insider detection. The algorithms used to over-sample are described in Sections 3.3.1 and 3.3.2.

This work uses a simple random under-sampling technique to decrease the prevalence of the majority normal class in an attempt to balance the classes. However, many more sophisticated techniques such as tomek-link undersampling have been used to further improve classification performance on imbalanced datasets [11]. Tomek

link undersampling clarifies the border between the majority and minority classes by pairing up each minority class sample with its nearest majority class neighbour. The majority class sample is then deleted from the dataset for each pair, thus under-sampling the majority class. Alternatively, a clustering approach like the one used by Chattopadhyay et al. could be used to ensure the variance in normal behaviour modalities are well represented in the under-sampled dataset [7].

2.3 Insider Threat Detection Approaches

Many approaches have been taken to this problem, ranging from supervised learning models, unsupervised models, and semi-supervised models. Neural networks, Random Forest, Logistic Regression and XGBoost classifiers have been evaluated against different temporal pre-processing granularities of CERT R4.2, R5.2 and R6.2 [27]. Evolutionary computation methods have also been used as classifiers under streaming conditions to adapt as user behaviour changes over time [23], [24]. Isolation Forests (IF), Auto-encoders (AE), Lightweight On-line Detector of Anomalies (LODA) and Local Outlier Factor (LOF) have been shown to be effective as unsupervised anomaly detection methods for detecting insider threats [25], [26]. Their efficacy has proved resilient, even under adversarial conditions where insiders are included in the training dataset as normal exemplars. The performance of these models can be further improved by combining them into a vote-based unsupervised ensemble [26].

Unsupervised anomaly detection approaches have been extremely effective when applied to the insider threat detection problem because they can be trained on normal non-malicious user behaviour. While insider samples are exceedingly rare in real-world scenarios, normal samples are abundant and readily available to train detection systems. Auto-encoders for example, have been one of the most successful anomaly detection approaches for insider threat detection [25]. They are a type of neural network that consist of three components: an encoder, a bottleneck layer and a decoder. Training this network rewards the network's ability to compress the representation of the input data through the bottleneck layer and reconstruct it using the decoder. This forces the network to generalize from the examples it is trained on. This model can be turned into an anomaly detector by passing samples through the network and comparing the reconstruction error to a threshold value. Any sample

that produces a reconstruction error over this threshold will be flagged as an anomaly for further inspection by a domain expert. This is a powerful approach because it leverages the vast number of benign samples and only requires the insider samples for validation and testing.

2.4 GANs in Cyber Security

Msika et al. used GANs to make machine-learning based Intrusion Detection Systems (IDS) more robust against new attack types [31]. This was accomplished by retraining the machine learning models against adversarial examples generated by GANs. Although GANs can be used to strengthen cyber security defense systems, they can also be used to attack them. Chauhan et al. show the vulnerability of existing IDS systems to attacks generated by GANs [8]. GANs have also been proposed to improve the cyber defenses of critical infrastructure. Adiban et al. propose a STEP-GAN system that employs multiple generators in strengthening a discriminator to detect attacks aimed to disrupt smart power grids [1]. Singh et al. use GANs to generate synthetic malware images to build datasets that can be publicly shared and used to train malware classifiers [32]. Xie et al. show that GANs can also be extended to ensure differential privacy of the data they are trained on while still producing high quality samples [29]. This could be useful in the context of insider threat detection because it could allow for the public sharing of insider datasets without fear of sensitive data being derived from generated samples as a result of GAN training. These are just few examples from a wide range of applications of GANs within cyber security and privacy. Several literature surveys are available for a more exhaustive exploration of their applications to date, [10] [12] [6].

2.5 GANs for Data Augmentation

Data augmentation is a common data science practice, often used to counteract problems like class-imbalance for classification tasks. It can also help regularize a dataset and prevent models from overfitting. The ability of GANs to generate varied yet realistic samples makes them prime candidates to produce augmentation data. Zheng et al. propose using the WCGAN-GP architecture for handling imbalanced datasets

[37]. A similar architecture is used by Walia et al. to augment tabular datasets [34].

A few other recent works have even leveraged GANs to augment the CERT R4.2 insider threat dataset. Yuan et al. use a basic GAN to augment the CERT R4.2 dataset [35]. Their work leverages Long-Short-Term Memory (LSTM) auto-encoders to encode user action sequences. They use the encoder to translate the behaviour into a fixed-length representation that can be used to train GANs for dataset augmentation. They also evaluated the classification performance as the number of generated samples in the augmented dataset was increased, but they stopped at a maximum of 5500 generated samples. This thesis evaluates much higher augmentation rates, measuring the classifier performance on datasets augmented with up to 100000 insider samples from each insider class. A limitation of the work by Yuan et al. is that they used a binary context, with no label conditioning that allows for generating samples of each insider class individually. Further, they did not evaluate the performance of individual insider classes, instead grouping them all together under a binary classification context. This could result in prioritized detection of a majority insider class at the expense of the minority insider classes with fewer training samples. Gayathri et al. recently used CGAN to generate samples for multiple insider classes to improve classification using preprocessing techniques proposed by Chattopadhyay et al. [14],[7]. Similarly to this thesis, Chattopadhyay et al. computed statistical and count based features with a sliding-window to give daily feature vectors for each user. Chattopadhyay et al. employed several time-series analysis metrics such as mean, variance, Katz fractal dimension and spectral power density to measure each user's daily behaviour in relation to previous days within the window. Gayathri et al. use these preprocessing techniques for their CGAN experiments, but do not specify the window size or which of the time-series metrics are used. This puts the feature set size at 20 features, or 80 features if all four time-series metric permutations are used. This is a small feature set compared to the 504 features used in this thesis. In addition, this thesis included normal samples in the conditioned training of the GANs, where as Gayathri et al. trained their CGAN exclusively on insider classes. Notably, Gayathri et al. used a much lower max epoch of 300, and a much larger batch size of 64.

These works have demonstrated the efficacy of basic GANs and CGANs to augment the R4.2 dataset using different preprocessing techniques. A shared limitation of these works is that they only report results for a single run, where as this thesis reports the mean and variance across 10 runs. This additional validation is important given the datasets being tested consist of very few insider samples, making them prone to high variance between runs. This is also important given the well-documented instability of GAN training. A GAN that fails to train consistently might be too risky or unreliable to deploy into a production environment. This thesis further assesses the training stability and performance of multi-class CGAN by applying preprocessing techniques proposed by Le et al. [25]. In addition, this thesis introduces WCGAN-GP as a stable CGAN replacement for multi-class insider threat detection augmentation. The experiments are also extended to the R5.2 and R6.2 datasets to evaluate the robustness of the approach against different organizations and new attacks. Table 2.1 summarizes some of the novel contributions of this thesis, as well as illustrating the key differences between the works by Yuan et al. and Gayathri et al.

Table 2.1: Contribution comparison between this thesis and work by Yuan et al and Gayathri et al.

Component	Contribution	Yuan et al.	Gayathri et al.	This Thesis
User Behaviour Represention	Daily user features	✓	✓	✓
	Action sequence based (using LSTM Auto-encoder)	✓		
	Statistical and count based features (using sliding window)		✓	✓
	Weekends included		✓	✓
	Used large feature set			✓
	Leveraged psychometric data			✓
Data Augmentation	Basic/Vanilla GAN	✓		✓
	CGAN		✓	✓
	WCGAN-GP			✓
	Used SMOTE as a benchmark	✓	✓	✓
	Used Random Over Sampling as benchmark		✓	
	t-SNE plots used to evaluate sample authenticity		✓	✓
	Samples generated for each of the insider classes		✓	✓
	GAN trained on normal samples simultaneously			✓
Classification	Broke down performance by insider class		✓	✓
	Compared performance as # of generated samples increased	✓		✓
	Compared performance for large # of generated samples			✓
	Reported Kappa and MCC metrics		✓	
	Tested against CERT R4.2	✓	✓	✓
	Tested against CERT R5.2			✓
	Tested against CERT R6.2			✓
Overall	Results validated by running experiments 10x			✓
	Evaluated GAN training stability			✓
	Compared different feature sets			✓
	Evaluated ability to detect novel insider scenarios			✓
	Compared different network sizes and params			✓

Chapter 3

Methodology

This thesis proposes a system for detecting insider threats in a supervised context. The proposed system consists of three primary components, namely (1) User Behaviour Representation, (2) Data Augmentation and (3) Classification. Section 3.1 provides a high-level overview of the proposed system. The user behaviour representation used for all experiments in this thesis are discussed in Section 3.2. The primary focus of this thesis involves the exploration of different data augmentation techniques, which are presented in Section 3.3. Finally, classification methodology and performance metric selection are discussed in Sections 3.4 and 3.5 respectively.

3.1 Framework Overview

This research proposes an insider threat detection system consisting of the following three components.

1. **User Behaviour Representation**: each user's behaviour is extracted from logs, daily statistics and counts are calculated and a trailing window is used to compute the percentile of that user's activity each day.

2. **Data Augmentation**: a combination of oversampling and undersampling methods are used to re-balance the dataset for classification.

3. **Classification**: the augmented dataset is used to train classifiers to flag insider threats.

This thesis holds the user behaviour representation constant and compares different oversampling techniques to augment the insider threat detection datasets. Basic undersampling and SMOTE were used as benchmarks to compare against GAN based augmentation strategies. Initially, a basic GAN was implemented to generate insider samples. In order to generate insider samples of a particular class (scenario), the

GAN architecture was replaced with CGAN. After CGAN (like GAN), failed to reliably produce realistic samples of the insider classes, WCGAN-GP was implemented and used for the remaining experiments. These architectures are described further in sections 3.3.2, 3.3.3 and 3.3.4, respectively. Following the generation of additional insider samples, the resulting augmented datasets were used to train classifiers to detect insiders. Figure 3.1 shows how the end-to-end system would be trained to detect insider threats. Figure 3.2 shows how the classifiers trained on the augmented datasets would be used in a production environment to detect insider threats, sending alerts for further investigation.

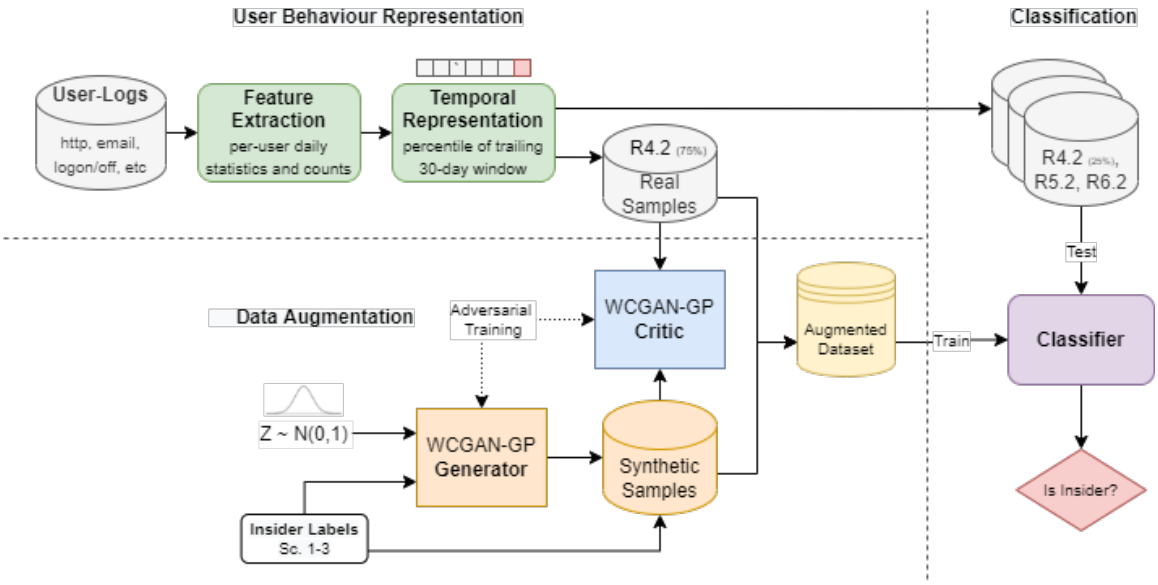


Figure 3.1: Insider threat detection system training

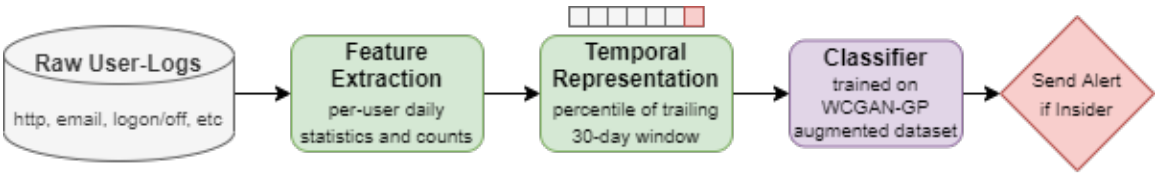


Figure 3.2: Real world application of insider threat detection system

3.2 User Behaviour Representation

Prior to training, the CERT dataset was pre-processed using a technique proposed and implemented by Le et al. [27] [22]. The technique involves the following steps.

1. **Grouping Log Events:** events are parsed from log data and sorted into buckets by user and time. This work used daily time buckets as they were the best performing time-representation in prior works [25].

2. **Feature Extraction:** various permutations of statistical and count-based features are calculated for each day for each user. For example, an employee who visits many job websites from their work computer during the work-day may be more likely to perform malicious insider actions. Table 3.2 shows the raw log-based features and various permutations applied.

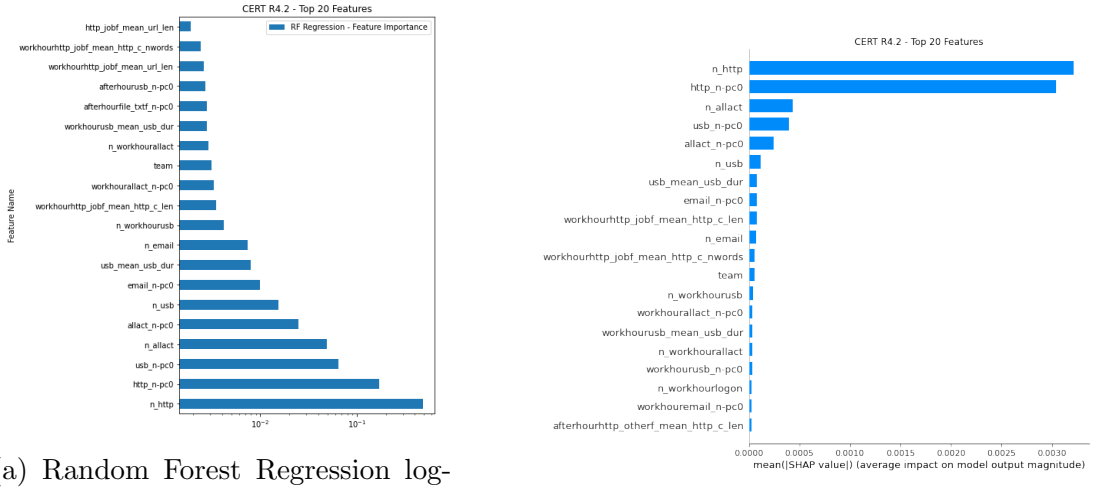
3. **Temporal Representation:** a sliding window is passed over the extracted daily features for each user, computing the percentile of the current day with respect to the trailing window for the same user. This work selected a window size of 30 days and the percentile metric due to its superior performance in prior works [25].

To improve the training stability and performance of the GANs, the features are further scaled from the range (0, 100) to (-1, 1). A useful feature of the trailing window percentile representation is that the data samples are automatically normalized with respect to that user's past behaviour. A limitation of this work is that the same normalization was applied to the organizational and psychometric features despite them being categorical in nature or belonging to a different range. This is not expected to make a large difference, but future works could investigate removing these features or scaling them more appropriately.

504 features were left after the preprocessing of CERT R4.2, including all organizational, psychometric, count-based and statistical features. To simplify the training and prototyping of the various GAN architectures, a reduced feature-set of 100 features was used. Random Forest Regression and SHAP (SHapley Additive exPlanations) were used to rank the importance of the features in predicting the output class. In this case, the top 100 features ranked by Random Forest regression log-scaled

Table 3.1: Psychological and Organizational Features

Psych. Profile	**Organization**	**Time**
Openness	ITAdmin	Weekend
Conscientiousness	Role	Weekday
Extraversion	Team	
Agreeableness	Department	
Neuroticism	Business Unit	
	Functional Unit	



(a) Random Forest Regression log-scaled feature importance

(b) SHAP feature importance

Figure 3.3: Top 20 features by importance for predicting insider class membership

importance were used to construct the reduced feature set. SHAP was only used to validate these top features using an alternate method. The top 20 important features for each algorithm are shown in Figure 3.3. Although CERT R5.2 and R6.2 contain a much larger feature set, only features present in all three datasets were used in this thesis. This allows us to train the classifiers on CERT R4.2 and test them against R5.2 and R6.2 for robustness.

Table 3.2: Feature Permutations

Time	PC	Event Type	Permutations	Raw Features
After Hours Working Hours	User's PC Shared PC Supervisor's PC Other PC	HTTP	Job Site Cloud Site Social Site Hacking Site Other Site Leak Site	# Sites Visited Url Length Url Depth Content Length # Words
		Email	Internal External	# Destinations # BCC # Attachments Email Size Text Length # Words
		File	Compressed Photo Document Text Executable	# Files Accessed
		Device		USB Duration # USB Conns
		Logon		# Logons

Table 3.3: Class counts after preprocessing into daily buckets for each user

		Insiders				
Dataset Name	**Normal**	**Sc.1**	**Sc.2**	**Sc.3**	**Sc.4**	**Sc.5**
CERT R4.2	307057	85	861	20	0	0
CERT R5.2	647441	85	863	20	339	0
CERT R6.2	1304406	3	20	2	1	1

3.3 Data Augmentation

The core of this work involves experimenting with different data augmentation strategies. As seen in Table 3.3, the insider scenarios are vastly outnumbered in the CERT datasets. This results in poor classification performance for the minority classes. Insider scenario 3 has only 20 samples, while the dataset contains over 300000 normal samples. To solve this issue, basic undersampling of the normal class is used for all experiments (including baseline). In addition, SMOTE, GAN, CGAN and WCGAN-GP oversampling are explored. This thesis uses the term "n_samples" to denote both the number of random samples included from the normal class and the number of samples for each insider class after the generated samples are added. For example, an n_samples value of 1000 and original dataset class counts of [307057, 85, 861, 20] would result in an augmented dataset with class counts [1000, 1000, 1000, 1000]. The first class would be undersampled down to 1000 samples, while enough samples would be generated for the remaining three classes to pad their class counts up to 1000 each. For the baseline augmentation strategy however, the first class would be undersampled to 1000 but the remaining classes would be included as is. These oversampling methods are described further in the following sections.

3.3.1 SMOTE

Synthetic Minority Oversampling Technique (SMOTE) is a popular technique for solving class imbalance problems proposed by Chawla et al. [9]. The algorithm works by finding the k-nearest neighbours of a randomly selected sample x and linearly interpolating between x and its neighbour x^k. Equation 3.1 shows how the linearity is calculated and applied to the sample x to create the new sample x' for each feature i.

$$x'_i = x_i + rand(0,1) * |x_i - x_i^k| \tag{3.1}$$

For this work, the python scikit-learn implementation of the SMOTE algorithm was used. The library takes an imbalanced dataset as input and outputs a dataset with enough SMOTE generated samples to bring all class counts up to that of the majority class. For example, given a dataset with counts [1000, 800, 20], SMOTE

would generate 200 and 980 new samples for class 2 and 3 respectively. This results in a dataset with class counts [1000, 1000, 1000].

3.3.2 Generative Adversarial Networks

Generative adversarial networks (or GANs) are a class of neural networks initially proposed by Goodfellow et al. in the field of computer vision [16]. They consist of two sub-networks (a generator and a discriminator) that are trained against each other in a min-max game. The goal of the generator is to generate realistic synthetic samples, while the discriminator aims to differentiate these generated samples from genuine samples. The loss function of the complete network is modeled as a min-max optimization problem as seen in equation 3.2. Goodfellow et al. show that this optimization problem effectively minimizes the Kullback-Leibler (KL) divergence between the probability distribution of the real samples $p_{\text{real}}(x)$ and the probability distribution of the generated samples $p_{\text{gen}}(z)$. This work implements this using binary cross entropy as the loss function. The discriminator D is trained to maximize the log probability of real data and the log inverse probability of synthetic data. The generator G has no effect on the $\mathbb{E}_{x\sim p_{\text{real}}(x)}[\log D(x)]$ term, so only needs to minimize $\mathbb{E}_{z\sim p_{\text{gen}}(z)}[1 - \log D(G(z))]$. The generator is therefor trained to minimize the log probability for synthetic data predicted by the discriminator. Trained together, the generator becomes better at generating realistic samples, and the discriminator becomes more adept at spotting fakes.

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{real}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_{\text{gen}}(z)}[1 - \log D(G(z))] \tag{3.2}$$

GANs are notoriously difficult to train because the gradients for updating the weights of the discriminator and generator are dependant on each other. This leads to problems like mode collapse, vanishing gradients and poor visibility on model performance during training. Mode collapse can occur when the generator converges towards producing the same sample for multiple inputs, instead of generating a diverse range of samples. This can occur if the generator happens across a particularly convincing sample that the discriminator can't differentiate. This can lead to the discriminator getting stuck in a local minima. This is especially detrimental to the

training of the GAN because the generator needs a useful gradient from the discriminator network in order to escape the collapsed mode. Another common issue when training GANs is vanishing gradients. If the discriminator outpaces the generator and $D(x)$ approaches 1, the overall loss will go to zero the generator will no longer have a gradient to climb. An over-arching problem with GANs is the lack of evaluation metrics during training. Since the generator and discriminator are trained based on each others outputs, there is no clear metric with which to measure performance. This means it is hard to evaluate the performance of a GAN during training and makes it hard to establish stopping conditions. When training GANs to generate images, the generated outputs can be visually inspected to qualitatively assess the performance of the GAN and help troubleshoot the above problems. This method is not as feasible in the case of tabular datasets like CERT. Instead, t-SNE plots can be created to visualize the results in lower-dimensions. This is discussed further in Section 3.3.6.

3.3.3 Conditional GANs

Conditional GAN (or CGAN) is a variant of the GAN architecture introduced by Mirza and Osindero [30]. CGANs introduce auxiliary data inputs such as class labels to the discriminator and generator. This has been demonstrated to stabilize the training process and also allows us to select a particular class when generating samples. This is particularly useful when augmenting datasets because it lets us generate samples from minority classes to solve imbalanced data problems. The loss function for CGAN is shown in Equation 3.3. It is identical to the original GAN's loss function, except that the probabilities for the data sample x and noise vector z are conditional on the label y.

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{real}}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_{\text{gen}}(z)}[1 - \log D(G(z|y))] \tag{3.3}$$

3.3.4 Wasserstein GANs

Wasserstein GAN (or WGAN) is another GAN variant proposed by Arjovsky, Chintala and Bottou [4]. WGANs abandon the KL-divergence in favor of Wasserstein distance, also known as the Earth-Movers (EM) distance. Intuitively, earth-movers distance is named after the measure of how much energy it would take to transform a

pile of dirt from the shape of one probability distribution to another. For this GAN, the discriminator is renamed as a critic. This is because instead of "discriminating" the samples as real or fake, it outputs a score for the realness of the sample. This is more stable and tractable as a training problem because even if the critic predicts correctly for all samples, it still provides useful values and gradients for the generator to climb. The min-max objective for WGAN is given in Equation 3.4.

$$\min_{G} \max_{D \in \mathbb{D}} \mathbb{E}_{x \sim p_{\text{real}}(x)}[D(x)] + \mathbb{E}_{z \sim p_{\text{gen}}(z)}[1 - D(G(z))] \tag{3.4}$$

The objective is constructed from the Kantorovich-Rubinstein duality, which has the limitation that $D(x) \in \mathbb{D}$, where $\mathbb{D}$ is the set of 1-Lipschitz functions [18]. The initial WGAN architecture uses gradient-clipping as a method for meeting the 1-lipschitz constraint. This involves clamping the gradients down to a fixed range before every update of the critic's weights. This is a crude technique that can result in vanishing or exploding gradients. Gulrajani et al. proposed gradient-penalty as an alternative, demonstrating it's ability to stabilize training with minimal hyper-parameter tuning [18]. Instead of clipping the gradient, a gradient penalty is calculated as the squared difference from norm 1 and applied to the gradient before updating the weights. WGAN-GP can easily be extended to use the conditioning of CGAN, resulting in a WCGAN-GP architecture as proposed by Zheng et al. [37]. For this thesis, the WGAN and WGAN-GP architectures were skipped and WCGAN-GP was implemented directly.

3.3.5 WCGAN-GP Architecture

The WCGAN-GP architecture for this thesis was implemented using Keras and Tensorflow, with two variations on the network architecture. These networks were given the short-hands DeepV1 and ShallowV1 for easy reference. The parameterizations of these networks are shown in Table 3.4 and their architectures are presented in Figures 3.4 and 3.5. Leaky Rectified Linear Units (ReLUs) were used for the activation function of each hidden layer with the exception of the output layer of the generator (which used tanh). The CGAN implementation used a sigmoid as its discriminator output activation function to bound its discriminations between zero and one. Since WGAN replaces the discriminator with a critic, the sigmoid is replaced with ReLU

so the critic can output scores. To avoid overfitting, dropout layers were added to the generator's hidden layers [34]. Unlike the earlier GAN architectures implemented for this work, WCGAN-GP had stable training and performance across different parameterizations and network sizes. This is consistent with the results in the literature [4] [18]. The observed stability of this architecture is discussed further in Section 4.1.

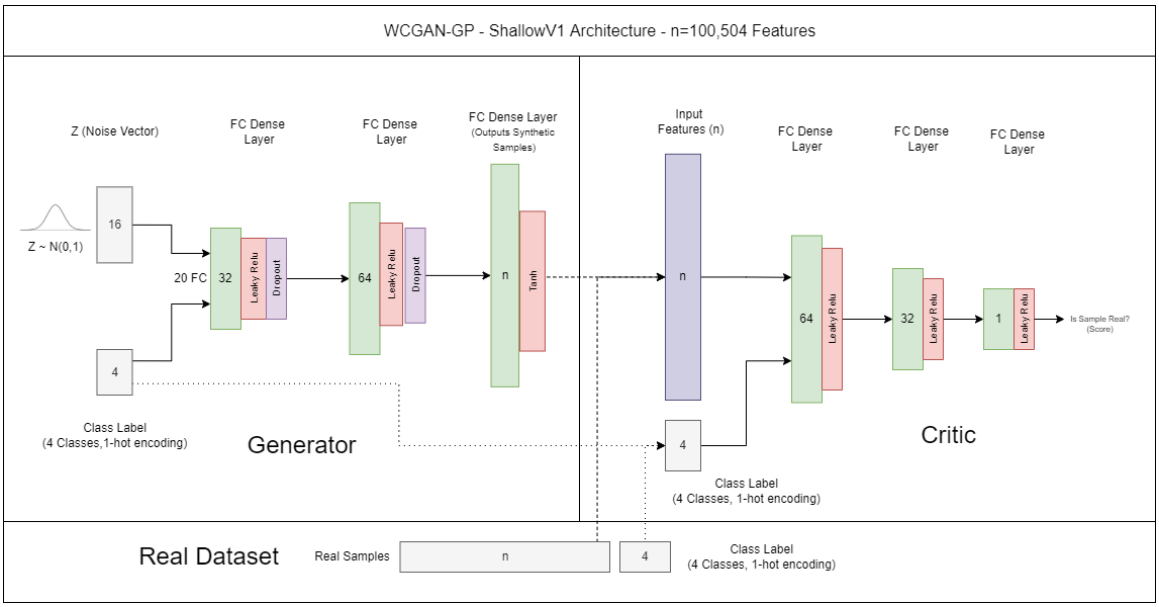


Figure 3.4: WCGAN-GP Shallow Architecture

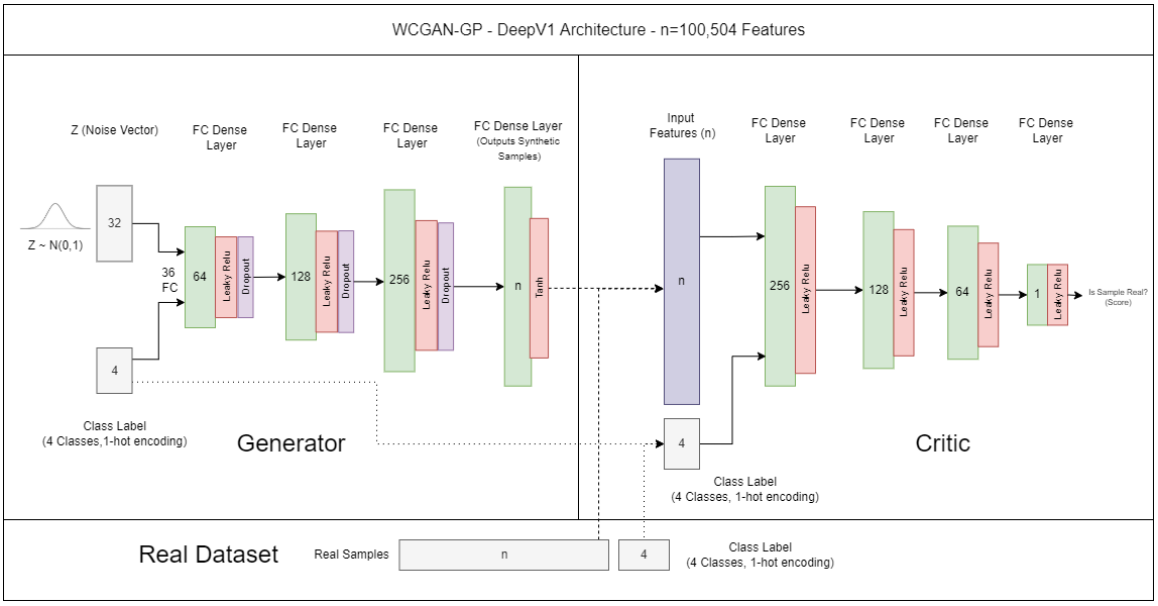


Figure 3.5: WCGAN-GP Deep Architecture

Table 3.4: Augmentation Strategy Parameterization

Strategy	Parameters
SMOTE	k_neighbors=5
WCGAN-GP DeepV1	n_hidden_layers=3 latent_noise_vector_size=32 hidden_layer_sizes=[256, 128, 64]
WCGAN-GP ShallowV1	n_hidden_layers=2 latent_noise_vector_size=16, hidden_layer_sizes=[64, 32]

3.3.6 Synthetic Sample Realness

t-distributed Stochastic Neighbour Embedding (t-SNE) was proposed by Van Der Maaten and Hinton to help visualize high-dimensional data [33]. This involves calculating a joint probability distribution of each data points likeliness to be a neighbour of each other point. Next, a random dataset of points with k dimensions (usually 2 or 3 for visualization) are generated and their neighbour likelihood joint-distributions are calculated. Finally, the values of the generated low-dimensional data points are modified using gradient descent to minimize the KL-divergence between the two joint-distributions. The result is a low-dimensional clustering representation of the high-dimensional dataset. In this thesis, t-SNE is used to visualize the similarity between the original CERT samples and the generated samples.

3.4 Classification

After the R4.2 augmented training datasets were collected, four classifiers were used to test the data augmentation strategies. Logistic Regression, Random Forest (RF), Support Vector Machines (SVM) and Extreme Gradient Boosting (XGBoost) were selected based on their popularity and diversity of approach. The xgboost python package and sklearn python implementations of RF, SVM and Logistic Regression were used for this thesis. Section 4.3 discusses the performance of each of the classifiers in an initial testing phase. The results of those experiments were used to select a top classifier to carry out more in-depth data-augmentation experiments.

After the classifiers were trained on the augmented R4.2 training datasets, their performance was evaluated by predicting the classes of the R4.2 test partition, as well as the entire R5.2 and R6.2 datasets to measure robustness. R5.2 and R6.2 introduce two new insider classes, which were used to evaluate the ability of the system to generalize to never before seen insider attacks.

3.5 Metric Selection

In this thesis, the performance metrics of the classifiers were altered as a hybrid between binary and multi-class classification metrics. The overall performance of an insider threat detection classifier can be simplified as predicting each sample as either an insider or a normal class. However, it is useful to evaluate a classifiers performance on each of the insider classes independently. Instead of using the standard multi-class classification metrics, any insider class sample predicted as a different insider class was still considered a true positive classification. This was done to prevent penalization of classifiers that can ultimately detect insiders, while still capturing class-specific metrics.

Macro-averages for F1-score, recall and precision were used as primary measures of classifier performance, where the scores for each class were averaged without consideration for the support of each class. This is a useful metric in the context of insider threat detection because it rewards detecting the smallest minority classes (such as Insider Scenarios 1 and 3). Optimizing for the detection of minority classes can produce more false positives, but that can be acceptable in an alerting scenario where the threat justifies an increased investigation budget. In this insider threat dataset, IT sabotage only has a support of 20 samples, but could be significantly more costly to the organization than an information leak. A weakness of the macro-metrics is that each insider class is given the same weighting as the majority normal class. This means that a classifier predicting every sample as an insider could be given a higher score than a balanced classifier that finds most insiders without creating too many false positives. This effect became larger as additional insider classes were added in R5.2 and R6.2. Thus, it is important to contextualize the macro-averaged metrics with raw counts and weight-averaged metrics.

To improve on these metrics, future works could compute ROC curves and AUC

measures for each classifier to best understand classifier performance using augmented data. In terms of static measures, the Cohen's Kappa Coefficient and Mathew's Correlation Coefficient (MCC) metrics would more elegantly capture the classification performance on imbalanced multi-class insider threat datasets [14]. These static metrics could be especially useful to reduce the complexity of the analysis while still rewarding classifiers for the detection of minority classes.

Chapter 4

Results and Evaluation

Many augmentation strategies, classifiers, architectures and parameterizations were explored during this thesis. In Section 4.1, the sampling strategy and the stability of the various GAN architectures during training will be explained. From there, the experimental results will be reviewed and discussed chronologically as they were performed. The original GAN implementation performed very poorly, so will not be discussed here (no empirical results were captured). Section 4.2 contrasts the performance of the CGAN implementation against the WCGAN-GP implementation and justifies the selection of WCGAN-GP for further experiments. Section 4.3 explores preliminary WCGAN-GP augmentation performance for multiple classifiers before choosing the Random Forest classifier to carry out more in depth experiments. Section 4.4 does an in-depth review of the experiments performed on the 100 and 504 feature sets; the results of the different augmentation strategies are compared for all three CERT datasets (R4.2, R5.2 and R6.2). Finally, Section 4.5 contrasts the performance of the 100 and 504 feature datasets.

4.1 GAN Training

4.1.1 Sampling Strategy

After pre-processing, a random 25% of the R4.2 dataset (class-stratified) was set aside for testing, and the other 75% was used for training. Next, the WCGAN-GP is trained for 5000 epochs in weighted batches of 16 samples. The batches are constructed using the class frequency weights shown in Table 4.1, where a weight of 3/16 means that class is guaranteed three samples in each batch of 16. To avoid training on the same sample twice in one epoch, the number of batches per epoch is limited by the class with the fewest samples. Insider scenario 3 only has 15 samples in the training set and a class frequency batch weight of 1/16. Each epoch therefore contains 15 batches,

iterating over one sample per batch. The other classes are drawn randomly from the full training partition in correspondence with their weights.

The sampling weights used in the training of the GANs were selected purely based on the batch size and the ratios of class support in the training dataset. The small batch size was selected (as recommended) for maintaining training stability in the earlier GANs, but could potentially be increased to allow the more stable WCGAN-GP to train on more normal class samples. In future works, the insider classes could be over-sampled to allow larger batch sizes or more training iterations per epoch.

Table 4.1: GAN training label sampling weights

Label	**Class**	**Weight**
0	Normal	6/16
1	Insider (Sc.1)	4/16
2	Insider (Sc.2)	5/16
3	Insider (Sc.3)	1/16

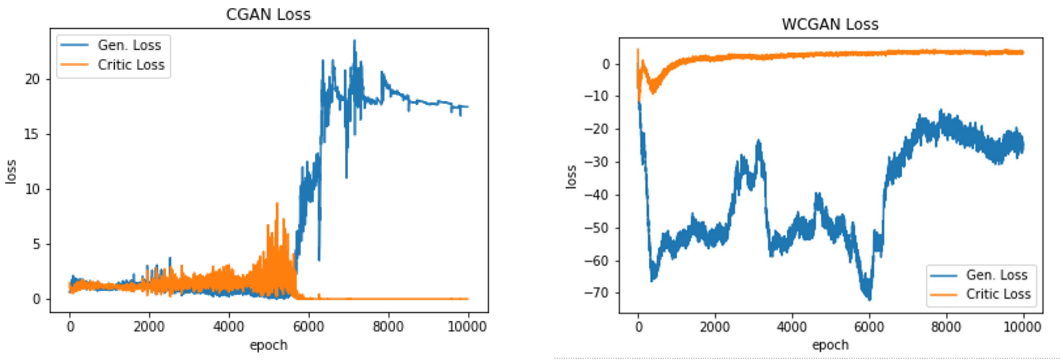
Table 4.2: WCGAN-GP training parameterization

Parameter	**Value**
Epochs	5000
Batch Size	16
Classes	4
Critic Steps	5
GP Weight	10.0
Gen. Dropout Rate	0.3
Leaky ReLU Alpha	0.2
Optimizer	Adam
Learning Rate	0.0001
Beta_1	0.5
Beta_2	0.9

4.1.2 Training Stability

This research quickly encountered the GAN training issues described in the literature when prototyping the initial GAN and CGAN implementations [16] [4]. Figure 4.1 shows sample training curves for CGAN and WCGAN-GP. These are not directly comparable because they use different loss functions. However, they can be used to compare the relative training stability of the two min-max approaches. Unlike

other neural network architectures, a decrease in loss is not necessarily indicative of improved overall performance but rather the performance has improved with respect to its adversary.



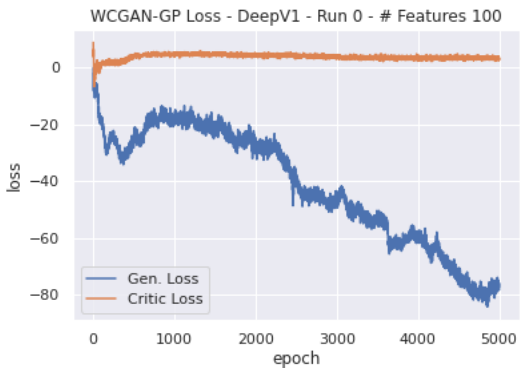
(a) CGAN (b) WCGAN-GP

Figure 4.1: GAN variant training loss curves

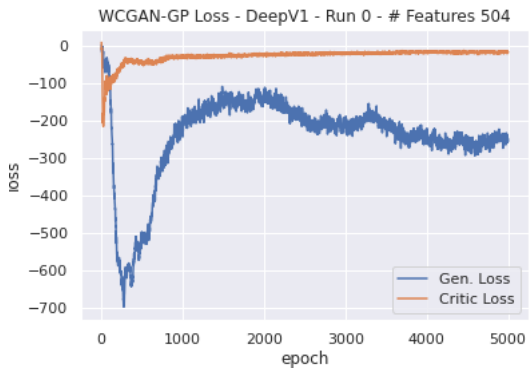
Figure 4.1a shows the binary cross-entropy training loss for the generator and discriminator of a CGAN trained for 10000 epochs. Although both losses fluctuated for 5000 epochs, the discriminator loss approached zero before 6000 epochs as the generator loss faced a sharp hike. For the remaining 4000 epochs, the discriminator loss stays at zero while the generator loss climbs to over four times its peak while the discriminator loss was non-zero. Some runs remained stable throughout the training process, but those that reached a discriminator loss of zero did not recover. The CGAN training was very sensitive to hyper-parameters such as number of features, network layer size and overall network depth; Smaller networks tended to be more stable.

Figure 4.1b shows the Wasserstein training loss for the generator and critic of a WCGAN-GP trained for 10000 epochs. The critic loss dipped initially but quickly stabilized around zero for the rest of the training run. Despite this, the generator loss fluctuated freely and maintained stability throughout every experimental run. When using the CGAN architectures, the training was more likely to become unstable the longer it lasted. For WCGAN-GP, the training was consistently stable, even when increasing training epochs to 20000. The Wasserstein loss metric rewards the critic for predicting higher scores for fake samples and lower scores for real samples, but does not constrain the network to produce scores centered around zero. For this

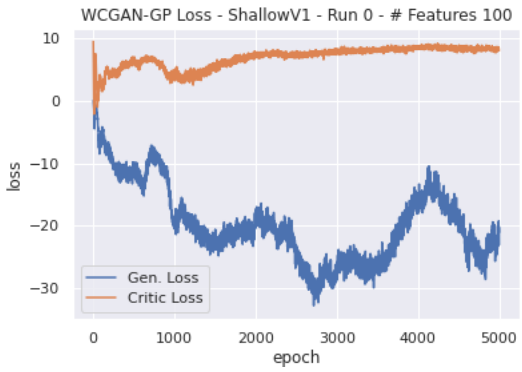
reason, the critic loss may be above or below zero and is not necessarily indicative of performance. This can be seen clearly in Figure 4.2, which shows sample training curves from four different architecture and feature set combinations. In Figure 4.2c, the critic loss converges around nine, while Figure 4.2b converges around -10. Despite changing the architecture, number of features, learning rate, dropout layers, gradient penalty weight and number of critic steps, the training remained consistently stable and resulted in WCGAN-GPs capable of generating convincing samples.



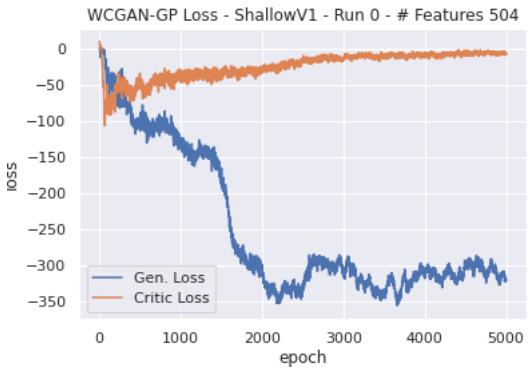
(a) 100 Features - DeepV1



(b) 504 Features - DeepV1



(c) 100 Features - ShallowV1



(d) 504 Features - ShallowV1

Figure 4.2: Sample training curves for WCGAN-GP

This thesis uses t-SNE to gauge the similarity between real data samples and generated samples used to augment the CERT dataset. The parameterizations found in Table 4.3 were used to compute and graph the t-SNE plots every 1000 epochs during training. A perplexity of 40 and the number of iterations of 1000 were selected based on recommendations from the original t-SNE paper [33]. Only a subset of samples were used due to the size of the data set and computational cost of running multiple t-SNE iterations.

Table 4.3: t-SNE parameterization

Parameter	**Value**
Components	2
Perplexity	40
Iterations	1000

A limitation of this thesis is that the t-SNE data was sampled from each class with replacement, resulting in oversampling of the real data. This is particularly noticeable in the minority classes, where single data points are represented by small tightly-clustered groups of points. This could easily be solved in future works by changing the data sampling technique to operate without replacement.

Figure 4.3 shows the t-SNE plot of the CGAN from Figure 4.1a at epoch 5000. The plot shows t-SNE points computed from a 100 feature subset of the training and CGAN generated samples, marked as colour-coded circles and crosses, respectively. Although the training curve was stable when this plot was created, the generated samples do not appear to follow the distributions of the training data. The generated classes are clearly separable from each other, but do not align with the corresponding class samples the CGAN was trained on.

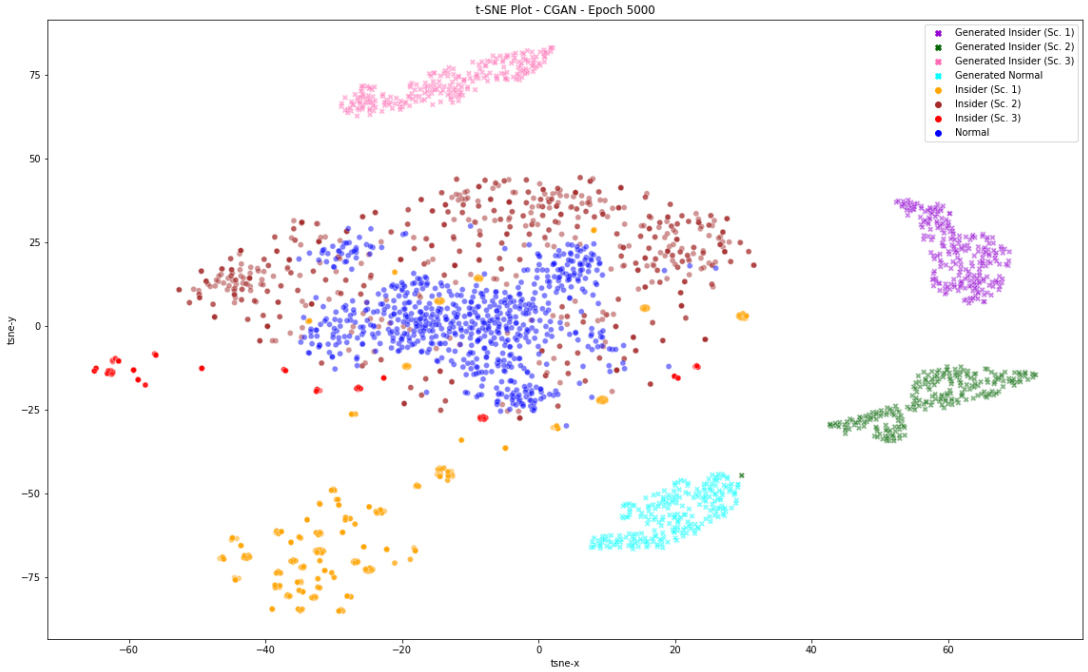


Figure 4.3: CGAN at epoch 5000 - t-SNE plot of generated samples against training samples

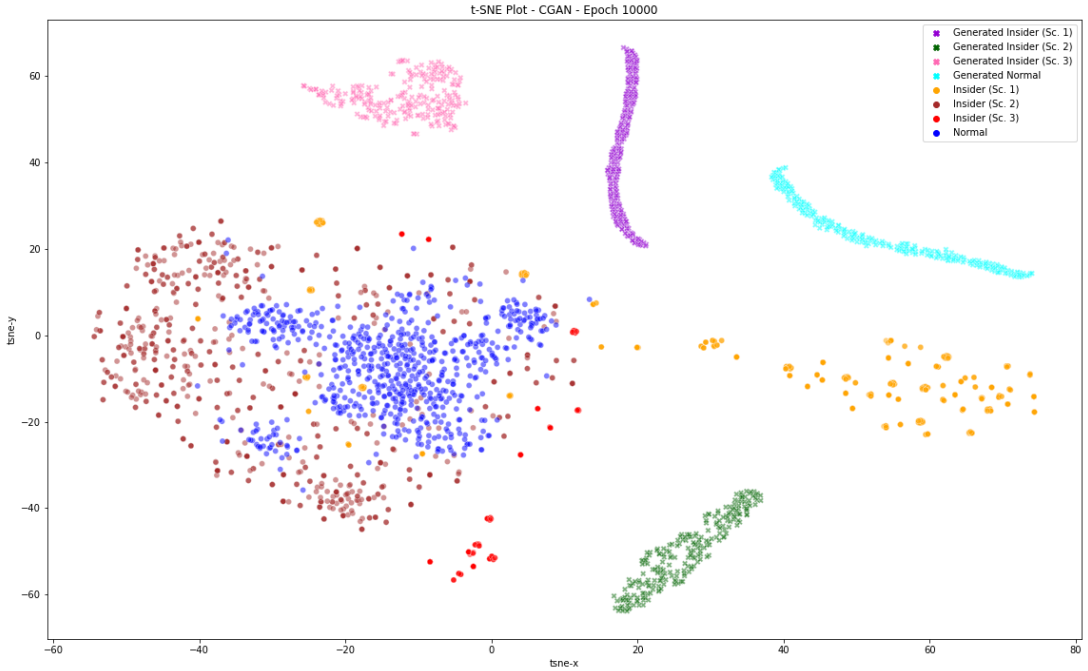


Figure 4.4: CGAN at epoch 10000 - t-SNE plot of generated samples against training samples

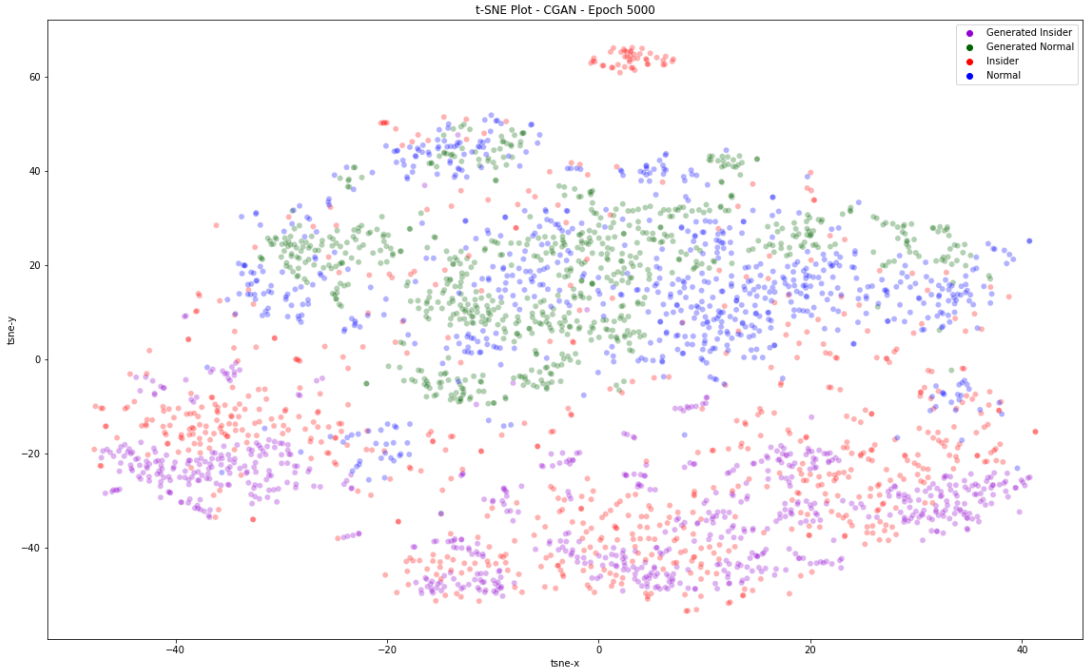


Figure 4.5: CGAN at epoch 5000 - t-SNE plot of generated samples against training samples using only two classes (insider and normal)

Shortly after epoch 5000, the discriminator's loss goes to zero and the generators loss skyrockets. This is likely the result of vanishing gradients, where the discriminator can easily predict all of the fake samples and the generator loses its training gradient. Figure 4.4 shows the same CGAN t-SNE plot at epoch 10000. Since epoch 5000, the generated samples for many of the classes had become clustered into long narrow distributions. This means that the samples were not very diverse, as they were easily mapped into a condensed latent space by the t-SNE algorithm. Figure 4.5 shows the t-SNE plot of a more successful CGAN, which was trained using binary labels (normal and insider) instead of normal and multiple insider classes. However, this method still leaves large clusters of insider samples under-represented in the generated samples. This can be observed at the top-middle of the figure, where a large cluster of insider samples have no generated samples anywhere near them. These most likely belong to one of the minority insider classes (Sc.1 or 3), but could be confirmed in future works by using the multi-class labels to create the t-SNE plot. This means that the oversampling could be missing the classes with the smallest support and could lead to producing an augmented dataset that does insufficiently boosted classification

performance for under-represented insider classes. Further, this method does not give us the capability to generate samples of a particular class at will. As such, this thesis focuses on multi-class dataset augmentation and leaves further comparisons between multi-class and binary sample generation and dataset augmentation to future works.

Figures 4.6 and 4.7 show sample 100 feature t-SNE plots for the ShallowV1 and DeepV1 WCGAN-GP architectures, respectively. Figures 4.8 and 4.9 show the same plots for the 504 feature set. Compared to the CGAN plots, all of the WCGAN-GP plots show a much closer alignment between the generated samples and their real-world counterparts. Interestingly, generated sample distributions for the 504 feature set seem more evenly placed and closely aligned with the real distributions. The DeepV1 architecture also seems to achieve more realistic generated sample distributions than ShallowV1. While the CGAN t-SNE plots varied widely from run to run, WCGAN-GP produced consistent results.

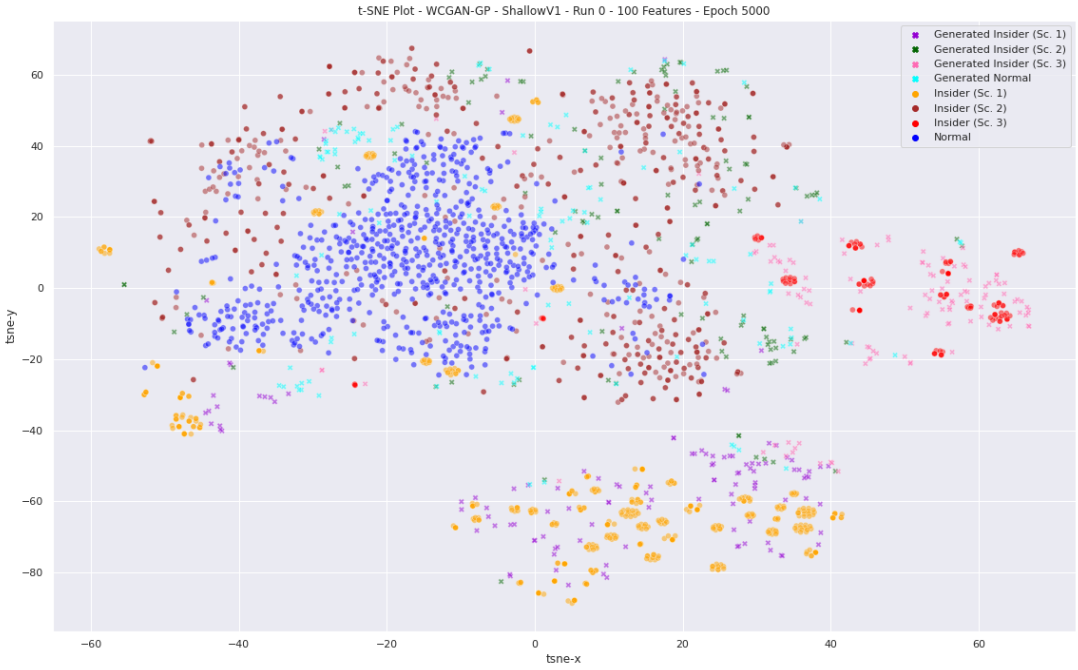


Figure 4.6: WCGAN-GP ShallowV1, 100 features: t-SNE plot of real and generated samples after 5000 epochs of training

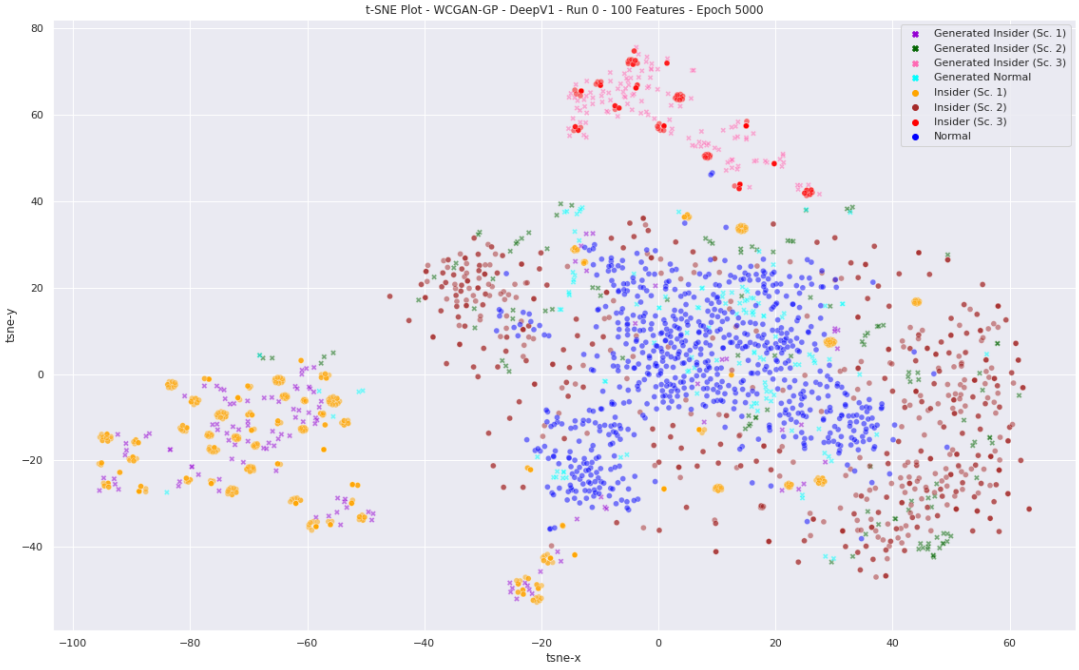


Figure 4.7: WCGAN-GP DeepV1, 100 features: t-SNE plot of real and generated samples after 5000 epochs of training

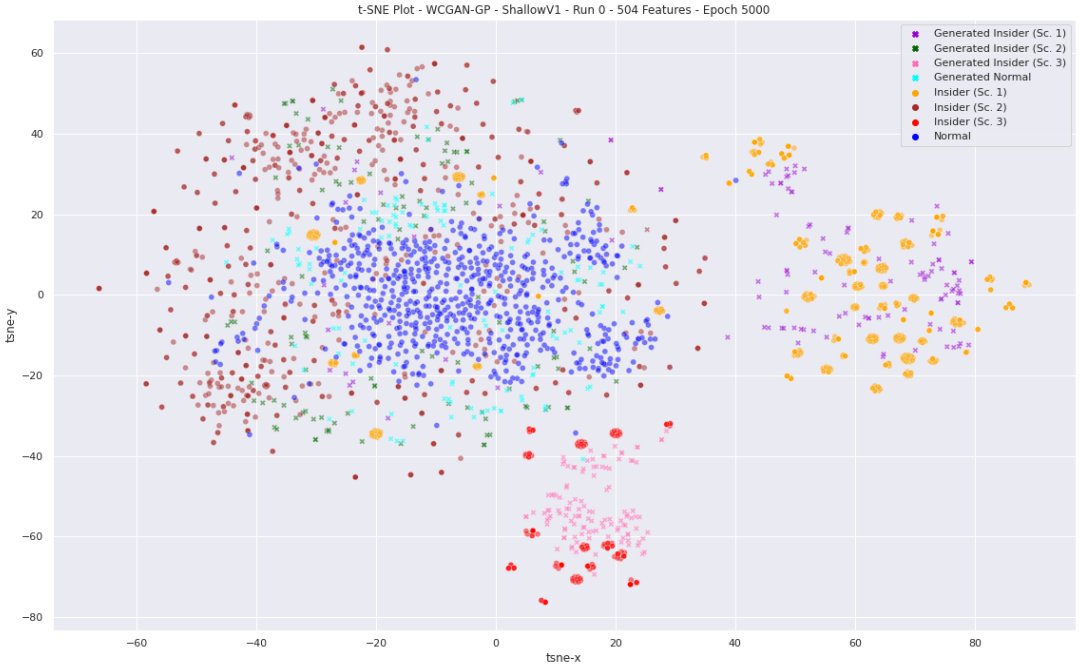


Figure 4.8: WCGAN-GP ShallowV1, 504 features: t-SNE plot of real and generated samples after 5000 epochs of training

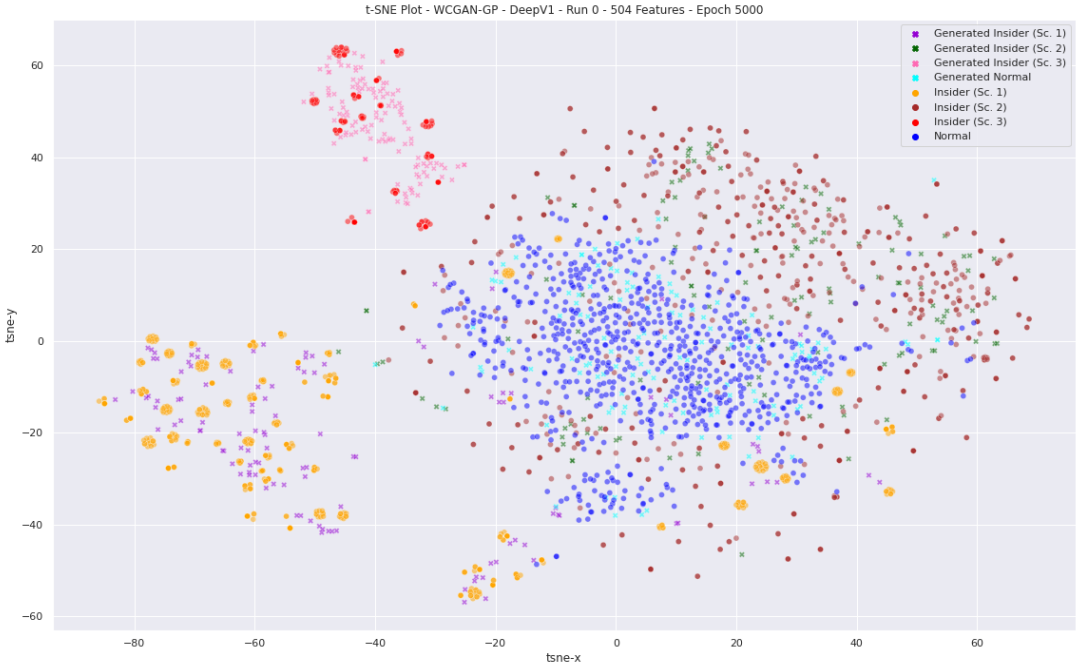
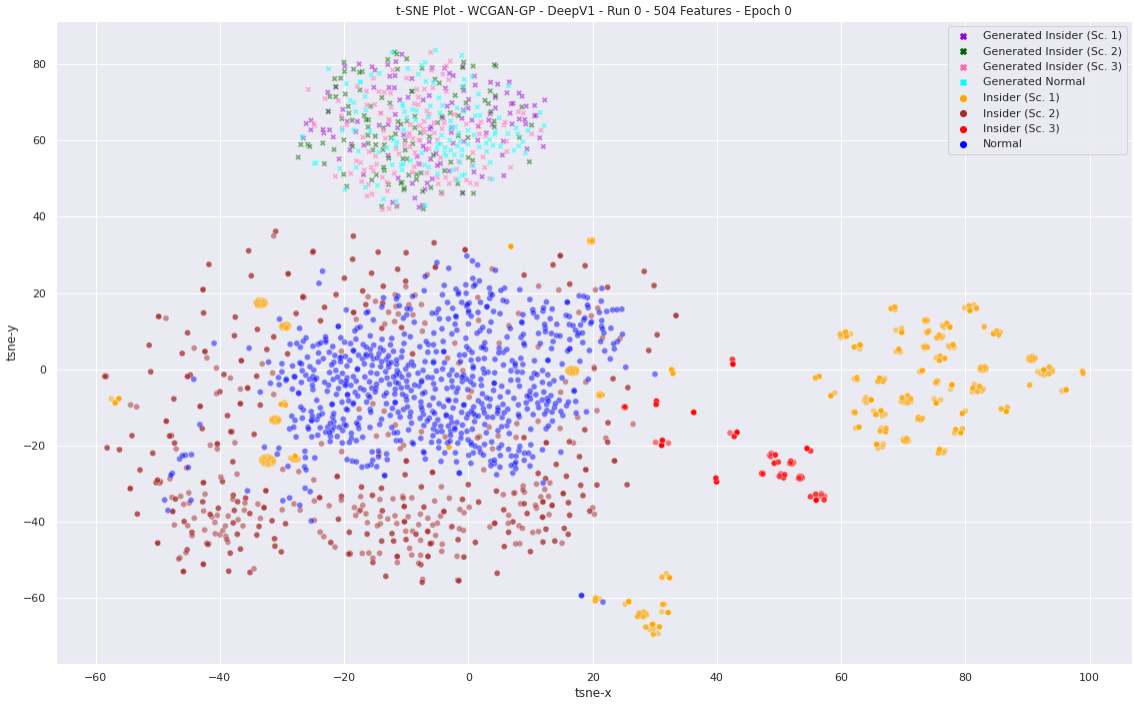
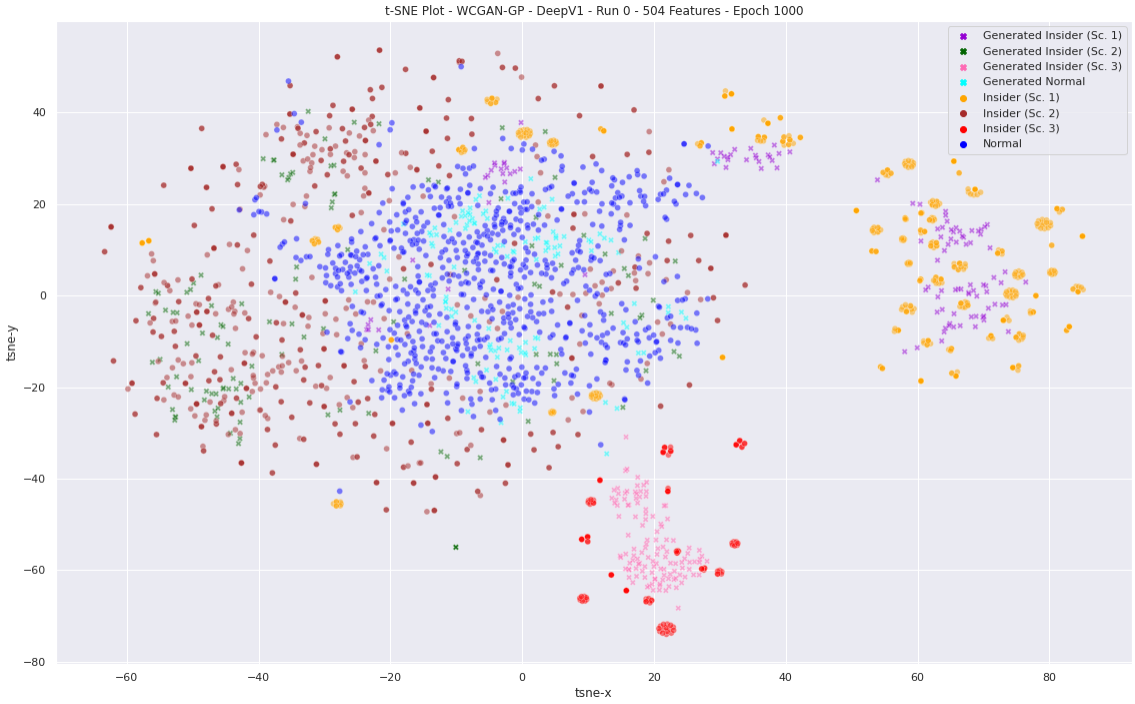


Figure 4.9: WCGAN-GP DeepV1, 504 features: t-SNE plot of real and generated samples after 5000 epochs of training

Figures 4.10, 4.11 and 4.12 show the training progress every 1000 epochs for the 504 feature, DeepV1 GAN. At epoch zero, the network weights had just been initialized, where the generated samples for each class are all randomly clustered together in the same latent space. By the end of epoch 1000, the generated classes were already closely aligned with their real-world pairings. Throughout the next 4000 epochs, the generated samples slowly filled the latent space to more closely resemble the real distributions. Interestingly, Figures 4.11 and 4.12 seems to show no significant changes in generated sample distributions from epoch 3000 to 5000. It is left to future work to investigate an effective mechanism for stopping GAN training early if no performance benefits can be gained. Future works should also evaluate the likelihood and impact of overfitting.

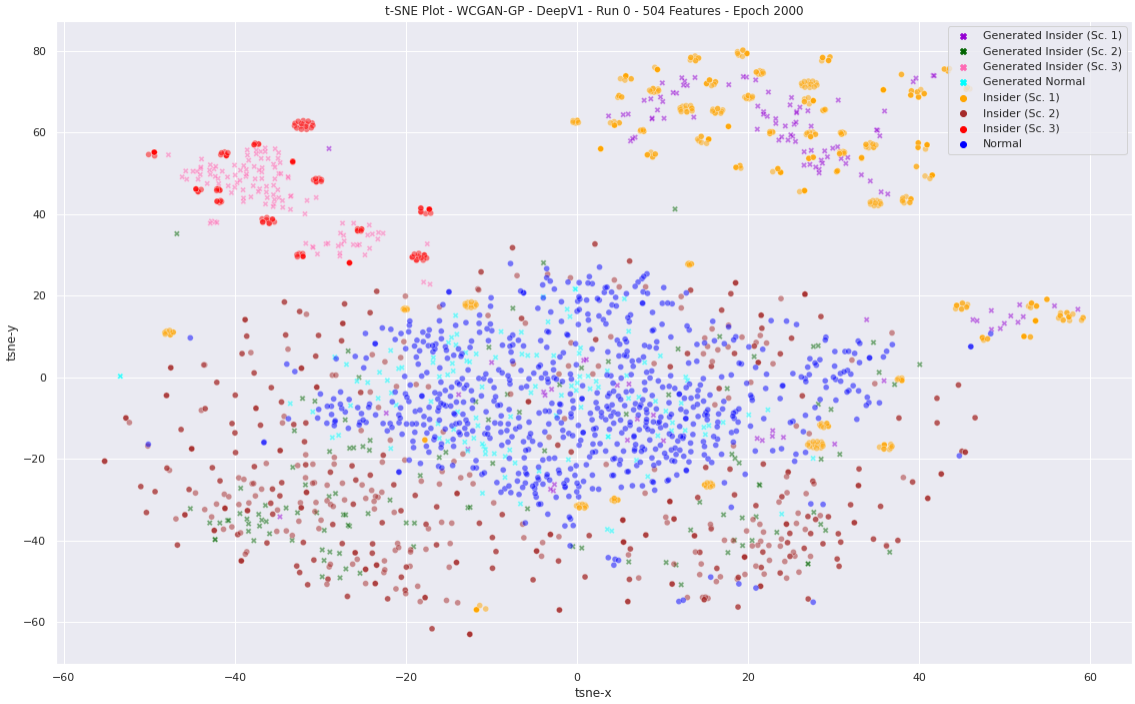


(a) Epoch 0

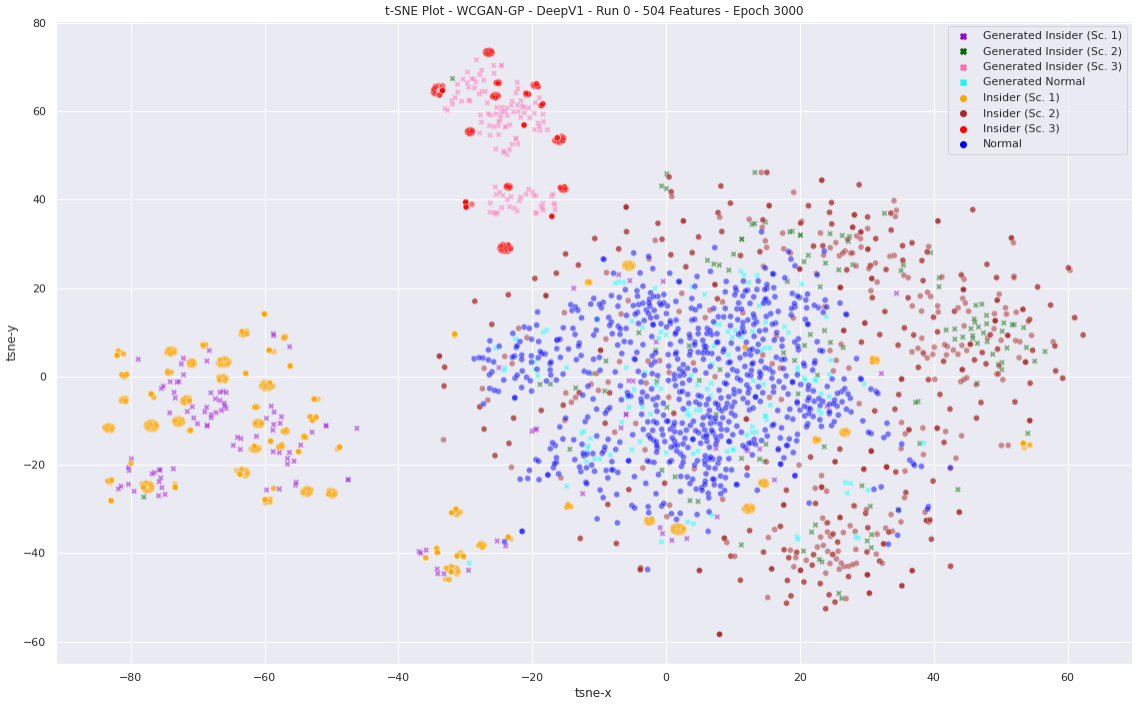


(b) Epoch 1000

Figure 4.10: WCGAN-GP DeepV1, 504 features, Epochs 0-1000: t-SNE plots of generated samples throughout training

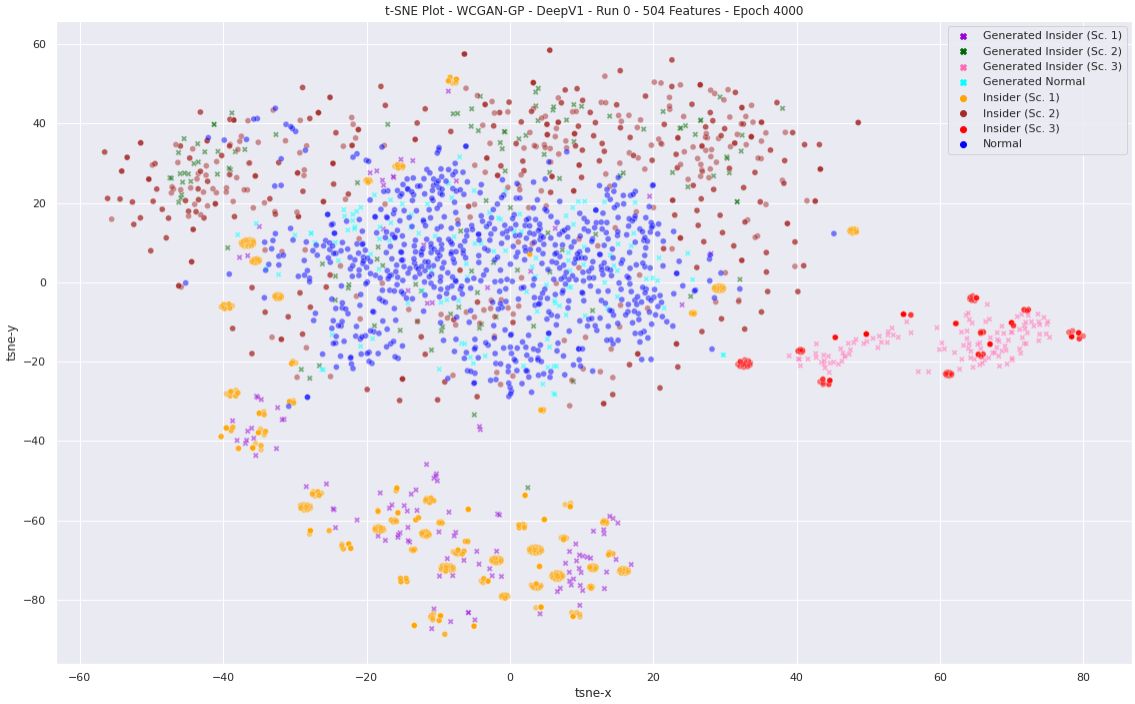


(a) Epoch 2000

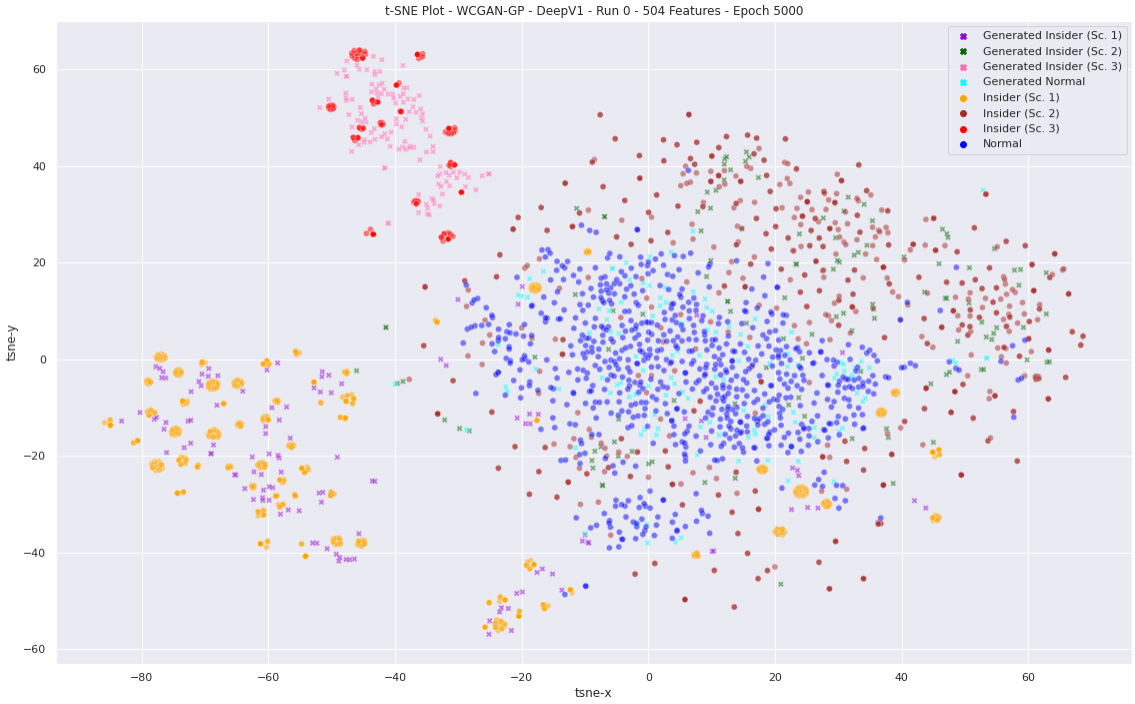


(b) Epoch 3000

Figure 4.11: WCGAN-GP DeepV1, 504 features, Epochs 2000-3000: t-SNE plots of generated samples throughout training



(a) Epoch 4000



(b) Epoch 5000

Figure 4.12: WCGAN-GP DeepV1, 504 features, Epochs 4000-5000: t-SNE plots of generated samples throughout training

Figure 4.13 shows the t-SNE plot of the SMOTE augmented training set against the test set. The implementation of the SMOTE library had a limitation where the original samples were not distinguishable from the generated samples in the SMOTE output. As a result, this graph is not directly comparable to the t-SNE plots of the GAN generated samples. It is instead meant to help visualize how SMOTE adds linearities to the existing samples to create the augmented dataset.

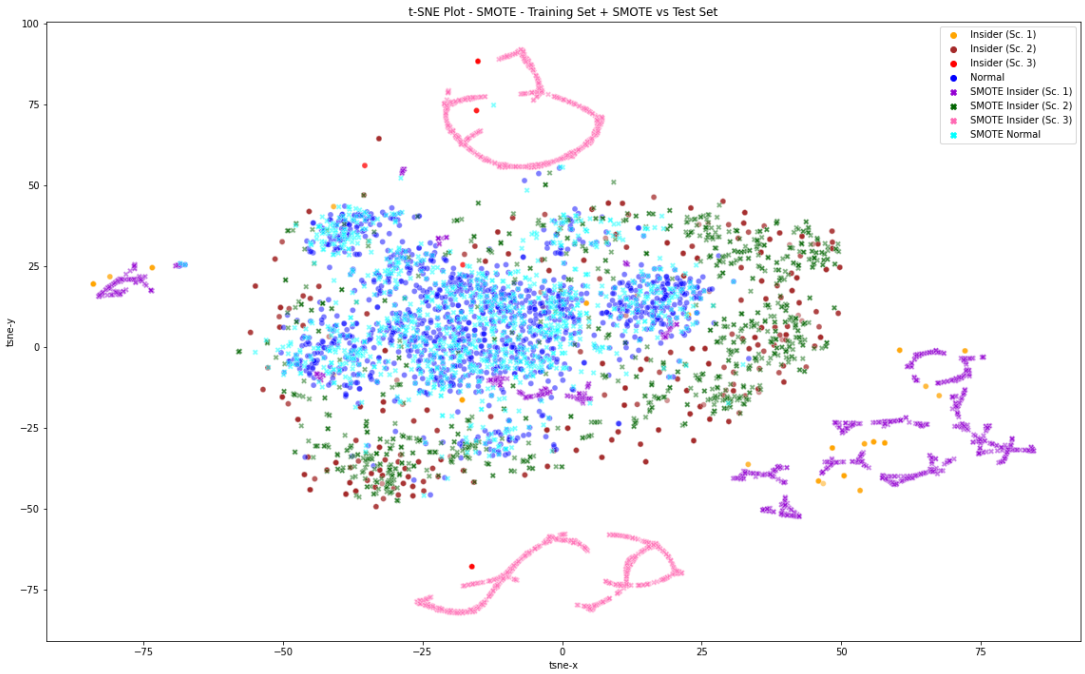
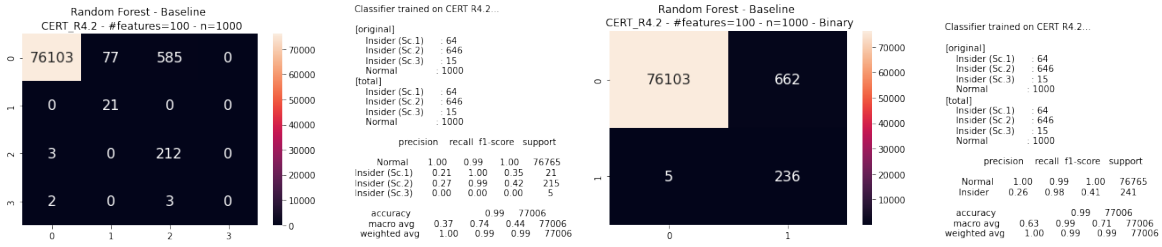


Figure 4.13: t-SNE plot for SMOTE generated samples

4.2 CGAN vs WCGAN-GP

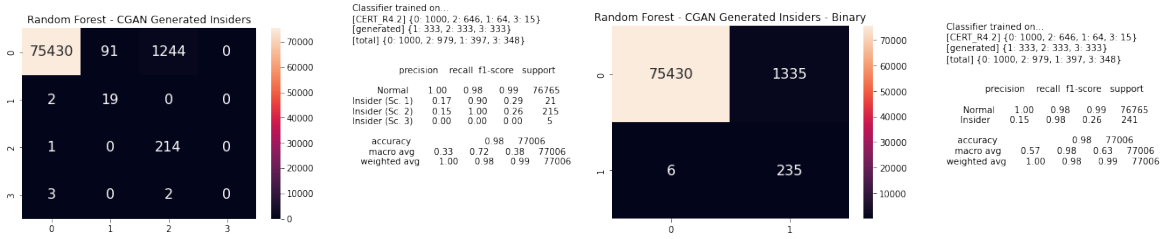
A baseline training dataset was created by taking all insider samples and 1000 randomly selected normal samples. The trained CGAN and WCGAN-GP were each used to generate 333 samples of each insider class. An augmented dataset was created for each of the GANs using the baseline dataset and their respectively generated samples. A Random Forest classifier was then trained on each of the two augmented datasets, as well as the baseline dataset. Sample confusion matrices of the classification results can be seen in Figure 4.14. The matrices on the left show the multi-class matrices with all four classes (Normal, Insider Sc.1-3), while the matrices on the right show the binary classification (Normal vs Insider). Both multi-class and binary confusion matrices refer to the same classifier outputs; the binary matrices were just created to provide a simpler metric. These results were obtained during the earlier stages of the research, so results were only captured for the 100 feature dataset. Notably, these results were captured prior to adding dropout to WCGAN-GP and defining the DeepV1 vs ShallowV1 architectures.

Figure 4.14 shows that the WCGAN-GP strategy resulted in 3 more insiders being detected, but the false-positive rate was doubled from 600 to 1200. The CGAN strategy still performed surprisingly well given it's t-SNE plot, but that could be due to the classifier learning additional modes for each class. Regardless, this resulted in double the false positives and fewer insiders detected (compared to the baseline). Based on these classification results, the training curves and the t-SNE plots, WCGAN-GP was used for the remaining experiments.



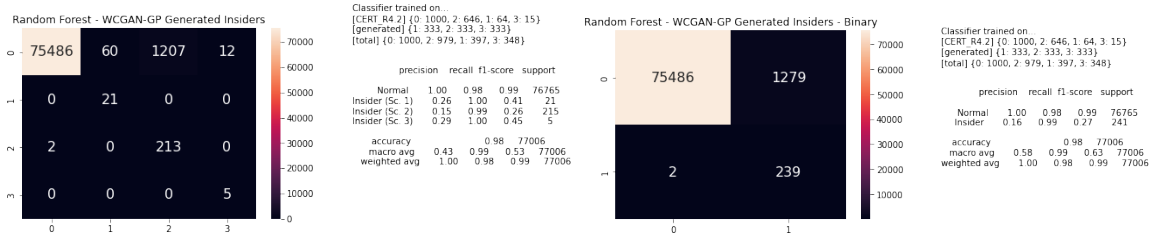
(a) Baseline - Multi-class

(b) Baseline - Binary-class



(c) CGAN - Multi-class

(d) CGAN - Binary-class



(e) WCGAN-GP - Multi-class

(f) WCGAN-GP - Binary-class

Figure 4.14: Random Forest, 100 features: Confusion matrices comparing Baseline, CGAN and WCGAN-GP

4.3 Classifier Selection

In order to fully test the augmentation strategies, multiple classifiers were selected. These included Support Vector Machines (SVM), Random Forests, Logistic Regression and Extreme Gradient Boosting (XGBoost). The parameterizations for these classifiers are provided in Table 4.4. Due to time and computational constraints, the four classifiers were only tested against the baseline, SMOTE and WCGAN-GP DeepV1 augmentation strategies. Further, these classifiers were only tested against the 100 feature dataset and with n_samples values from 2000 to 10000 in intervals of 2000. From the results of these experiments, a top-performing classifier was selected to further explore the additional augmentation strategies, feature counts and n_samples.

Table 4.4: Classifier Parameterization

Classifier	**Parameters**
SVM	kernel = linear
Random Forest	n_estimators = 10
Logistic Regression	C = 0.1
XGBoost	max_depth = 6

The computational cost of training GANs and classifiers is another item to consider. Due to the low number of insider training samples, the GAN training was considerably cheaper than other deep-learning based approaches. However, training classifiers on hundreds of thousands of synthetic samples has a computational cost that must be justified. The Random Forest classifier had a low computational cost, even at high numbers of generated samples. However, some classifiers (such as SVM) had considerably higher computational costs and took much longer to run. This was another factor in selecting Random Forest as the top classifier to carry out the remaining experiments. It is likely that the same results could be achieved with a lower number of samples if more sophisticated undersampling techniques were applied to the majority normal class. It is left to future works to investigate this hypothesis and to reduce the computational cost of the proposed framework.

Table 4.5 shows the macro-averaged metrics for every combination of strategy and classifier across all classes for each dataset. The metrics were averaged across all 10

Table 4.5: 100 features, n_samples=10000: Macro-averaged data augmentation strategy classification metrics for all classifiers

strategy	classifier	CERT R4.2			CERT R5.2			CERT R6.2		
		F1	P	R	F1	P	R	F1	P	R
Baseline	Logistic Regression	0.672	0.767	0.624	0.512	0.632	0.452	0.447	0.484	0.470
	Random Forest	**0.803**	**0.807**	**0.857**	0.515	0.671	0.634	0.546	0.578	0.647
	SVM	0.678	0.775	0.703	**0.527**	**0.718**	0.545	**0.561**	**0.603**	0.576
	XGBoost	0.333	0.498	0.807	0.416	0.585	**0.805**	0.482	0.585	**0.865**
SMOTE	Logistic Regression	0.252	0.263	0.798	0.221	0.404	0.618	0.473	0.483	0.742
	Random Forest	**0.624**	**0.537**	**0.846**	0.413	0.517	0.587	0.418	0.419	0.582
	SVM	0.332	0.302	0.750	0.259	0.419	0.553	0.362	0.367	0.590
	XGBoost	0.425	0.425	0.750	**0.540**	**0.540**	**0.800**	**0.617**	**0.617**	**0.833**
WCGAN-GP - DeepV1	Logistic Regression	0.298	0.278	0.711	0.279	0.408	0.578	**0.479**	**0.483**	0.704
	Random Forest	**0.666**	**0.576**	**0.878**	**0.406**	**0.535**	0.665	0.411	0.389	0.695
	SVM	0.382	0.336	0.716	0.300	0.430	0.563	0.449	0.450	0.667
	XGBoost	0.014	0.007	0.750	0.205	0.223	**0.800**	0.333	0.333	**0.833**

experimental runs and the macro-averages were calculated across all classes for each dataset. For the R4.2 dataset, RF consistently outperformed the other classifiers on F1-score, recall and precision for all augmentation strategies. Although the baseline outperformed the GAN in F1-score and precision, GAN had a higher recall. The class-wise classification metrics for datasets R4.2, R5.2 and R6.2 can be found in the Appendix Tables A.1, A.2 and A.3 respectively.

4.4 Augmentation Strategy Comparison

By limiting the classification experiments to a single classifier, it became possible to expand the n_samples to 20000 sample intervals up to 100000 samples per insider class.

4.4.1 100 Feature Set

Figures 4.15, 4.16 and 4.17 show the macro-averaged box and line plots for the 100 feature R4.2, R5.2 and R6.2 datasets, respectively. The line plots in the left column show macro-averaged precision, recall and F1-score as n_samples is increased to 100000. The box plots in the right column show the mean and variance of the same metrics when n_samples reaches 100000 samples. They show that as n_samples is increased, the precision and F1-score generally trend up for all augmentation strategies. However, recall drops sharply at first, then levels out as n_samples approaches

100000. Table 4.6 shows the macro-averaged metrics for each augmentation strategy and dataset, with the highest performing score for each metric column highlighted in bold. SMOTE had the poorest performance on all macro-averaged metrics for all datasets. The baseline strategy was the top performer for F1-score and precision for all datasets. The shallow GAN architecture had the highest recall for the R4.2 and R5.2 datasets, but the baseline claimed the highest recall for R6.2 by a narrow margin. The two GAN architectures had similar performance, with the shallow GAN scoring higher for R4.2 and R5.2, and the deep GAN scoring higher for R6.2.

Table 4.6: 100 features, n_samples=100000, Random Forest: Macro-averaged data augmentation strategy classification metrics

Dataset	CERT R4.2			CERT R5.2			CERT R6.2		
Measure	F1	P	R	F1	P	R	F1	P	R
Baseline	**0.860**	**0.951**	0.806	**0.604**	**0.801**	0.615	**0.653**	**0.690**	**0.673**
SMOTE	0.752	0.792	0.744	0.504	0.581	0.503	0.415	0.445	0.464
WCGAN-GP - DeepV1	0.817	0.860	0.789	0.542	0.645	0.604	0.522	0.493	0.662
WCGAN-GP - ShallowV1	0.848	0.892	**0.816**	0.562	0.700	**0.629**	0.487	0.468	0.667

As expected, the R4.2 test partition had the highest detection scores, with the baseline strategy reporting an F1-score of 0.860. Figure 4.15 shows that the shallow GAN strategy is competitive with the baseline as n_samples increases. Figure 4.18 shows weight-averaged binary classification plots for the R4.2 dataset that are consistent with the macro-averaged plots. Figures 4.20 and 4.19 show class-wise metrics with individual plots for each insider class. This plot reveals that SMOTE has a higher weight-averaged recall than the other strategies because it is more likely to detect the majority insider class (Sc. 2). However, this comes with the trade-off of SMOTE miss-classifying many normal samples as insiders, resulting in a much lower precision and F1-score. This makes sense when looking back at the t-SNE plots for normal vs scenario 2 classes, the two most overlapping distributions. The weight-averaged detection metrics for each class are given in Table 4.7. Confusion matrices from the first experimental run are presented in Figure 4.21. These sample matrices show the same patterns observed in the earlier graphs and tables. SMOTE detects the most insiders, but has double the false positives of the other strategies. The shallow GAN detects three more insiders than the baseline, but has 10 more false positives.

This appears to be a particularly bad run for detecting the minority insider class, where only the shallow GAN is able to catch a few scenario 3 insiders. This is still consistent with the other averaged results because scenario 3 has only 20 samples, resulting in a high detection variance as seen in Figures 4.19 and 4.20.

Table 4.7: CERT R4.2 - 100 features - Insider Threat Detection Data Augmentation Strategy Classification Results

Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)		
Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Baseline	1.000	1.000	1.000	0.864	0.983	0.776	0.894	0.921	0.870	0.682	0.900	0.580
SMOTE	0.999	1.000	0.999	0.781	0.831	0.752	0.842	0.788	0.904	0.387	0.550	0.320
WCGAN-GP - DeepV1	1.000	1.000	1.000	0.838	0.880	0.810	0.884	0.906	0.866	0.545	0.655	0.480
WCGAN-GP - ShallowV1	1.000	1.000	1.000	0.863	0.933	0.810	0.889	0.904	0.874	0.640	0.730	0.580

Table 4.8: CERT R5.2 - 100 features - Insider Threat Detection Data Augmentation Strategy Classification Results

Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)			Insider (Sc.4)		
Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Baseline	0.998	0.999	0.998	0.837	0.953	0.747	0.481	0.359	0.772	0.691	0.994	0.555	0.011	0.700	0.006
SMOTE	0.999	0.999	0.999	0.822	0.936	0.733	0.555	0.468	0.684	0.143	0.304	0.100	0.001	0.200	0.001
WCGAN-GP - DeepV1	0.998	0.999	0.997	0.728	0.703	0.775	0.478	0.370	0.775	0.498	0.553	0.470	0.010	0.600	0.005
WCGAN-GP - ShallowV1	0.998	0.999	0.998	0.801	0.834	0.784	0.502	0.387	0.783	0.491	0.478	0.575	0.015	0.800	0.008

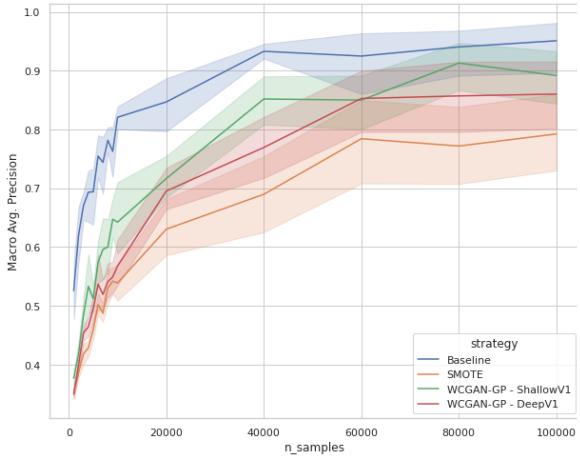
Table 4.9: CERT R6.2 - 100 features - Insider Threat Detection Data Augmentation Strategy Classification Results

Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)			Insider (Sc.4)			Insider (Sc.5)		
Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Baseline	1.000	1.000	1.000	0.800	1.000	0.667	0.352	0.242	0.670	0.767	0.900	0.700	0.000	0.000	0.000	1.000	1.000	1.000
SMOTE	1.000	1.000	1.000	0.800	1.000	0.667	0.124	0.070	0.565	0.167	0.200	0.150	0.000	0.000	0.000	0.400	0.400	0.400
WCGAN-GP - DeepV1	1.000	1.000	1.000	0.587	0.540	0.667	0.261	0.161	0.755	0.285	0.256	0.550	0.000	0.000	0.000	1.000	1.000	1.000
WCGAN-GP - ShallowV1	1.000	1.000	1.000	0.587	0.605	0.667	0.180	0.105	0.735	0.156	0.098	0.600	0.000	0.000	0.000	1.000	1.000	1.000

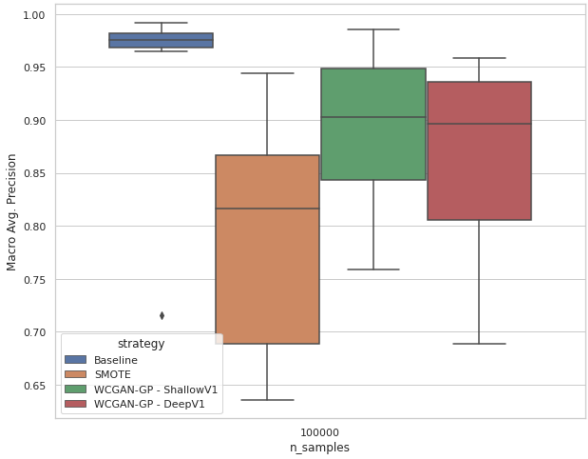
Despite the classifiers and augmentation strategies only being trained on R4.2, they demonstrated a modest ability to generalize detection of insiders in the R5.2 and R6.2 datasets. Baseline strategy macro-averaged F1-scores of 0.604 and 0.653 were reported for the R5.2 and R6.2 datasets, respectively (down from 0.860 in R4.2). From the macro-averaged plots in Figure 4.16, it can be seen that the broad patterns from R4.2 are maintained. The shallow GAN still has a marginal advantage over the baseline in recall, but the baseline widened its lead on precision and therefore F1-score. The individual R5.2 detection metrics for each class are presented in Table 4.8. R5.2 introduces a new class (Insider Sc. 4) to the classification problem, with a

support of 339 real samples in the test dataset. All strategies resulted in poor scenario 4 detection, with the highest F1-score (0.015) being reported by the shallow GAN.

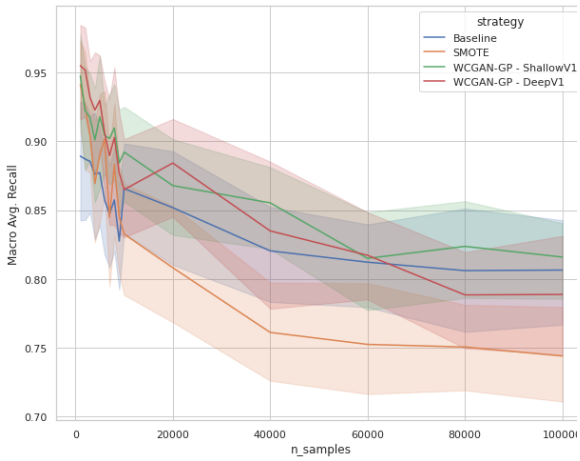
In R6.2, an additional insider class (Sc. 5) is introduced and the number of insider samples is drastically reduced. As presented in Table 3.3 earlier in this thesis, R6.2 only contains [3, 20, 2, 1, 1] samples for scenarios 1-5. Table 4.9 shows that similarly to the R5.2 results, no strategies were able to detect the single scenario 4 sample. Inversely, all but the SMOTE strategy got perfect F1-scores for the new scenario 5 sample. This indicates that some of the insider scenarios have behaviour that can be generalized using the proposed approach, but other scenarios cannot. Future works could investigate why this is the case, starting with creating t-SNE plots of the newly introduced insider classes. The remaining R5.2 and R6.2 figures can be found in the appendix. The confusion matrix samples for R5.2 and R6.2 can be seen in Figures A.2 and A.3. Figures A.6, A.7, and A.8 show the R5.2 binary class, multi-class line and box plots, respectively. Their R6.2 counterparts can be found in tables A.9, A.10 and A.11.



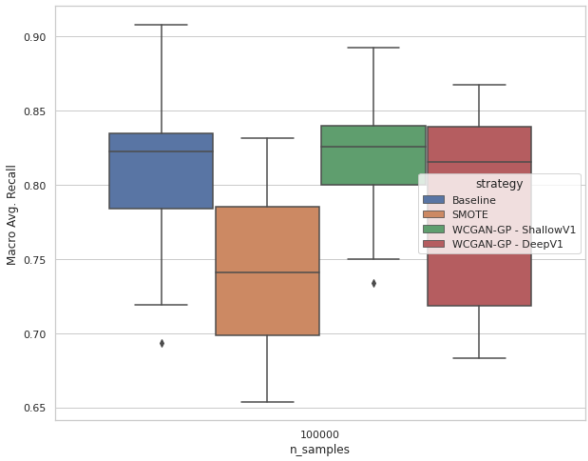
(a) Precision (Macro Avg.)



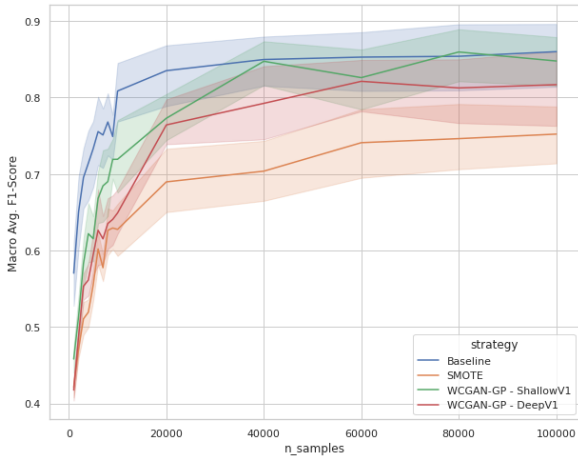
(b) Precision (Macro Avg.)



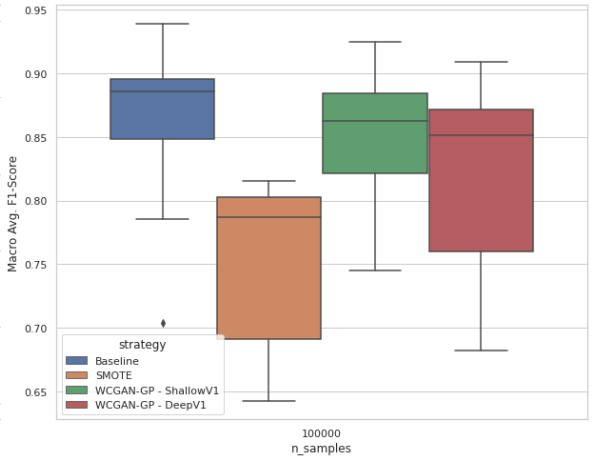
(c) Recall (Macro Avg.)



(d) Recall (Macro Avg.)

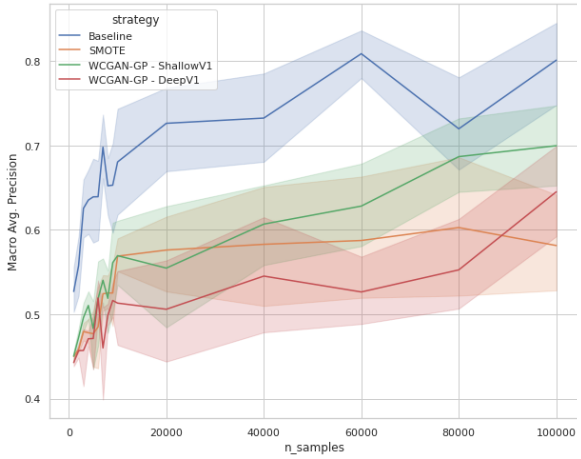


(e) F1-Score (Macro Avg.)

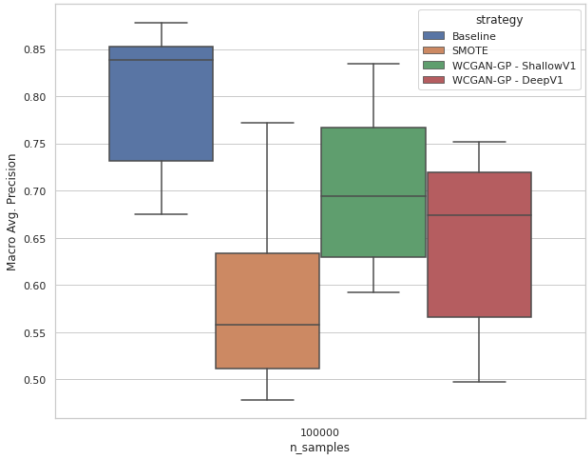


(f) F1-Score (Macro Avg.)

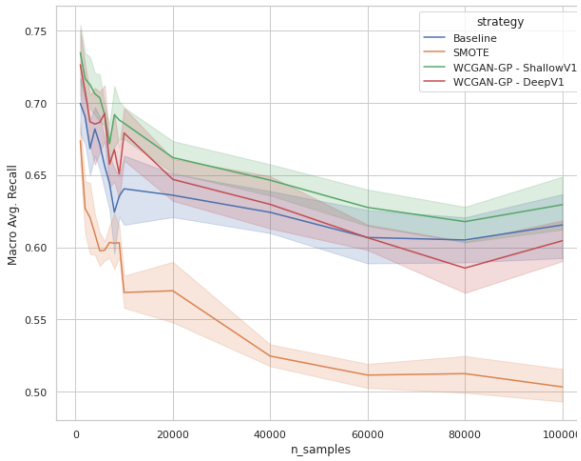
Figure 4.15: CERT R4.2, 100 features: Macro-averaged measures from classification results



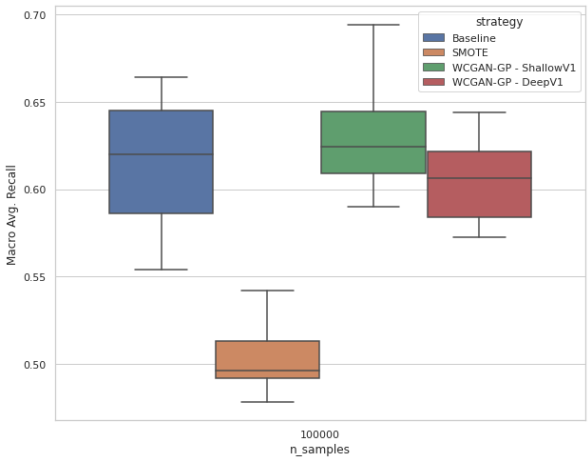
(a) Precision (Macro Avg.)



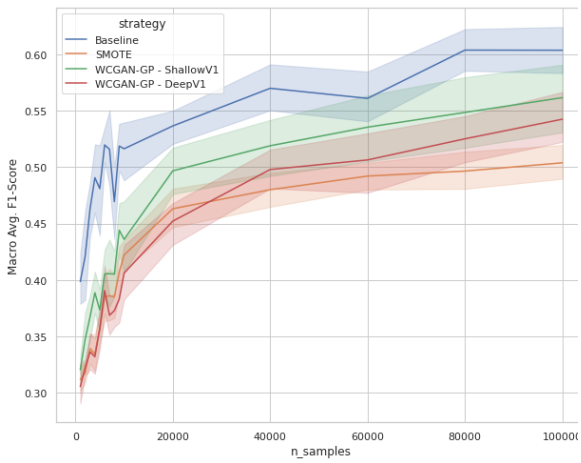
(b) Precision (Macro Avg.)



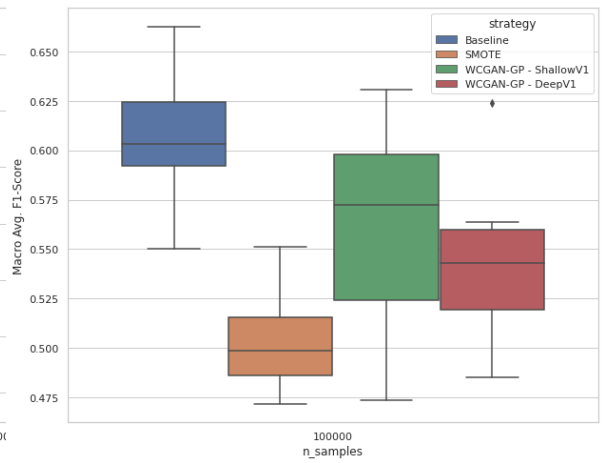
(c) Recall (Macro Avg.)



(d) Recall (Macro Avg.)

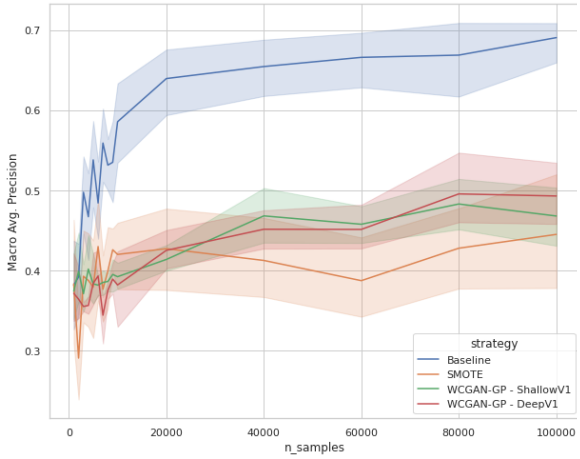


(e) F1-Score (Macro Avg.)

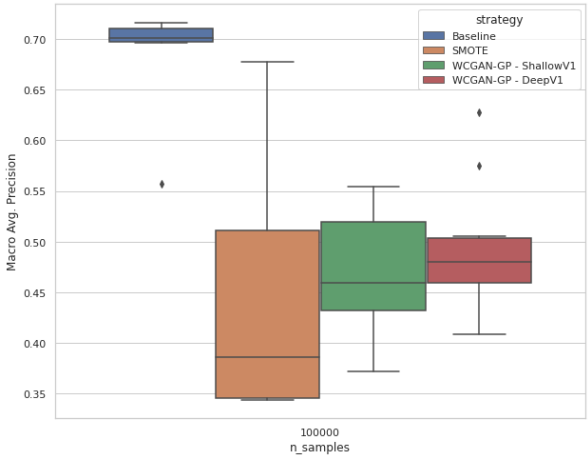


(f) F1-Score (Macro Avg.)

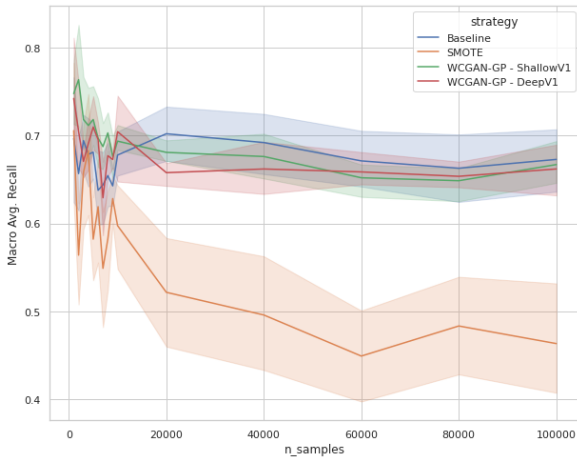
Figure 4.16: CERT R5.2, 100 features: Macro-averaged measures from classification results



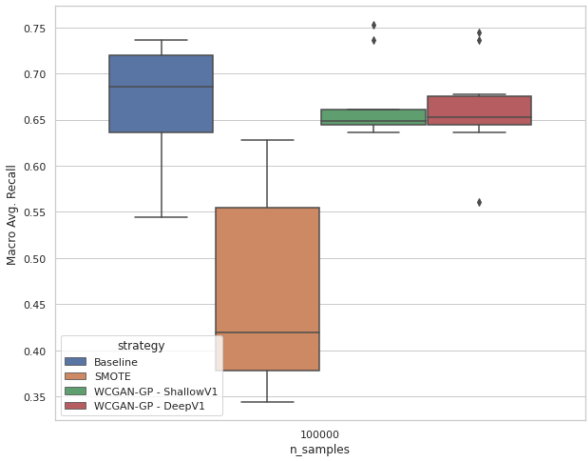
(a) Precision (Macro Avg.)



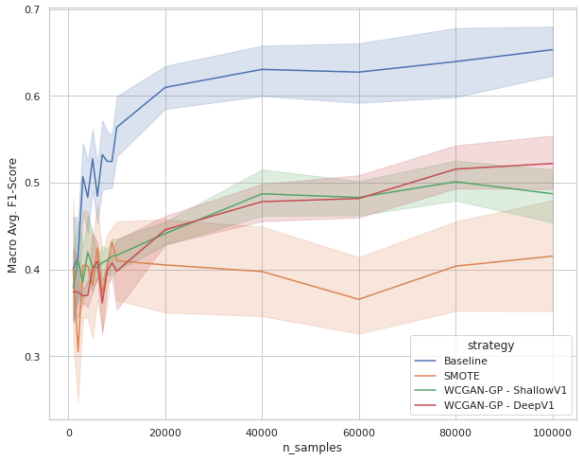
(b) Precision (Macro Avg.)



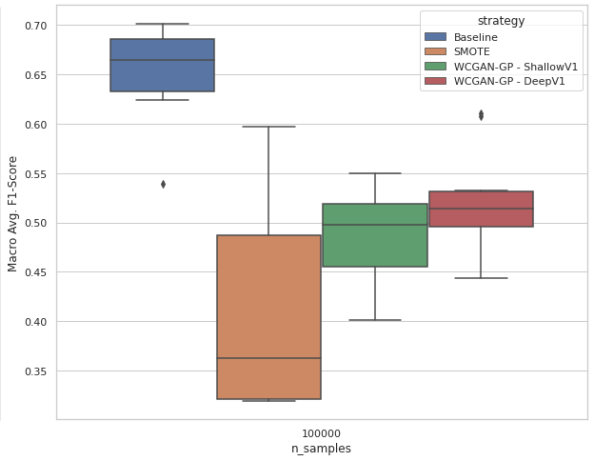
(c) Recall (Macro Avg.)



(d) Recall (Macro Avg.)



(e) F1-Score (Macro Avg.)



(f) F1-Score (Macro Avg.)

Figure 4.17: CERT R6.2, 100 features: Macro-averaged measures from classification results

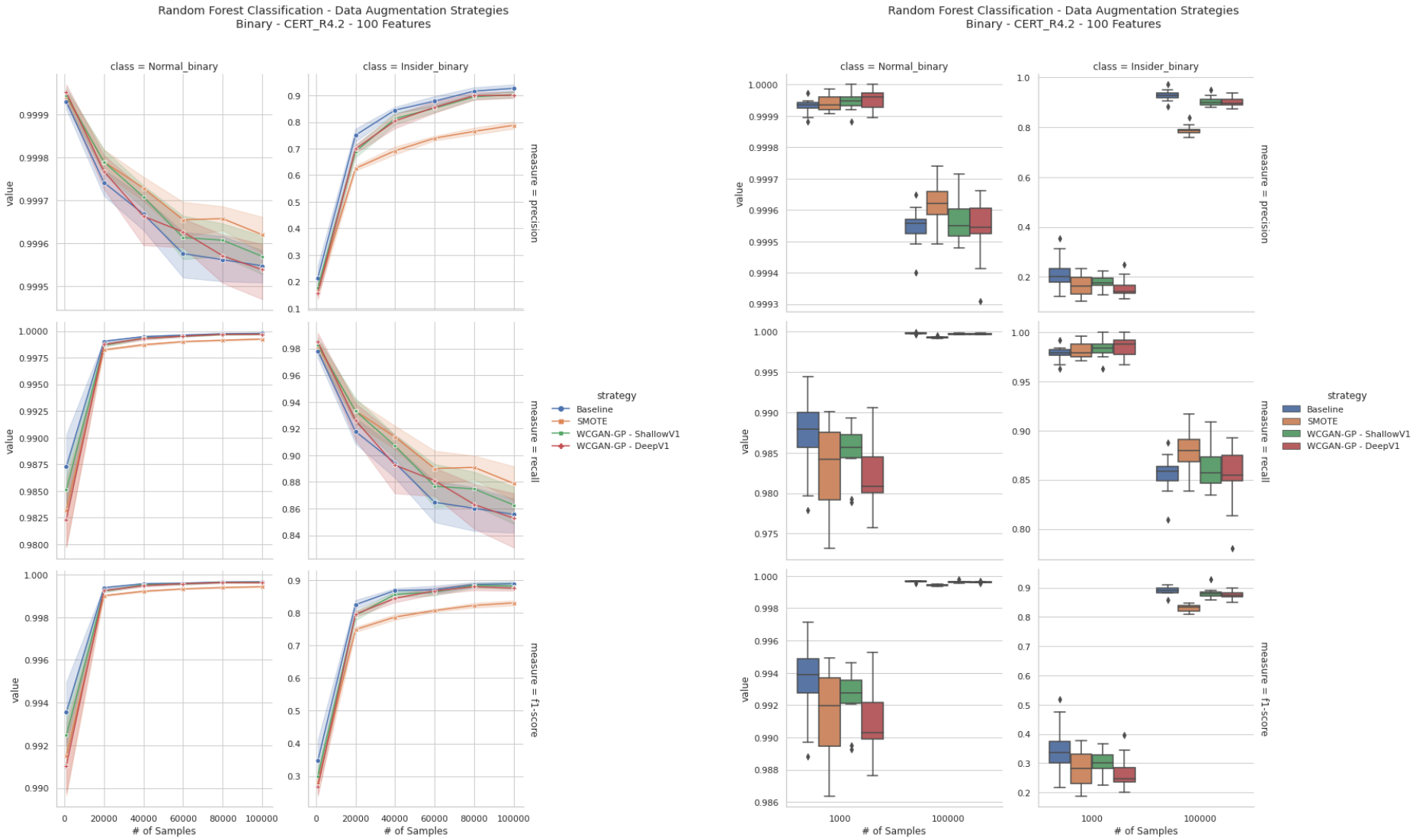


Figure 4.18: CERT R4.2, 100 features: Binary classification plots for all augmentation strategies

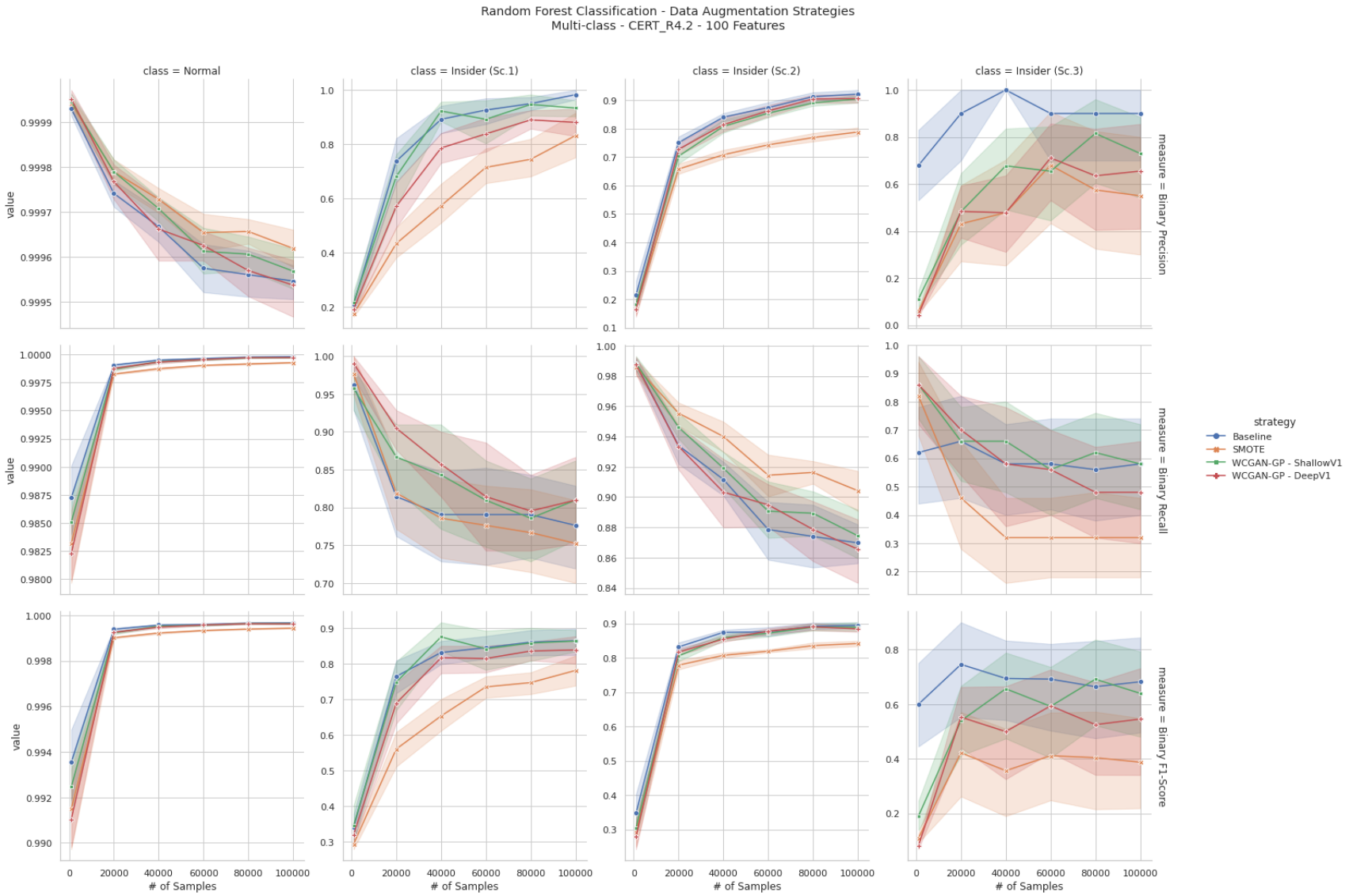


Figure 4.19: CERT R4.2, 100 features: Classification metrics as n_samples increases for all augmentation strategies

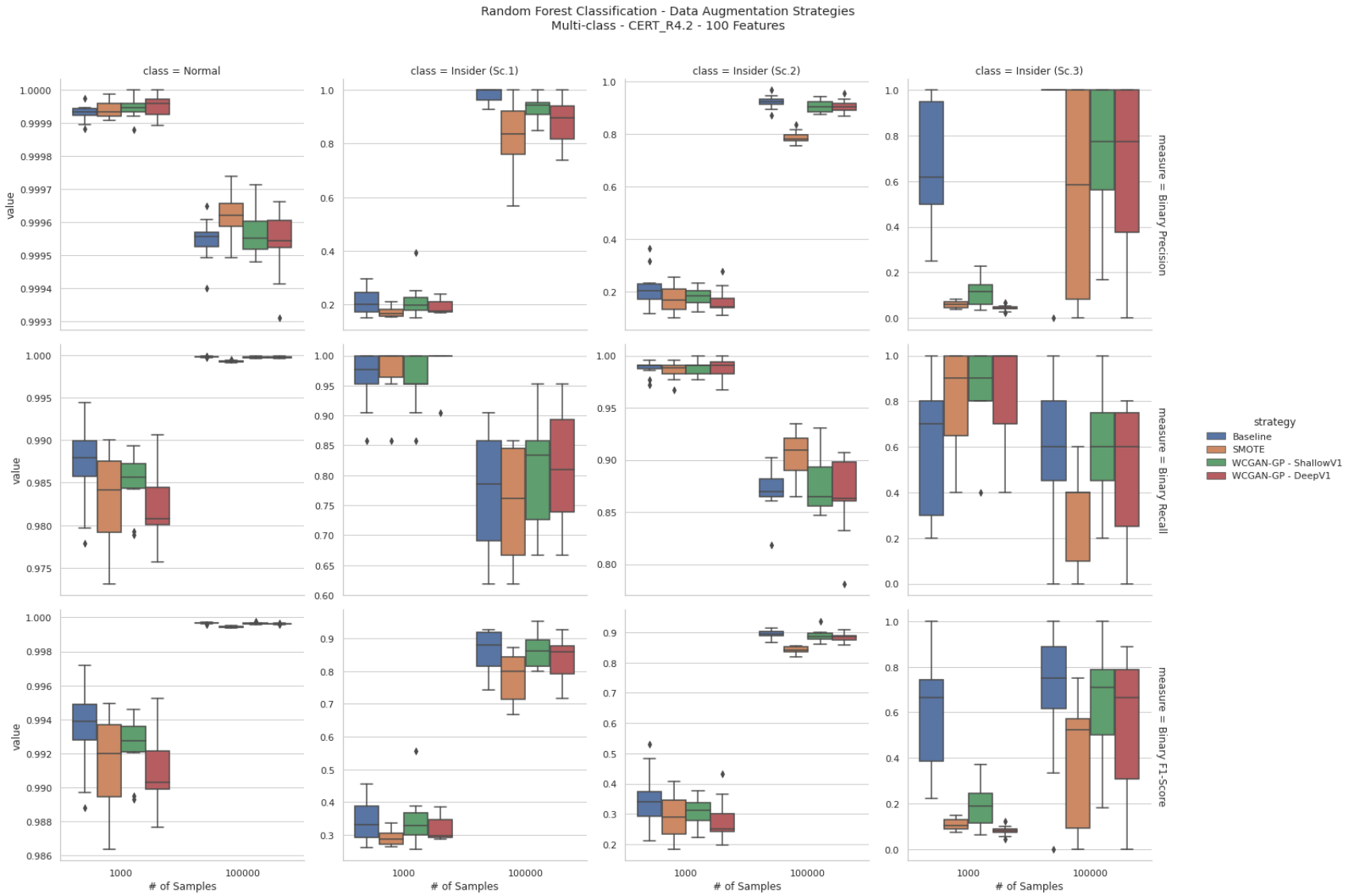
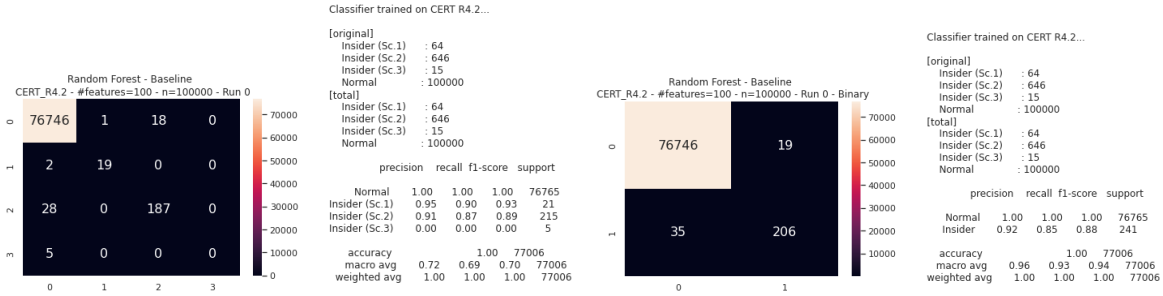
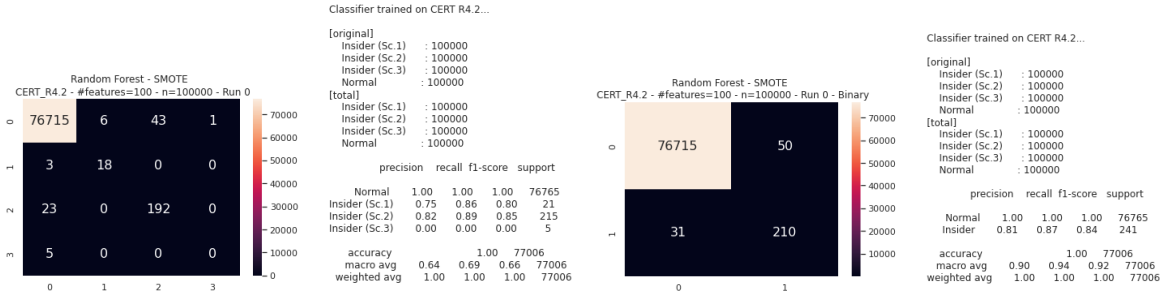


Figure 4.20: CERT R4.2, 100 features: Classification metrics at 1000 and 100000 samples per class for all augmentation strategies



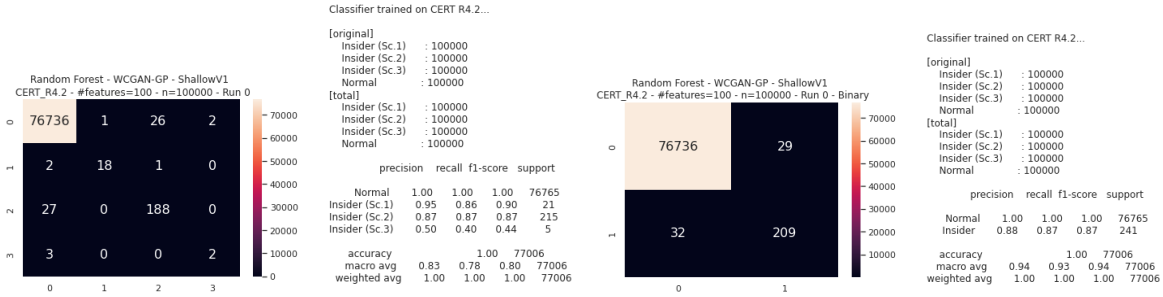
(a) Baseline - Multi-class

(b) Baseline - Binary-class



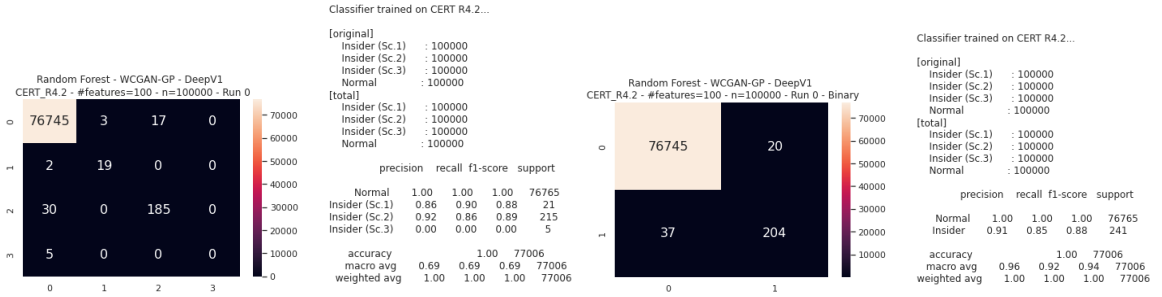
(c) Smote - Multi-class

(d) Smote - Binary-class



(e) WCGAN-GP ShallowV1 - Multi-class

(f) WCGAN-GP ShallowV1 - Binary-class



(g) WCGAN-GP DeepV1 - Multi-class

(h) WCGAN-GP DeepV1 - Binary-class

Figure 4.21: CERT R4.2, 100 features: Sample confusion matrices for Random Forest classification

4.4.2 504 Feature Set

The exact same experiments were repeated using the 504 feature set. Figures 4.22, 4.23 and 4.24 show the macro-averaged box and line plots for the 504 feature R4.2, R5.2 and R6.2 datasets, respectively. They show the same broad trends as their 100 feature counterparts; F1-score and precision rise quickly at first, then gradually as n_samples approaches 100000, while recall drops inversely. However, the figures show that both of the GAN architectures outperform SMOTE and baseline on almost all macro-averaged measures. This is corroborated by the macro-averaged dataset and augmentation strategy metrics in Table 4.10. Deep GAN reports the top macro-averaged F1-score of 0.911 for the R4.2 dataset and 0.614 for the CERT R6.2 dataset. The shallow GAN reports the top macro-averaged F1-score of 0.617 for R5.2. Baseline was consistently competitive on precision, but the GANs consistently outperformed on recall. Figure 4.25 shows the binary classification plots, which show the GANs are a modest improvement from the baseline when using weight-averaged metrics. Class-wise metrics can be found in Table 4.11 as well as Figures 4.26 and 4.27. These plots show that the GANs can maintain the precision of the baseline while identifying additional insiders from the minority classes. Figure 4.28 shows some sample confusion matrices from the same experimental run.

Table 4.10: 504 features, n_samples=100000, Random Forest: Macro-averaged data augmentation strategy classification metrics

Dataset	CERT R4.2			CERT R5.2			CERT R6.2		
Measure	F1	P	R	F1	P	R	F1	P	R
Baseline	0.892	0.973	0.833	0.582	**0.809**	0.596	0.556	0.586	0.575
SMOTE	0.853	0.896	0.834	0.584	0.789	0.518	0.448	0.474	0.443
WCGAN-GP - DeepV1	**0.911**	0.972	**0.865**	0.605	0.800	**0.673**	**0.614**	0.596	**0.738**
WCGAN-GP - ShallowV1	0.910	**0.975**	0.863	**0.617**	**0.809**	0.656	0.606	**0.604**	0.696

Similar to the 100 feature set, the 504 feature set performance dropped considerably between the R4.2 dataset it was trained on, to the R5.2 and R6.2 datasets. Table 4.12 and 4.13 show the augmentation strategy classification metrics broken down by class for the R5.2 and R6.2 datasets, respectively. Insider Sc.4 still proved difficult to detect, with the deep GAN strategy reporting the top F1-score of 0.061 for the R5.2 dataset, and 0.2 for R6.2. The other strategies failed to detect the single Sc.4 sample

Table 4.11: CERT R4.2 - 504 features - Insider Threat Detection Data Augmentation Strategy Classification Results

Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)		
Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Baseline	0.999	0.999	1.000	0.853	0.994	0.752	0.813	0.936	0.721	0.901	0.963	0.860
SMOTE	0.999	0.999	0.999	0.770	0.785	0.767	0.774	0.801	0.750	0.868	1.000	0.820
WCGAN-GP - DeepV1	1.000	0.999	1.000	0.851	0.966	0.767	0.835	0.941	0.752	0.958	0.983	0.940
WCGAN-GP - ShallowV1	1.000	0.999	1.000	0.852	0.984	0.757	0.834	0.934	0.754	0.955	0.983	0.940

Table 4.12: CERT R5.2 - 504 features - Insider Threat Detection Data Augmentation Strategy Classification Results

Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)			Insider (Sc.4)		
Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Baseline	0.998	0.999	0.996	0.847	0.995	0.738	0.314	0.228	0.601	0.722	0.923	0.630	0.029	0.900	0.015
SMOTE	0.999	0.998	1.000	0.825	0.958	0.725	0.365	0.703	0.249	0.730	0.984	0.615	0.002	0.300	0.001
WCGAN-GP - DeepV1	0.996	0.999	0.993	0.810	0.910	0.736	0.213	0.130	0.669	0.947	0.961	0.935	0.061	1.000	0.032
WCGAN-GP - ShallowV1	0.997	0.999	0.995	0.835	0.947	0.748	0.322	0.227	0.674	0.905	0.974	0.850	0.027	0.900	0.014

Table 4.13: CERT R6.2 - 504 features - Insider Threat Detection Data Augmentation Strategy Classification Results

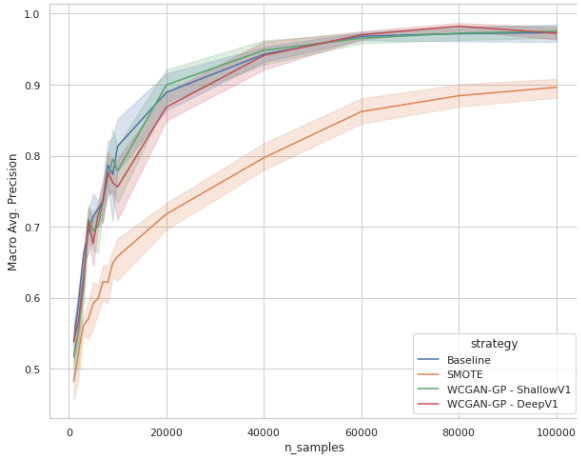
Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)			Insider (Sc.4)			Insider (Sc.5)		
Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Baseline	1.000	1.000	1.000	0.773	0.933	0.667	0.260	0.186	0.535	0.800	0.900	0.750	0.000	0.000	0.000	0.500	0.500	0.500
SMOTE	1.000	1.000	1.000	0.800	1.000	0.667	0.019	0.014	0.040	0.667	0.633	0.750	0.000	0.000	0.000	0.200	0.200	0.200
WCGAN-GP - DeepV1	1.000	1.000	0.999	0.556	0.566	0.667	0.078	0.046	0.560	0.853	0.767	1.000	0.200	0.200	0.200	1.000	1.000	1.000
WCGAN-GP - ShallowV1	1.000	1.000	1.000	0.735	0.885	0.667	0.167	0.106	0.610	0.833	0.733	1.000	0.000	0.000	0.000	0.900	0.900	0.900

in any runs. The non-GAN strategies also had low F1-scores for the Sc.5 sample, while the GAN strategies had near perfect scores.

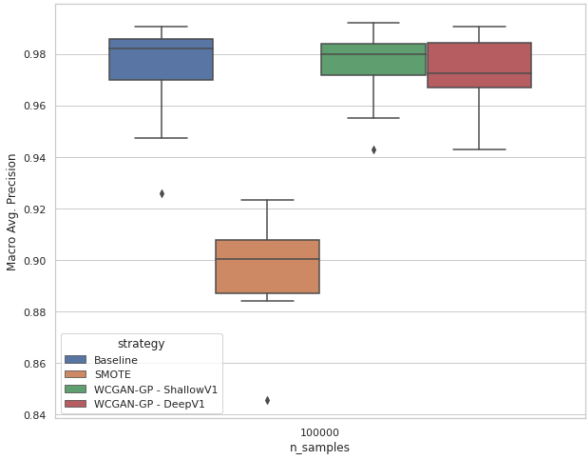
For R5.2, SMOTE actually had the best weight-averaged F1-score due to its high precision in the majority insider class (Sc. 2). This can be seen clearly in Figures A.12 and A.14. For R6.2, it is instead the baseline strategy that benefits from high Sc. 2 precision, with SMOTE reporting very low precision and recall for the same class. This can been seen in Figures A.15 and A.17. It is unclear why the SMOTE performance dropped so significantly between R5.2 and R6.2, but that analysis is left to future works.

Sample confusion matrices for the R5.2 and R6.2 datasets can be found in Figures A.4 and A.5. They show that although the deep GAN has a superior recall and F1-score, the lack of precision can result in a significant raw count of false positives. Compared to the shallow GAN in this example, the deep GAN incurred over 4000

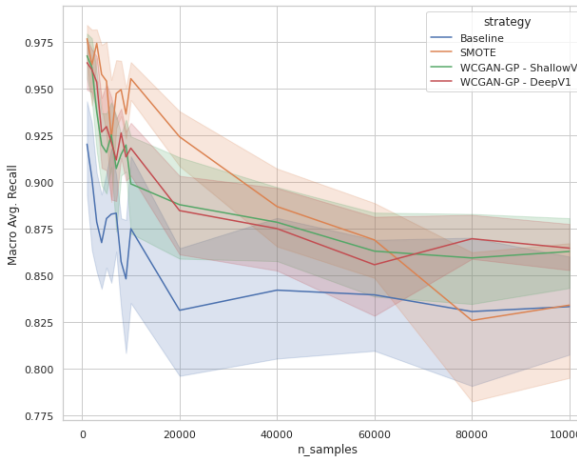
more false positives to detect an additional two insiders.



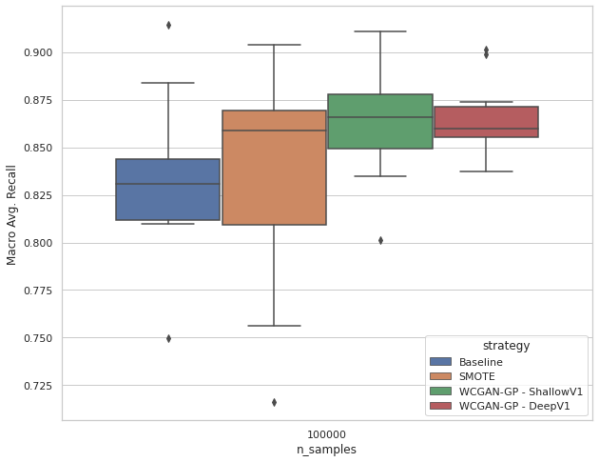
(a) Precision (Macro Avg.)



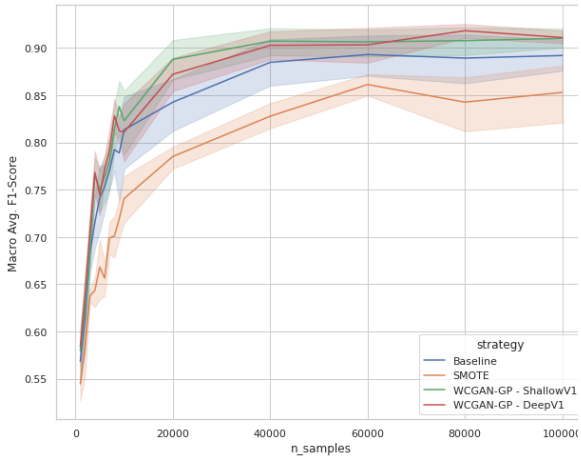
(b) Precision (Macro Avg.)



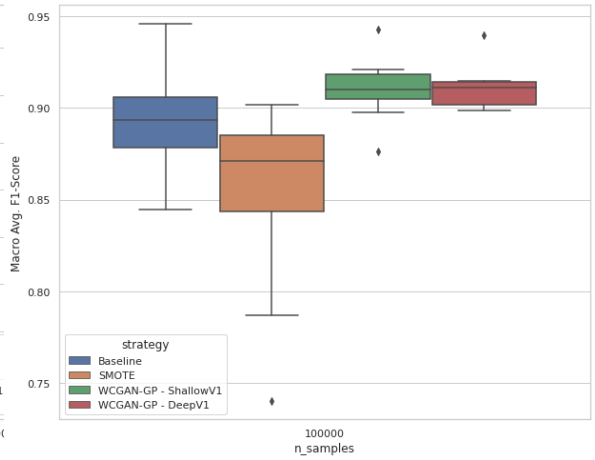
(c) Recall (Macro Avg.)



(d) Recall (Macro Avg.)

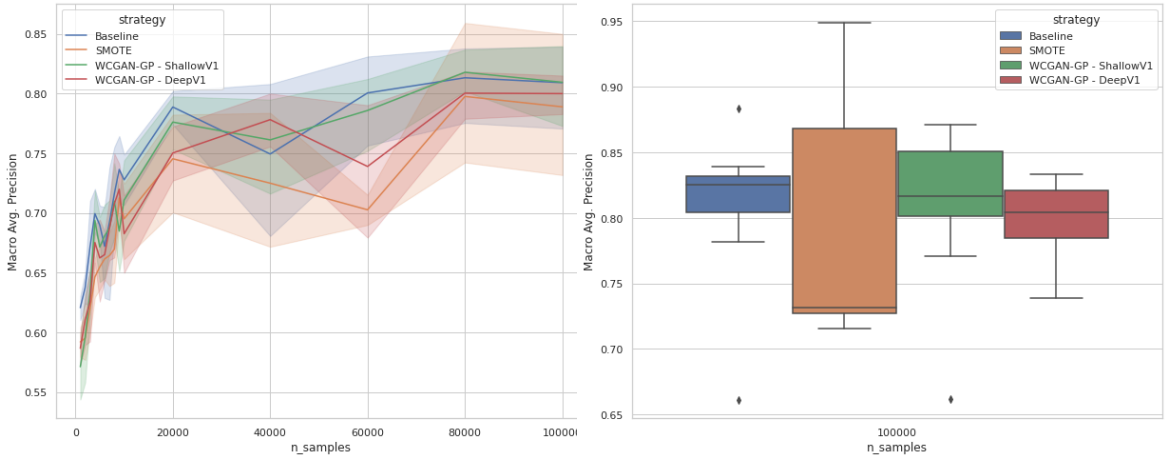


(e) F1-Score (Macro Avg.)



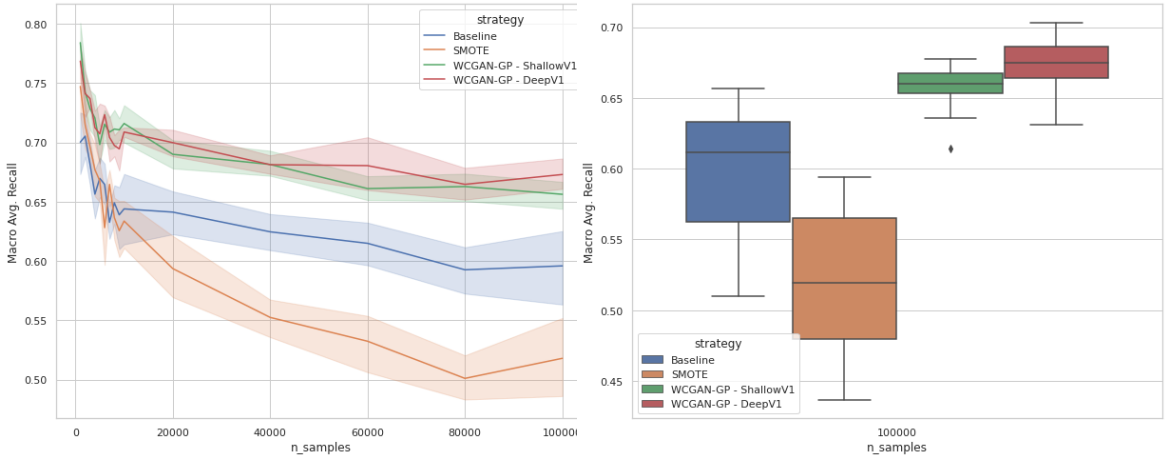
(f) F1-Score (Macro Avg.)

Figure 4.22: CERT R4.2, 504 features: Macro-averaged measures from classification results



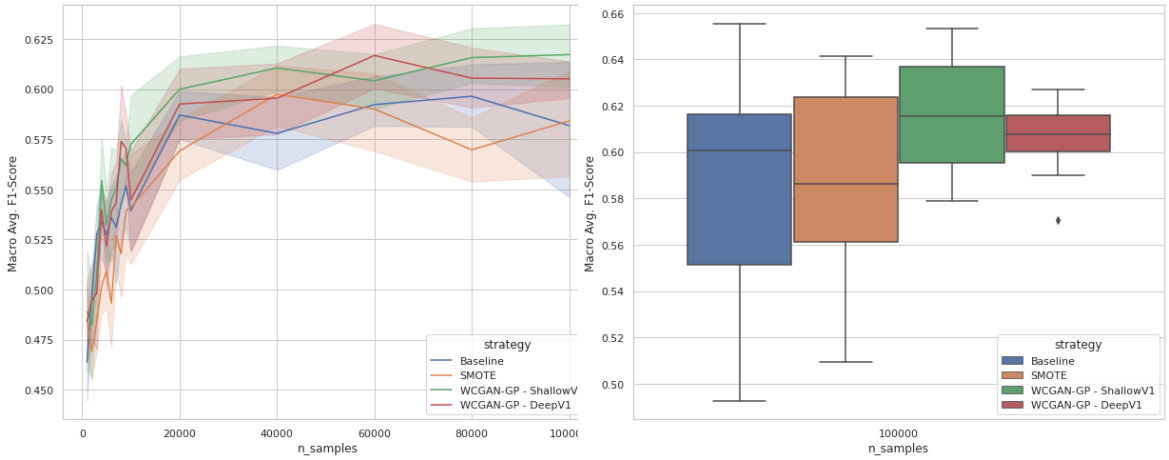
(a) Precision (Macro Avg.)

(b) Precision (Macro Avg.)



(c) Recall (Macro Avg.)

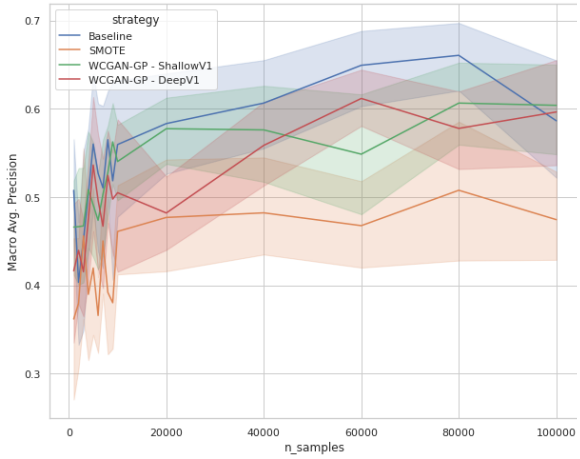
(d) Recall (Macro Avg.)



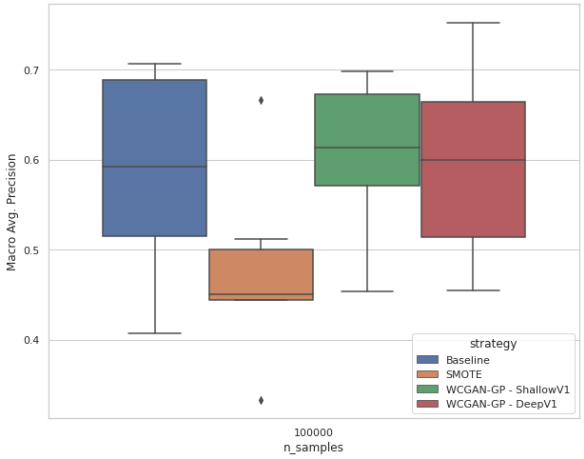
(e) F1-Score (Macro Avg.)

(f) F1-Score (Macro Avg.)

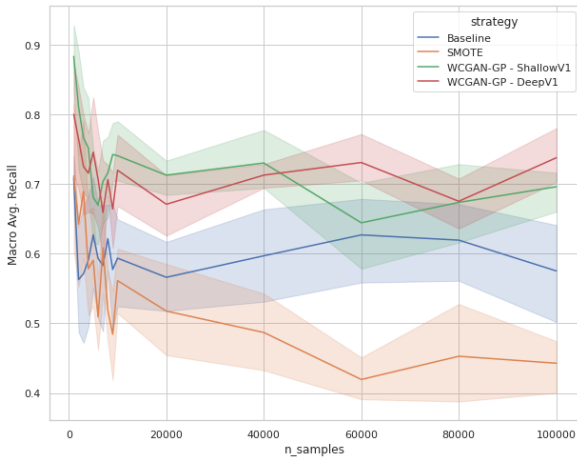
Figure 4.23: CERT R5.2, 504 features: Macro-averaged measures from classification results



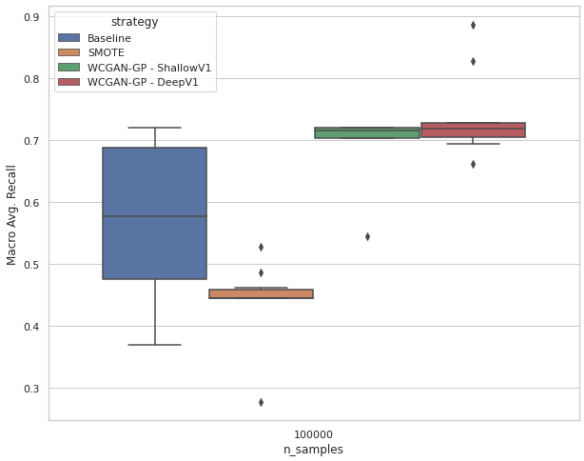
(a) Precision (Macro Avg.)



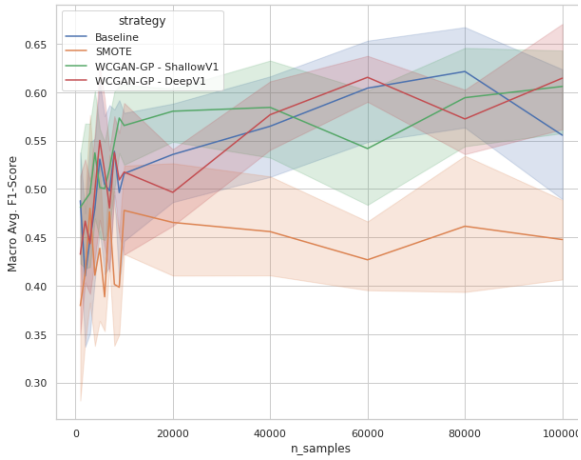
(b) Precision (Macro Avg.)



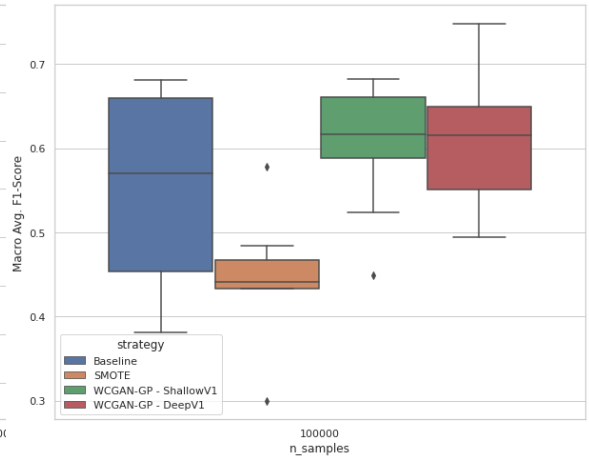
(c) Recall (Macro Avg.)



(d) Recall (Macro Avg.)



(e) F1-Score (Macro Avg.)



(f) F1-Score (Macro Avg.)

Figure 4.24: CERT R6.2, 504 features: Macro-averaged measures from classification results

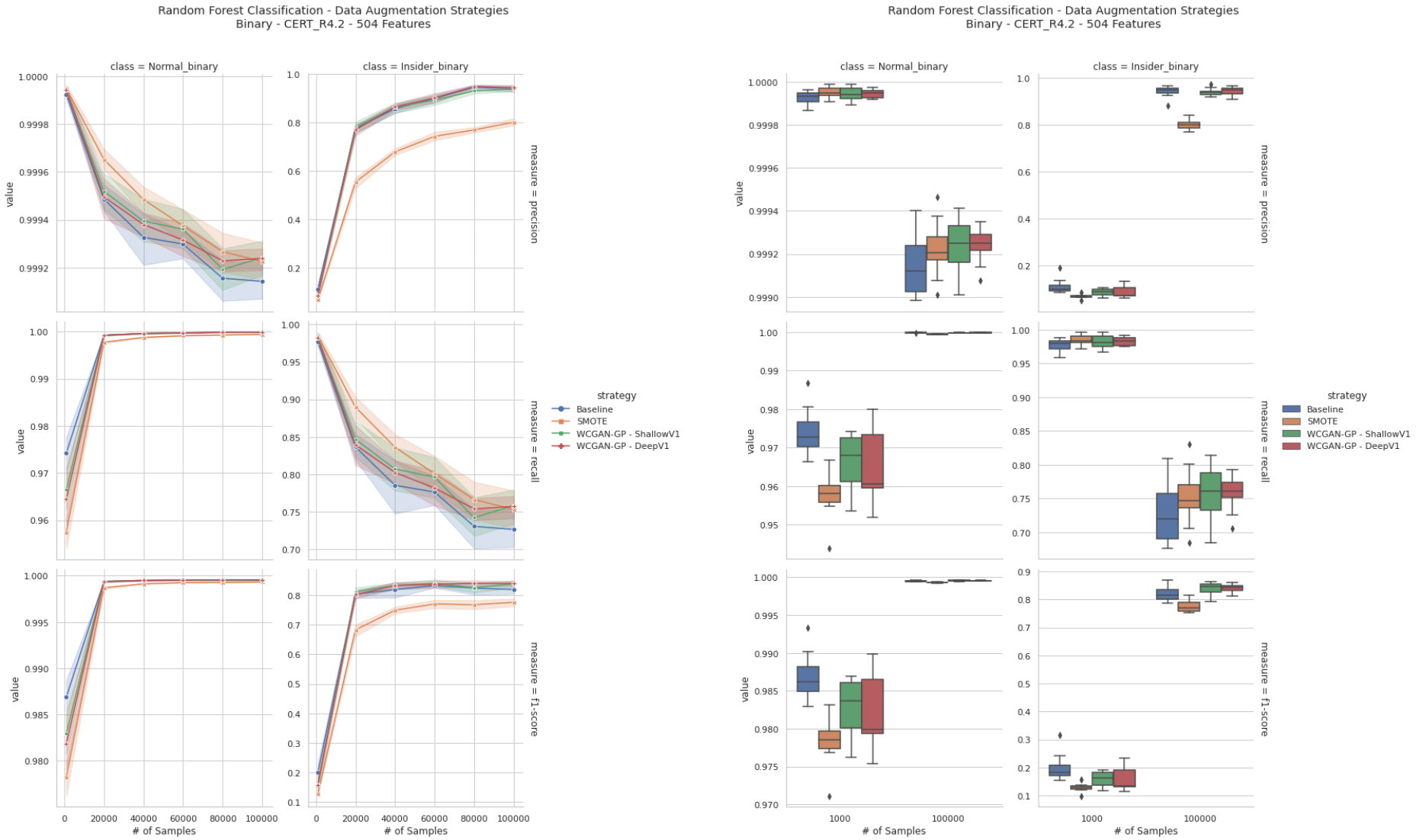


Figure 4.25: CERT R4.2, 504 features: Binary weight-averaged classification plots for all augmentation strategies

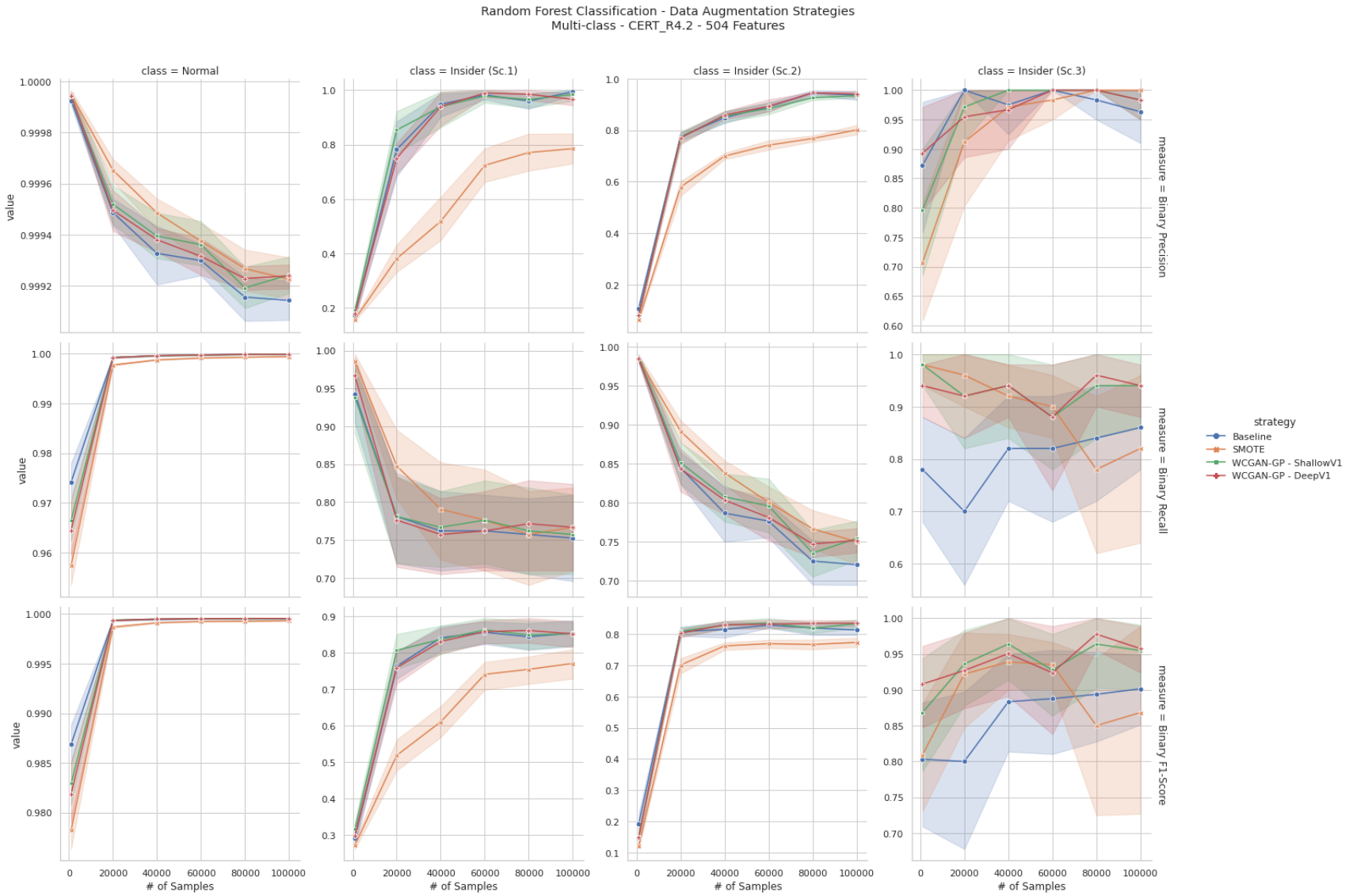


Figure 4.26: CERT R4.2, 504 features: Classification metrics as n_samples increases for all augmentation strategies

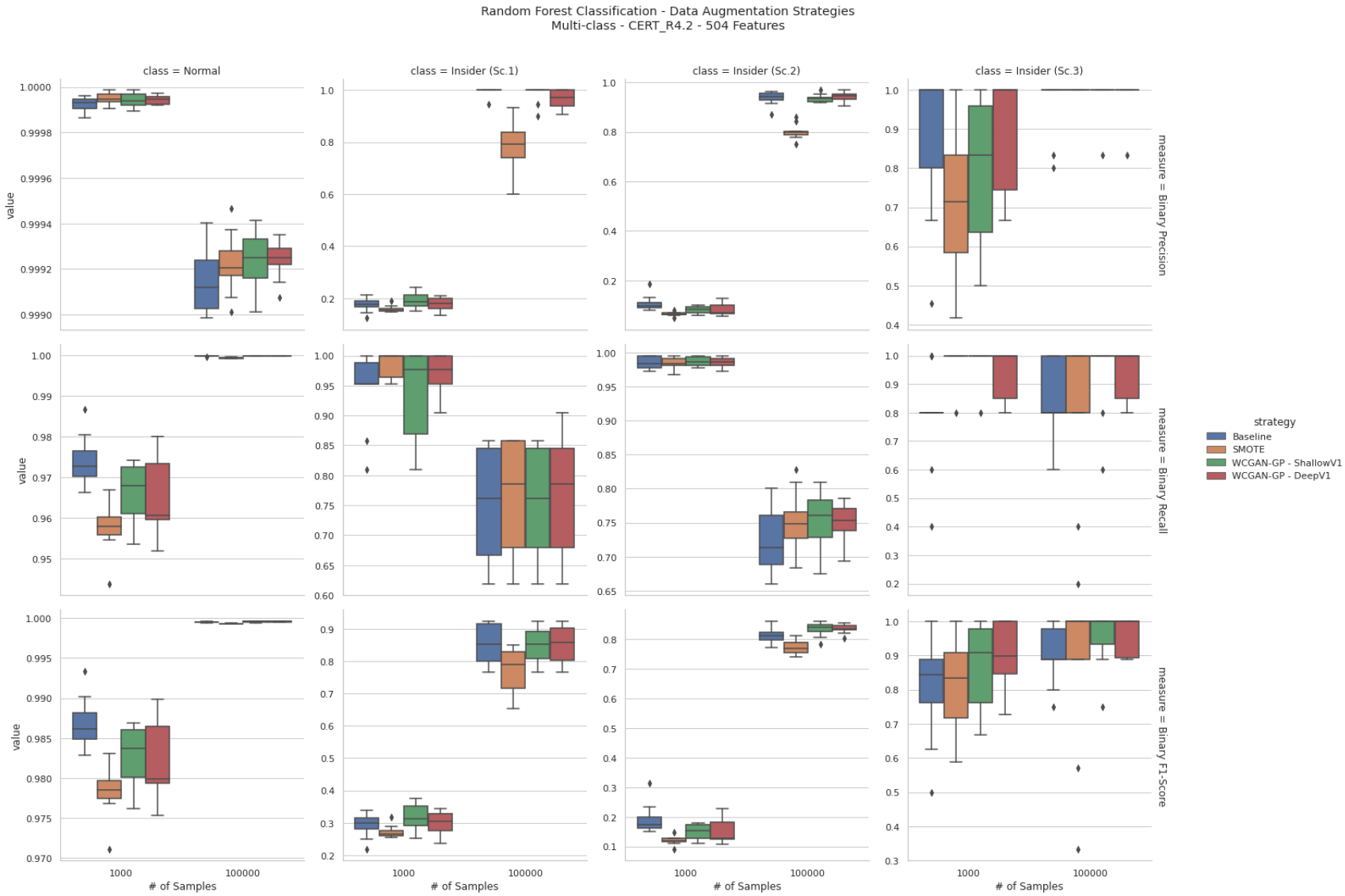
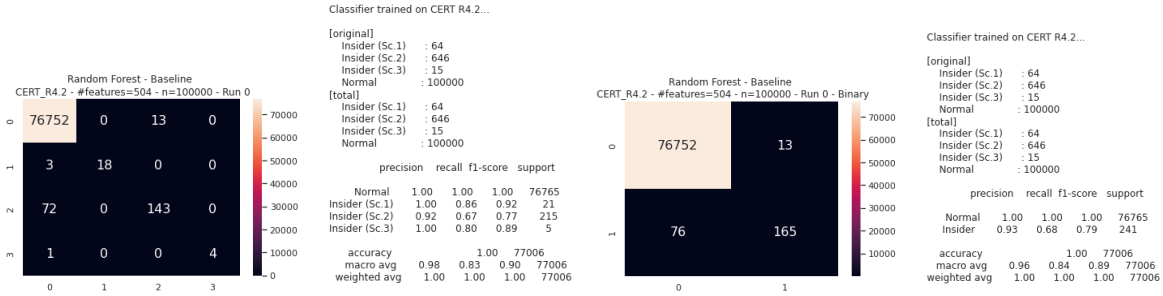


Figure 4.27: CERT R4.2, 504 features: Classification metrics at 1000 and 100000 samples per class for all augmentation strategies

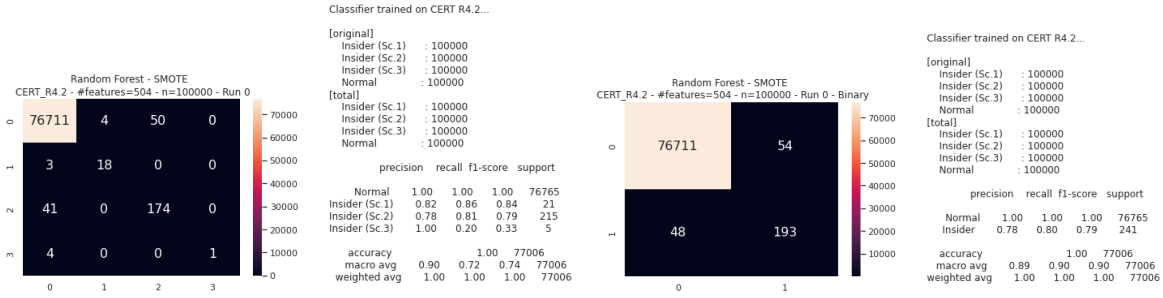
4.5 Feature Set Comparison

This section will review a head-to-head comparison between the various augmentation strategies for the 100 and 504 feature sets. Figure 4.30 shows the binary R4.2 classification results for the baseline and WCGAN-GP DeepV1 augmentation strategies. When using the binary classification metrics, the 100 feature set outperforms 504. However, the class-wise metrics for each insider class in Figures 4.31 and 4.32 show that this is primarily because of superior performance on the majority insider class (Sc. 2). The higher class count in Scenario 2 causes higher weight-averaged scores for the 100 feature set. However, the macro-averaged scores are higher for the 504 feature set because of improved performance on the minority insider classes (Sc. 1, 3, 4 and 5). Table 4.6 and 4.10 show that the 504 feature set outperforms the 100 feature set on all macro-averaged metrics of every augmentation strategy for R4.2. Interestingly, the baseline strategy had stronger macro-averaged scores when using 100 features on R5.2 and R6.2, while the other strategies performed better when using 504 features.



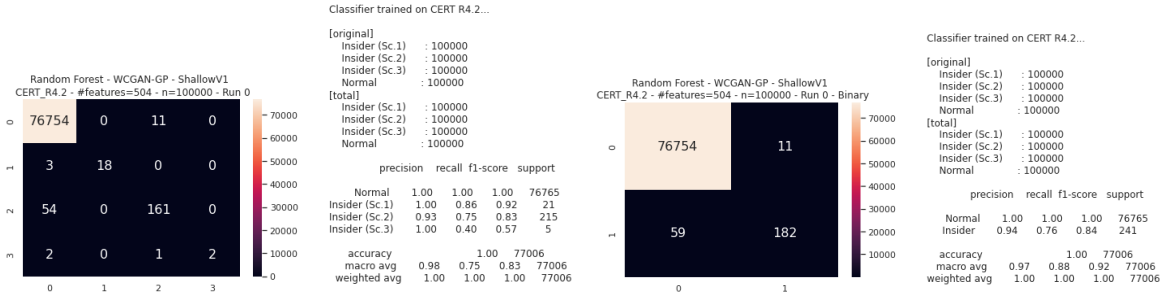
(a) Baseline - Multi-class

(b) Baseline - Binary-class



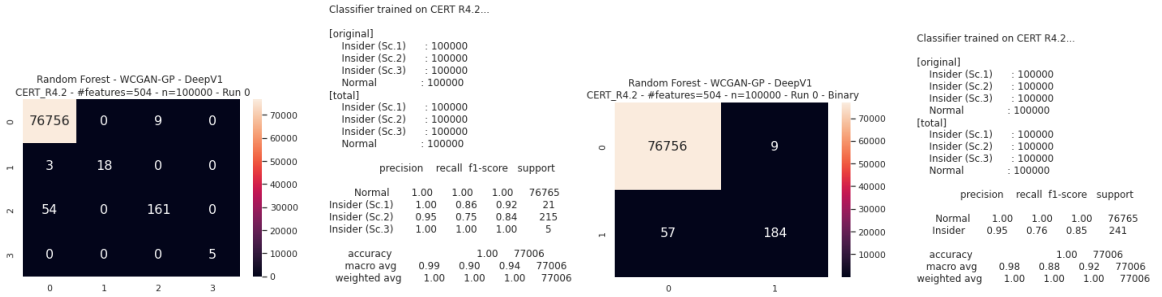
(c) Smote - Multi-class

(d) Smote - Binary-class



(e) WCGAN-GP ShallowV1 - Multi-class

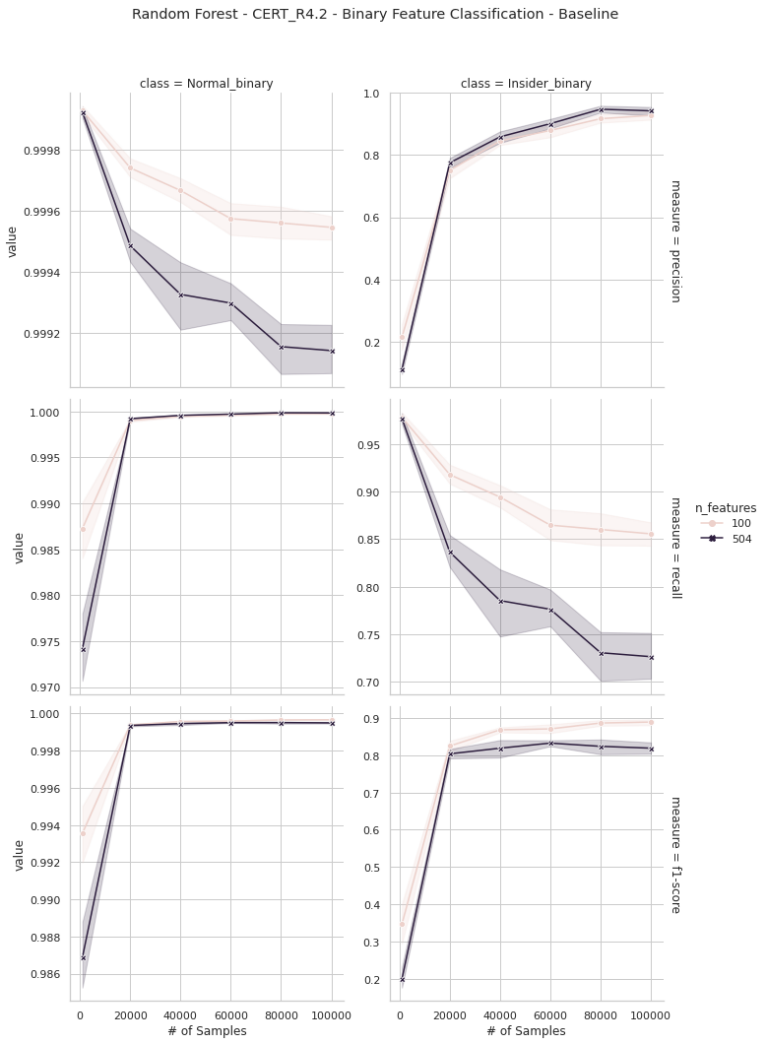
(f) WCGAN-GP ShallowV1 - Binary-class



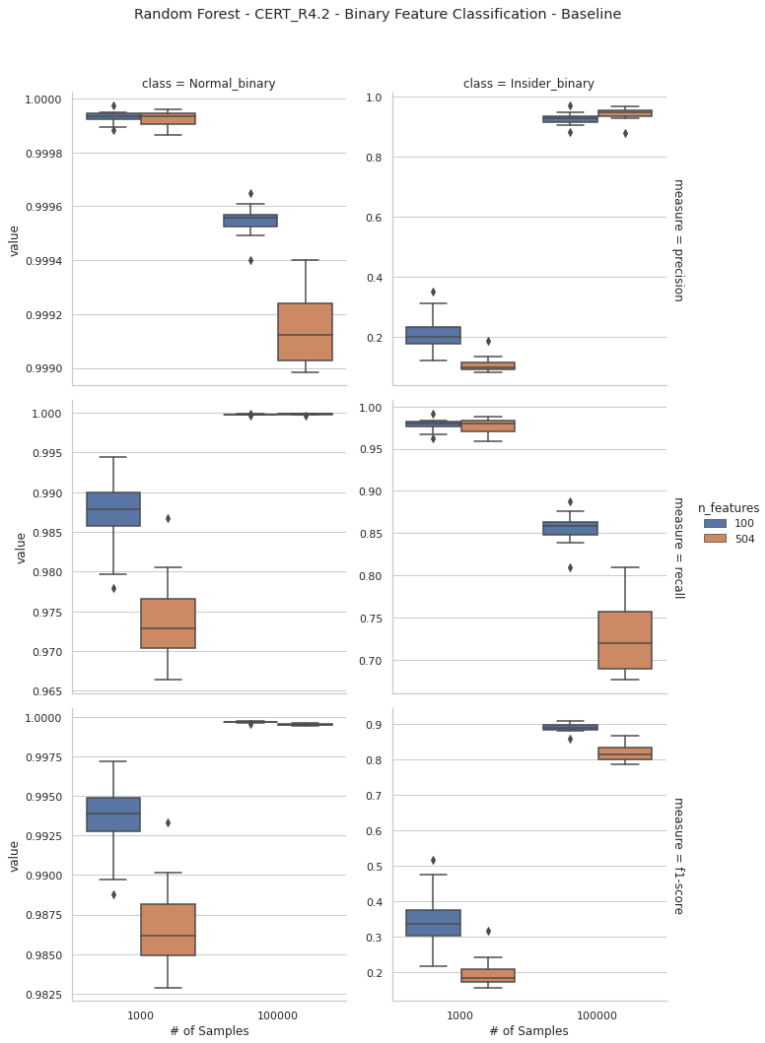
(g) WCGAN-GP DeepV1 - Multi-class

(h) WCGAN-GP DeepV1 - Binary-class

Figure 4.28: CERT R4.2, 504 features: Sample confusion matrices for Random Forest classification

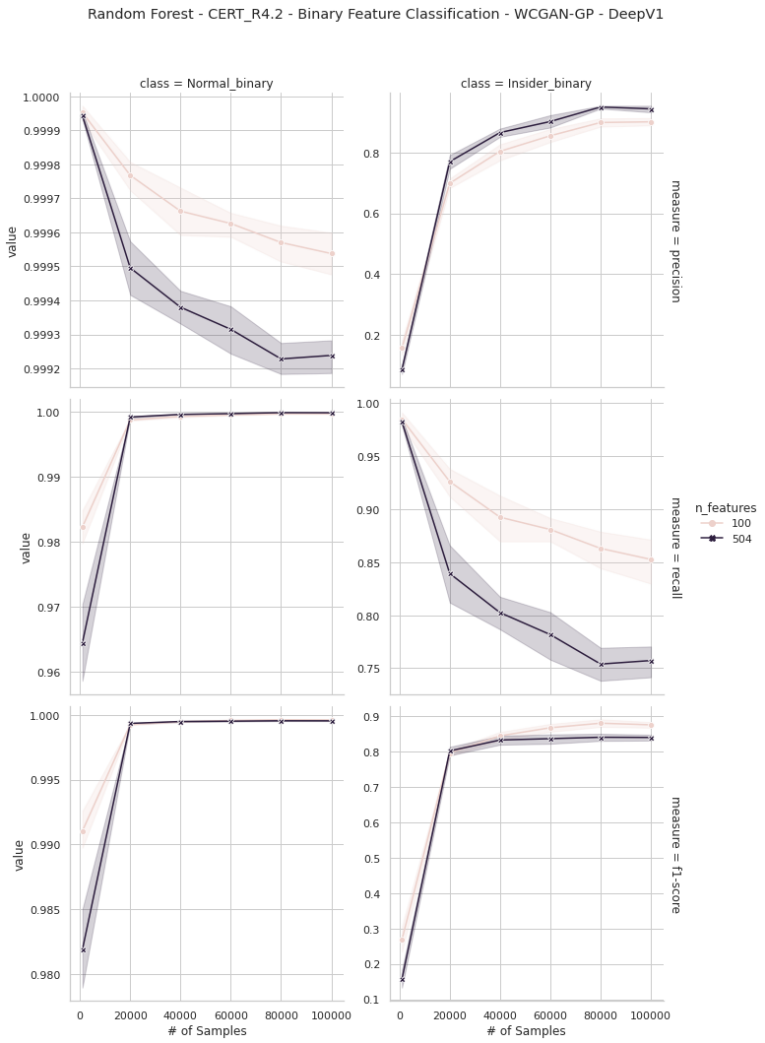


(a) Baseline Line Plot

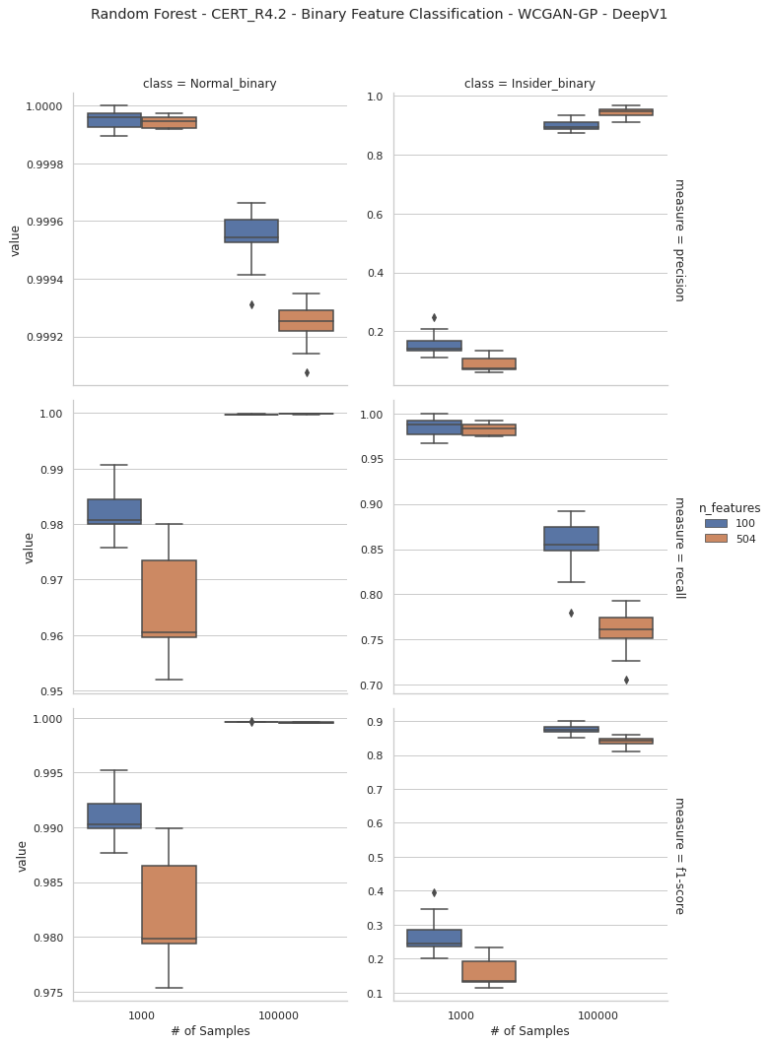


(b) Baseline Box Plot

Figure 4.29: R4.2, 100 vs 504 feature set RF binary classification metrics as n_samples is increased for the Baseline Augmentation Strategy



(a) WCGAN-GP DeepV1 Line Plot



(b) WCGAN-GP DeepV1 Box Plot

Figure 4.30: R4.2, 100 vs 504 feature set RF binary classification metrics as n_samples is increased for the WCGAN-GP DeepV1 Augmentation Strategy

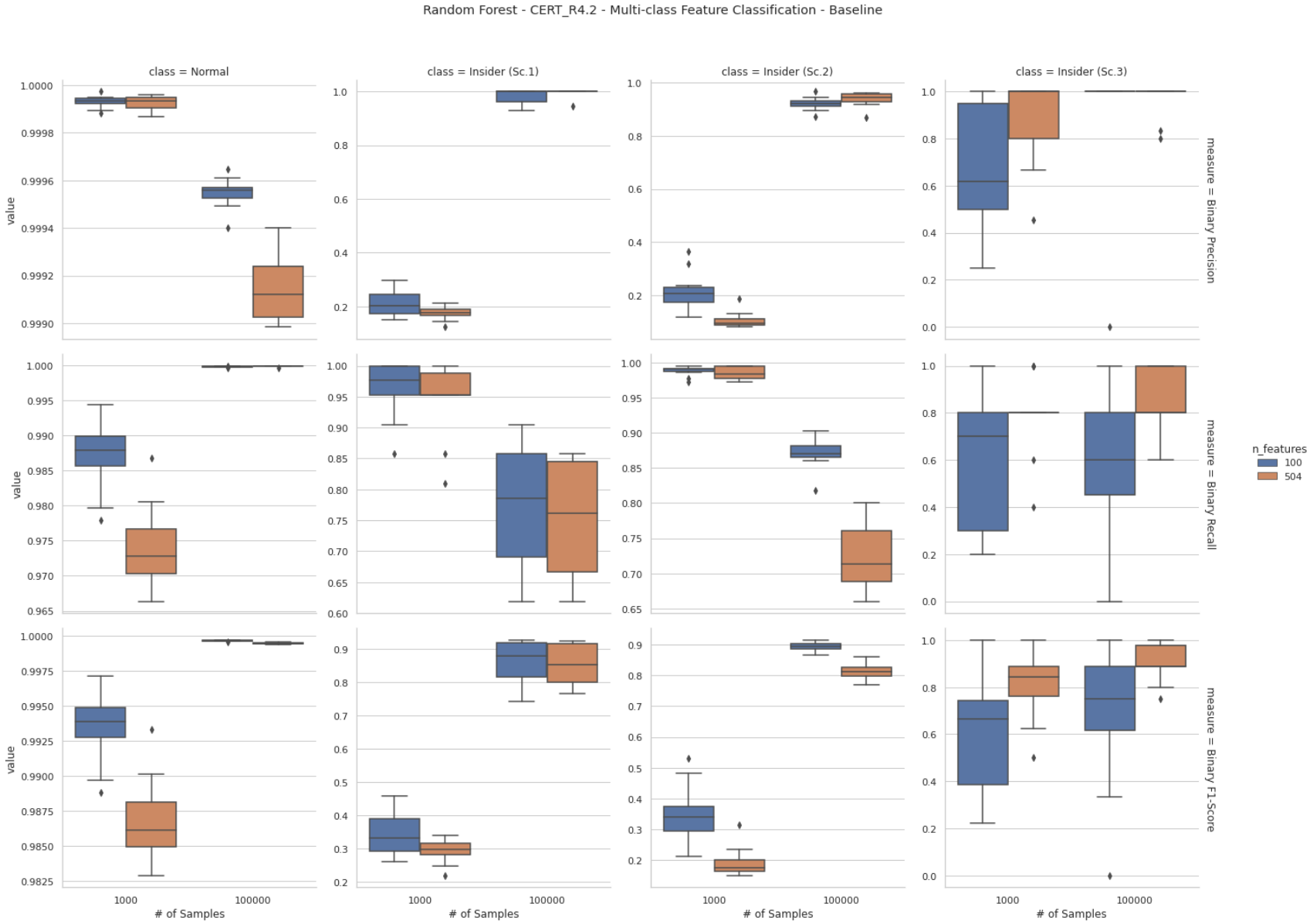


Figure 4.31: R4.2, Baseline: 100 vs 504 feature set RF binary classification metric box plots

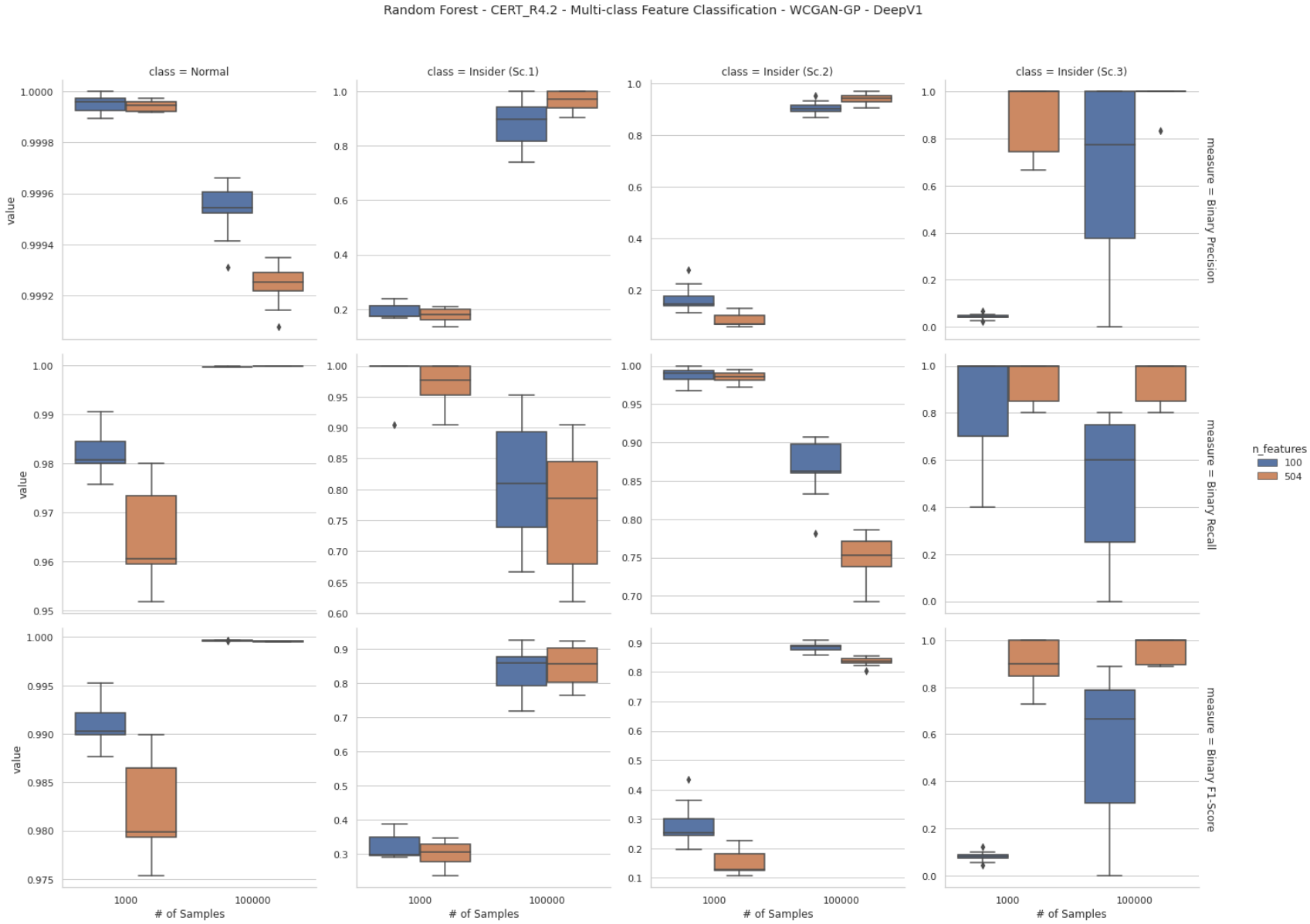


Figure 4.32: R4.2, WCGAN-GP DeepV1: 100 vs 504 feature set RF binary classification metric box plots

In summary, the CGAN architecture had highly unstable training that lead to poor performance in generating realistic insider samples. The WCGAN-GP architecture had consistently stable training without hyper-parameter tuning and was able to produce realistic synthetic insider samples. Both WCGAN-GP architectures (DeepV1 and ShallowV1) were the top performing augmentation strategies for all three CERT datasets when using the 504 feature set. The baseline strategy had similar performance on classification based on raw counts of detections, but the GANs were particularly strong in detecting minority insider classes with low support in the training dataset. This was reflected in higher macro-averaged performance metrics. The GAN's did not perform as well on the 100 feature datasets, as they were outperformed by the baseline strategy on all but minority class recall. However, the GAN strategy's performance on the 504 feature dataset was superior to the baseline strategy on the 100 feature set.

Overall, the framework proved effective at detecting the majority of insider classes with a relatively low false positive rate. It showed some ability to generalize to never before seen insider attack types, such as Sc. 5 in the R6.2 dataset. However, it had difficulty with some new attacks, namely Sc. 4 in R5.2 and R6.2. Finally, it should be noted here that none of the previous works employing similar techniques reported performance metrics broken down by insider class for all three datasets. To the best of my knowledge this thesis is the first in this regard.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The main objective of this thesis is to explore the ability of GANs to augment imbalanced insider threat detection datasets to enhance detection in a supervised context. Further, this thesis draws from past work and proposes an insider threat detection system that:

1. Transforms user behaviour into a tabular representation by computing daily percentiles of user behaviour statistics and counts [25]
2. Augments the datasets with realistic insider samples to solve the class imbalance problem
3. Detects insider threats using supervised classifiers trained on the augmented datasets

CGAN had difficulty generating samples for multiple insider scenarios due to unstable training and limited samples for some classes. However, WCGAN-GP was able to consistently learn the class distributions and generate realistic samples for all insider classes. One of the most commonly reported issues with GANs is the lack of training consistency and stability. However, over multiple runs, WCGAN-GP converged every time despite variation in hyper-parameters. Two WCGAN-GP architectures were tested, using SMOTE and an under-sampled baseline as benchmark augmentation strategies. All strategies were tested against two feature sets (100 and 504), extracted from all three CERT datasets. Both WCGAN-GP architectures performed similarly. The ShallowV1 architecture slightly outperformed DeepV1 on f1-Score for the 100 feature dataset, while DeepV1 outperformed ShallowV1 on the 504 feature set. The 504 feature set had better results for all augmentation strategies, but the WCGAN-GP architectures performed particularly well. This resulted in

WCGAN-GP achieving the top macro-averaged F1-scores of **0.911**, **0.617** and **0.614** for R4.2, R5.2 and R6.2 respectively. This shows that WCGAN-GP augmentation enhanced classifier performance against insider threats with particularly low support within the original training dataset. Although the performance drops from R4.2 to R5.2 and R6.2, the results demonstrate the frameworks ability to generalize to new organizations and insider threats. Insider scenarios 2 and 4 however, had a particularly poor detection rate amongst all strategies applied to R5.2 and R6.2. It is left to future work to investigate what insider behaviour or changes in the dataset creation led to the poor performance on these classes in particular.

5.2 Future Work

5.2.1 Framework Improvements

Due to the many components of the framework proposed in this thesis, not all avenues were explored. Many permutations of hyper-parameters and approaches were only tested during the initial research phase when only basic GANs and CGANs had been implemented. These could be revisited using the WCGAN-GP architecture, the stability of which could provide better results using different techniques. For example, multi-class trained GANs could be compared against GANs trained using binary (normal and insider) samples. It was also predicted that training a single GAN against normal and all insider classes would help the network learn a shared latent representation that would enrich the generated insider samples. If this is not the case, normal behaviour could be completely removed from the training process. Future works could also employ more sophisticated under-sampling techniques such as tomek-links or sampling from clusters [11],[7].

Temporal relations in user behaviour could also be further explored in future works. This work relied solely on a percentile sliding-window based approach to capture temporal elements in the feature vectors [27]. Once the feature vectors were constructed, the samples were shuffled randomly before classification. Future works could potentially improve classification performance by using temporal methods to capture the behaviour drift of an organization over time.

To increase observability throughout training, a training hook could be introduced to run augmented dataset generation and subsequent classification experiments throughout the training process. Although costly, this would provide insights as to the relation between performance and training duration. t-SNE provides a qualitative guide to identify mode collapse, but augmentation validation could provide us with a programmatic stopping condition to prevent overfitting. In addition, ROC curves and AUC measures could be used to better understand the performance of each classifier. Further, simplified static metrics such as MCC and Kappa Coefficient could be used to reduce the complexity of the analysis while capturing the important features of multi-class classification on an imbalanced dataset.

5.2.2 Dataset Generation

GANs could be leveraged in the synthetic dataset generation itself. This could be by training on real-world insider threat scenarios and releasing public datasets consisting of GAN generated samples. A reasonable amount of privacy could be ensured for the organizations and their users by using differentially private GANs. The GANs could also be used to replicate real world user behaviour in particular domains. This work avoided NLP techniques because the content of web pages and emails were created using bag-of-word models. Although this could be sufficient for some key-word based approaches, it would not reward any approaches that looked to contextualize the content. Recent advancements in GAN research have made them one of the best class of models for generating text [20]. GANs could be used to produce more realistic content that could reward context-sensitive NLP approaches to insider threat detection. GANs could also be used to generate more realistic relationship graphs to drive behaviour between users and systems [15]. GANs have a demonstrated ability to learn and generate graphs [19]. This could be used to learn graph structures from a wide variety of organization types and sizes and generate a broader test-bed for exploring insider threat detection algorithms.

5.2.3 Adversarial Attacks

Goodfellow et al. showed that neural networks have a weakness against small perturbations in the input data that can lead them to misclassify the input [17]. Fladby

et al. showed that such a vulnerability could be exploited by an attacker to perturb network flow data and evade IDS systems [13]. Although such an attack is less likely against a system with a human in the loop, it is plausible that an insider could modify their behaviour as to evade detection. Future works should investigate the robustness of existing unsupervised and supervised insider threat detection systems against such attacks. GANs have been proven effective at generating adversarial samples with high attack-success rates [5]. While this could be used to assess the robustness of existing detection methods, it could also be used by threat actors. Luckily, GANs can also be used to bolster the defences and robustness of machine-learning based classification systems [36]. GAN-based anomaly detectors have also been proposed, and could lend themselves well to creating robust insider threat detection systems [2], [3].

Bibliography

[1] Mohammad Adiban, Arash Safari, and Giampiero Salvi. Step-gan: A step-by-step training for multi generator gans with application to cyber security in power systems. *ArXiv*, abs/2009.05184, 2020.

[2] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 622–637, Cham, 2019. Springer International Publishing.

[3] Samet Akçay, Amir Atapour-Abarghouei, and Toby P. Breckon. Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[5] Tao Bai, Jun Zhao, Jinlin Zhu, Shoudong Han, Jiefeng Chen, Bo Li, and Alex Kot. Ai-gan: Attack-inspired generation of adversarial examples, 2020.

[6] Zhipeng Cai, Zuobin Xiong, Honghui Xu, Peng Wang, Wei Li, and Yi Pan. Generative adversarial networks: A survey toward private and secure applications. *ACM Comput. Surv.*, 54(6), jul 2021.

[7] Pratik Chattopadhyay, Lipo Wang, and Yap-Peng Tan. Scenario-based insider threat detection from cyber activities. *IEEE Transactions on Computational Social Systems*, 5(3):660–675, 2018.

[8] Ravi Chauhan and Shahram Shah Heydari. Polymorphic adversarial ddos attack on ids using gan. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, 2020.

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

[10] Indira Kalyan Dutta, Bhaskar Ghosh, Albert Carlson, Michael Totaro, and Magdy Bayoumi. Generative adversarial networks in security: A survey. In *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0399–0405, 2020.

[11] Tusneem Elhassan, Aljourf M, Al-Mohanna F, and Mohamed Shoukri. Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method. *Global Journal of Technology and Optimization*, 01, 01 2016.

[12] Liyue Fan. A survey of differentially private generative adversarial networks. 2020.

[13] Torgeir Fladby, Hårek Haugerud, Stefano Nichele, Kyrre Begnum, and Anis Yazidi. Evading a machine learning-based intrusion detection system through adversarial perturbations. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '20, page 161–166, New York, NY, USA, 2020. Association for Computing Machinery.

[14] R G Gayathri, Atul Sajjanhar, Yong Xiang, and Xingjun Ma. Anomaly detection for scenario-based insider activities using cgan augmented data. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 718–725, 2021.

[15] Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. In *2013 IEEE Security and Privacy Workshops*, pages 98–104, 2013.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[17] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[18] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5769–5779, Red Hook, NY, USA, 2017. Curran Associates Inc.

[19] Xiaojie Guo and Liang Zhao. A systematic survey on deep generative models for graph generation. *ArXiv*, abs/2007.06686, 2020.

[20] Touseef Iqbal and Shaima Qureshi. The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences*, 2020.

[21] Abdul Jabbar, Xi Li, and Bourahla Omar. A survey on generative adversarial networks: Variants, applications, and training. *ACM Comput. Surv.*, 54(8), oct 2021.

[22] Duc C. Le. Feature extraction for cert insider threat dataset. `https://github.com/lcd-dal/feature-extraction-for-CERT-insider-threat-test-dataset`, 2021.

[23] Duc C. Le, Sara Khanchi, A. Nur Zincir-Heywood, and Malcolm I. Heywood. Benchmarking evolutionary computation approaches to insider threat detection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, page 1286–1293, New York, NY, USA, 2018. Association for Computing Machinery.

[24] Duc C. Le, A. Nur Zincir-Heywood, and Malcolm I. Heywood. Dynamic insider threat detection based on adaptable genetic programming. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2579–2586, 2019.

[25] Duc C. Le and Nur Zincir-Heywood. Exploring adversarial properties of insider threat detection. In *2020 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2020.

[26] Duc C. Le and Nur Zincir-Heywood. Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management*, 18(2):1152–1164, 2021.

[27] Duc C. Le, Nur Zincir-Heywood, and Malcolm I. Heywood. Analyzing data granularity levels for insider threat detection using machine learning. *IEEE Transactions on Network and Service Management*, 17(1):30–44, 2020.

[28] Brian Lindauer. Insider threat test dataset. carnegie mellon university. dataset. http://dx.doi.org/10.4225/13/511C71F8612C3, 2020.

[29] Yi Liu, Jialiang Peng, James J.Q. Yu, and Yi Wu. Ppgan: Privacy-preserving generative adversarial network. *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2019.

[30] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[31] Simon Msika, Alejandro Quintero, and Foutse Khomh. Sigma : Strengthening ids with gan and metaheuristics attacks. *ArXiv*, abs/1912.09303, 2019.

[32] Abhishek Singh, Debojyoti Dutta, and Amit Saha. Migan: Malware image synthesis using gans. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:10033–10034, 07 2019.

[33] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[34] Manhar Walia, Brendan Tierney, and Susan Mckeever. Synthesising tabular data using wasserstein conditional gans with gradient penalty (wcgan-gp). 12 2020.

[35] Fangfang Yuan, Yanmin Shang, Yanbing Liu, Yanan Cao, and Jianlong Tan. Data augmentation for insider threat detection with gan. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 632–638, 2020.

[36] Haibin Zheng, Jinyin Chen, Hang Du, Weipeng Zhu, Shouling Ji, and Xuhong Zhang. Grip-gan: An attack-free defense through general robust inverse perturbation. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2021.

[37] Ming Zheng, Tong Li, Rui Zhu, Yahui Tang, Mingjing Tang, Leilei Lin, and Zifei Ma. Conditional wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification. *Information Sciences*, 512:1009–1023, 2020.

Appendix A

A.1 Classification Results for All Classifiers

A.2 100 vs 504 Feature Classification Results

A.3 R5.2 and R6.2 Confusion Matrices

A.3.1 100 Feature Set

A.3.2 504 Feature Set

A.4 R5.2 and R6.2 Classification Plots

A.4.1 100 Feature Set

A.4.2 504 Feature Set

Table A.1: CERT R4.2, 100 features, n_samples=10000 - Insider Threat Detection Data Augmentation Strategy Classification Results

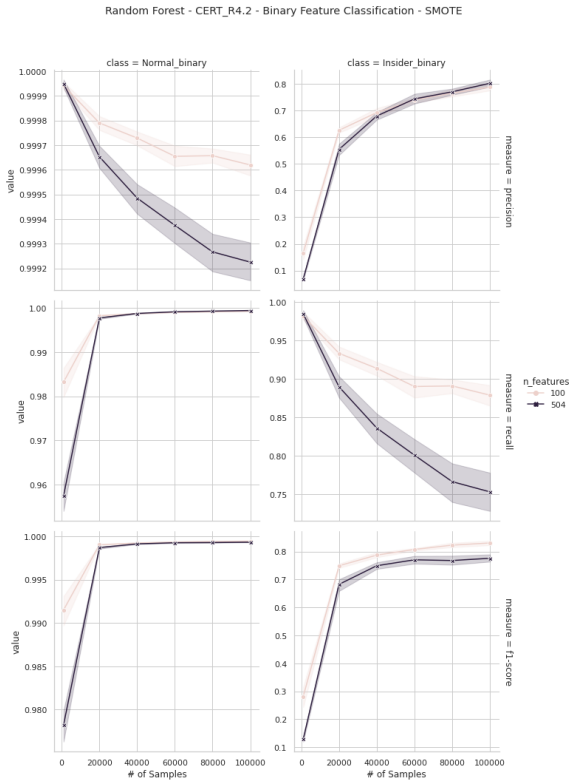
	Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)		
	Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R
strategy	classifier												
Baseline	Logistic Regression	0.998	0.998	0.997	0.846	1.000	0.738	0.256	0.235	0.282	0.589	0.833	0.480
	Random Forest	0.999	1.000	0.999	0.728	0.662	0.886	0.803	0.698	0.946	0.682	0.867	0.600
	SVM	0.999	0.998	1.000	0.859	1.000	0.757	0.636	0.965	0.475	0.217	0.138	0.580
	XGBoost	0.334	1.000	0.231	0.001	0.001	1.000	0.994	0.991	0.998	0.003	0.002	1.000
SMOTE	Logistic Regression	0.906	0.999	0.829	0.036	0.018	0.862	0.025	0.013	0.701	0.041	0.021	0.800
	Random Forest	0.999	1.000	0.997	0.490	0.349	0.900	0.709	0.561	0.965	0.300	0.240	0.520
	SVM	0.968	0.999	0.939	0.089	0.047	0.757	0.059	0.031	0.643	0.212	0.130	0.660
	XGBoost	0.000	0.000	0.000	0.702	0.701	1.000	1.000	1.000	1.000	0.000	0.000	1.000
WCGAN-GP - DeepV1	Logistic Regression	0.985	0.998	0.973	0.054	0.028	0.762	0.105	0.061	0.389	0.049	0.025	0.720
	Random Forest	0.999	1.000	0.998	0.522	0.389	0.895	0.764	0.637	0.958	0.377	0.280	0.660
	SVM	0.993	0.998	0.987	0.187	0.111	0.771	0.219	0.161	0.444	0.132	0.075	0.660
	XGBoost	0.000	0.000	0.000	0.001	0.001	1.000	0.020	0.010	1.000	0.034	0.017	1.000

Table A.2: CERT R5.2 - 100 features - Insider Threat Detection Data Augmentation Strategy Classification Results

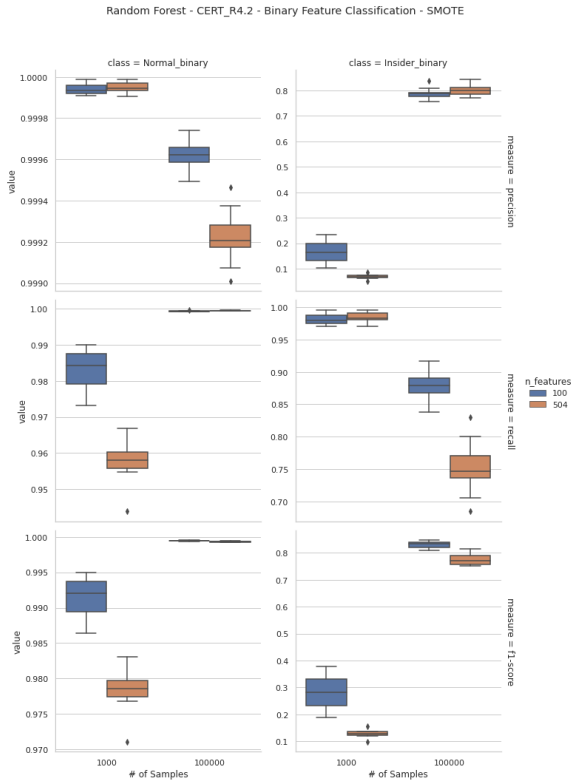
	Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)			Insider (Sc.4)		
	Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
strategy	classifier															
Baseline	Logistic Regression	0.999	0.998	0.999	0.805	1.000	0.674	0.247	0.264	0.233	0.508	0.899	0.355	0.000	0.000	0.000
	Random Forest	0.995	0.999	0.991	0.653	0.596	0.776	0.278	0.179	0.854	0.619	0.781	0.535	0.029	0.800	0.015
	SVM	0.999	0.999	1.000	0.824	0.940	0.738	0.657	0.962	0.499	0.149	0.092	0.485	0.004	0.600	0.002
	XGBoost	0.162	1.000	0.101	0.000	0.000	0.996	0.958	0.924	0.999	0.001	0.000	1.000	0.962	1.000	0.930
SMOTE	Logistic Regression	0.842	0.999	0.728	0.012	0.006	0.836	0.006	0.003	0.591	0.020	0.010	0.810	0.222	1.000	0.126
	Random Forest	0.997	0.999	0.995	0.543	0.425	0.816	0.332	0.209	0.826	0.175	0.153	0.290	0.017	0.800	0.009
	SVM	0.932	0.999	0.874	0.026	0.013	0.699	0.013	0.007	0.568	0.131	0.079	0.515	0.195	1.000	0.110
	XGBoost	0.000	0.000	0.000	0.700	0.700	1.000	1.000	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
WCGAN-GP - DeepV1	Logistic Regression	0.981	0.999	0.963	0.028	0.014	0.811	0.038	0.020	0.321	0.018	0.009	0.595	0.332	1.000	0.202
	Random Forest	0.995	0.999	0.991	0.510	0.393	0.809	0.233	0.135	0.899	0.225	0.146	0.590	0.066	1.000	0.035
	SVM	0.992	0.999	0.985	0.069	0.038	0.751	0.139	0.089	0.430	0.044	0.024	0.485	0.258	1.000	0.163
	XGBoost	0.000	0.100	0.000	0.001	0.000	1.000	0.011	0.005	1.000	0.014	0.007	1.000	1.000	1.000	1.000

Table A.3: CERT R6.2 - 100 features - Insider Threat Detection Data Augmentation Strategy Classification Results

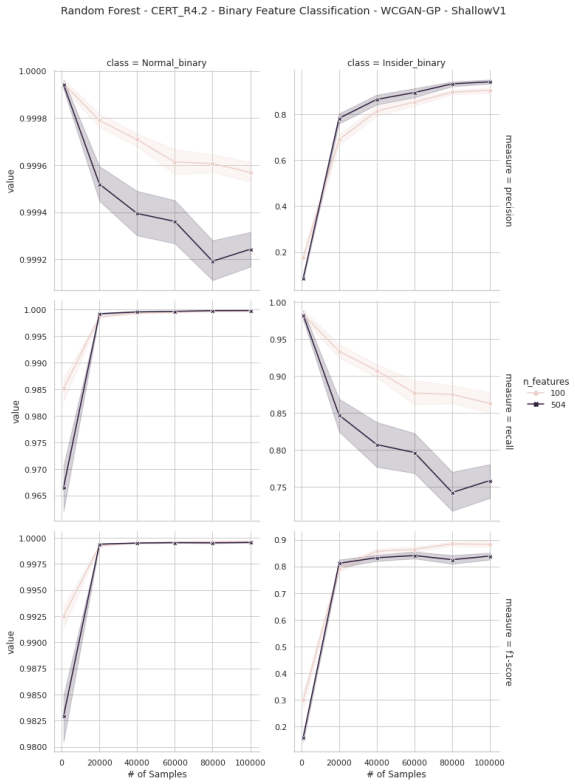
	Class	Normal			Insider (Sc.1)			Insider (Sc.2)			Insider (Sc.3)			Insider (Sc.4)			Insider (Sc.5)		
	Measure	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
strategy	classifier																		
Baseline	Logistic Regression	1.000	1.000	0.999	0.770	1.000	0.633	0.010	0.005	0.285	0.000	0.000	0.000	0.000	0.000	0.000	0.900	0.900	0.900
	Random Forest	1.000	1.000	1.000	0.684	0.740	0.667	0.139	0.076	0.865	0.450	0.650	0.350	0.000	0.000	0.000	1.000	1.000	1.000
	SVM	1.000	1.000	1.000	0.760	0.929	0.667	0.605	0.688	0.540	0.004	0.002	0.250	0.000	0.000	0.000	1.000	1.000	1.000
	XGBoost	0.278	1.000	0.191	0.000	0.000	1.000	0.613	0.510	1.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
SMOTE	Logistic Regression	0.937	1.000	0.882	0.000	0.000	0.733	0.000	0.000	0.535	0.000	0.000	0.400	0.900	0.900	0.900	1.000	1.000	1.000
	Random Forest	0.999	1.000	0.998	0.661	0.690	0.667	0.017	0.008	0.775	0.029	0.017	0.250	0.000	0.000	0.000	0.800	0.800	0.800
	SVM	0.969	1.000	0.939	0.000	0.000	0.633	0.000	0.000	0.520	0.003	0.002	0.250	0.200	0.200	0.200	1.000	1.000	1.000
	XGBoost	0.000	0.000	0.000	0.700	0.700	1.000	1.000	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
WCGAN-GP - DeepV1	Logistic Regression	0.973	1.000	0.948	0.000	0.000	0.767	0.000	0.000	0.310	0.000	0.000	0.300	1.000	1.000	1.000	0.900	0.900	0.900
	Random Forest	1.000	1.000	0.999	0.396	0.296	0.667	0.057	0.030	0.905	0.017	0.008	0.600	0.000	0.000	0.000	1.000	1.000	1.000
	SVM	0.989	1.000	0.978	0.000	0.000	0.667	0.002	0.001	0.410	0.001	0.001	0.250	0.700	0.700	0.700	1.000	1.000	1.000
	XGBoost	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	1.000	0.001	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000



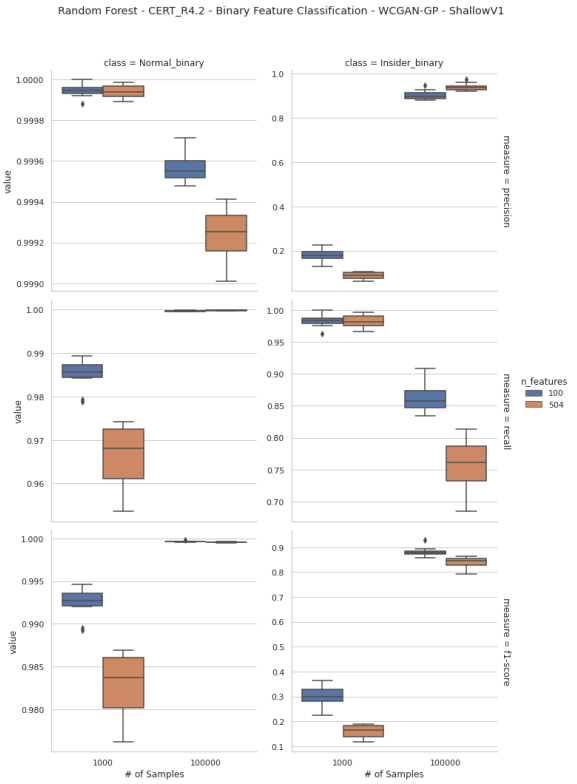
(a) SMOTE Line Plot



(b) SMOTE Box Plot

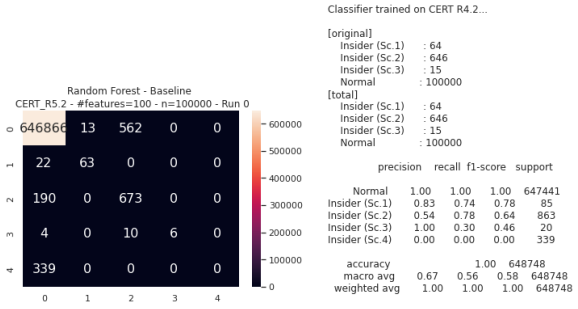


(c) WCGAN-GP ShallowV1 Line Plot

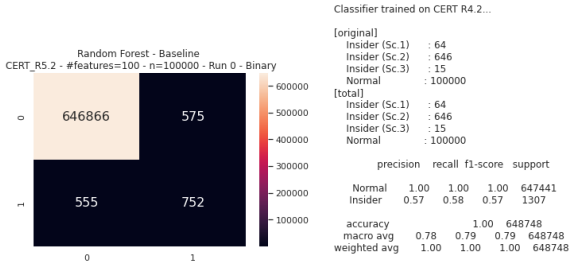


(d) WCGAN-GP ShallowV1 Box Plot

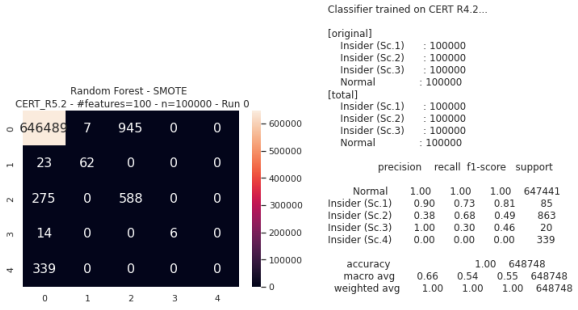
Figure A.1: R4.2, 100 vs 504 feature set RF binary classification metrics as n_samples is increased for SMOTE and WCGAN-GP ShallowV1



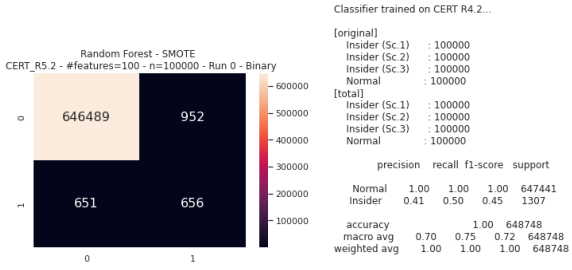
(a) Baseline - Multi-class



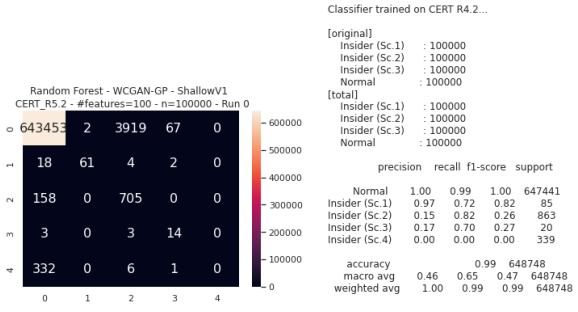
(b) Baseline - Binary-class



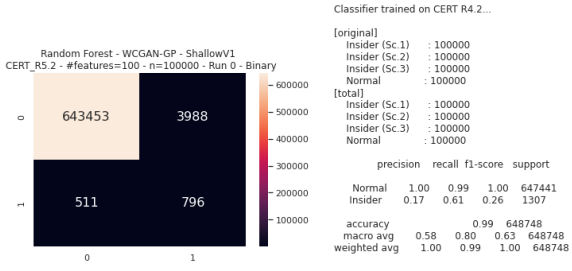
(c) Smote - Multi-class



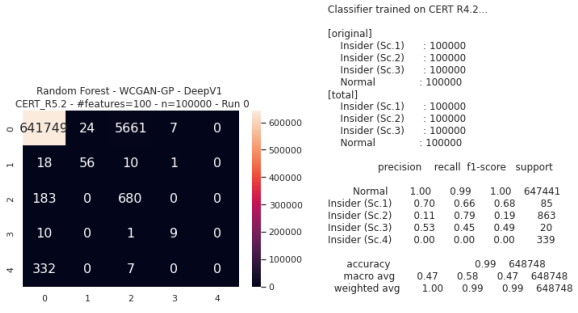
(d) Smote - Binary-class



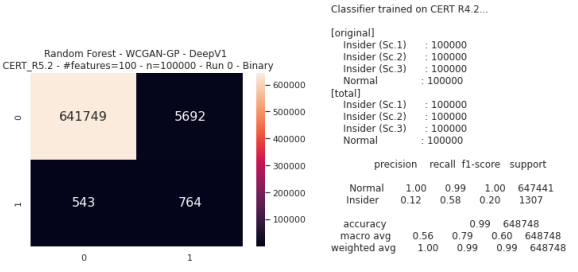
(e) WCGAN-GP ShallowV1 - Multi-class



(f) WCGAN-GP ShallowV1 - Binary-class

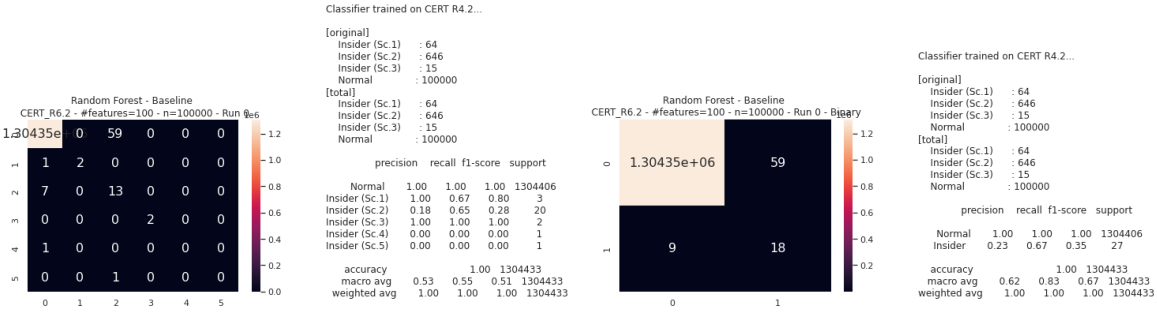


(g) WCGAN-GP DeepV1 - Multi-class

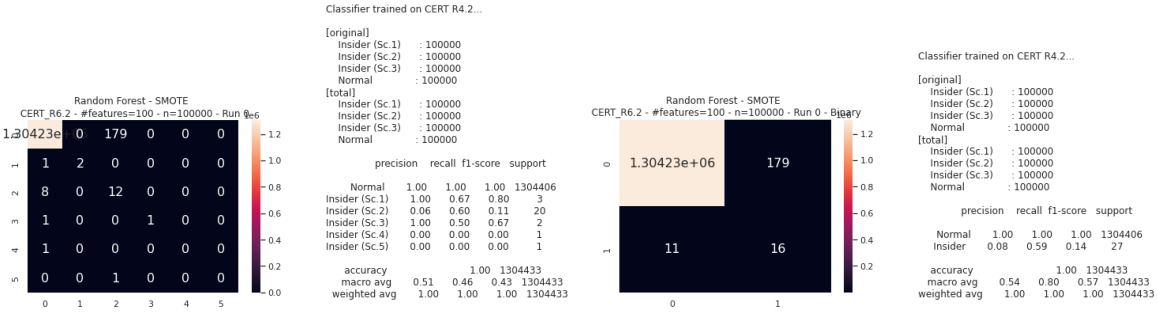


(h) WCGAN-GP DeepV1 - Binary-class

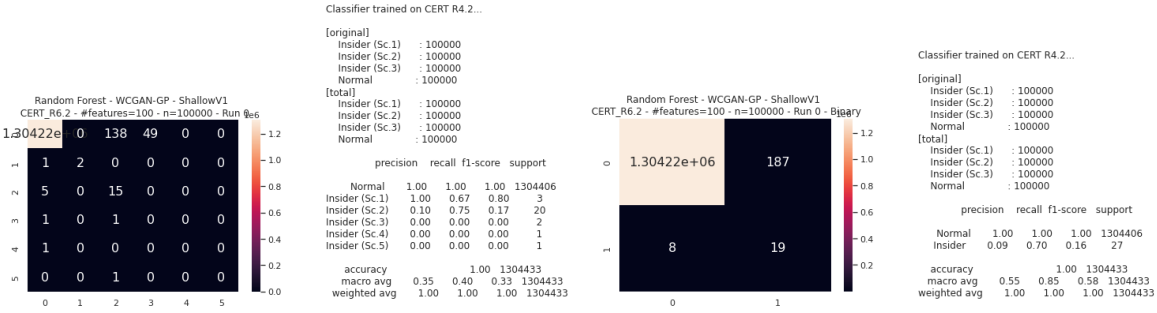
Figure A.2: CERT R5.2, 100 features: Sample confusion matrices for random forest classification



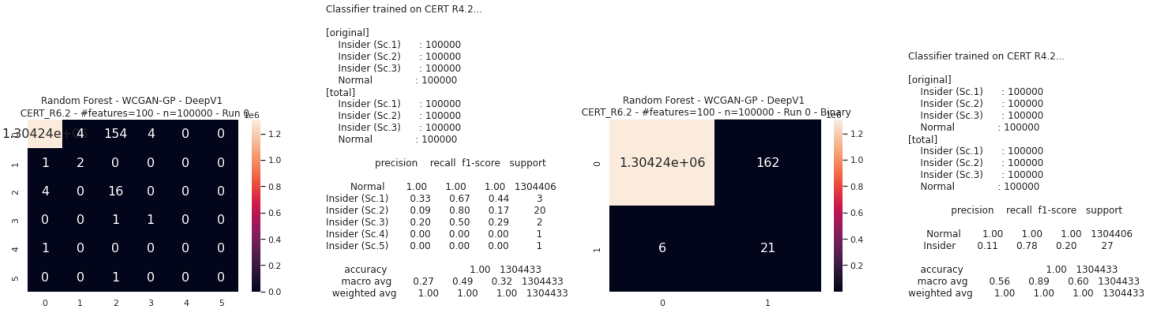
(a) Baseline - Multi-class (b) Baseline - Binary-class



(c) Smote - Multi-class (d) Smote - Binary-class

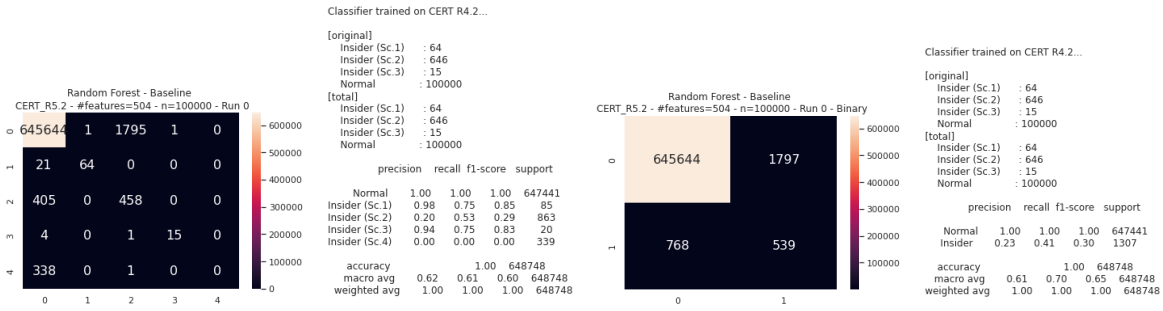


(e) WCGAN-GP ShallowV1 - Multi-class (f) WCGAN-GP ShallowV1 - Binary-class



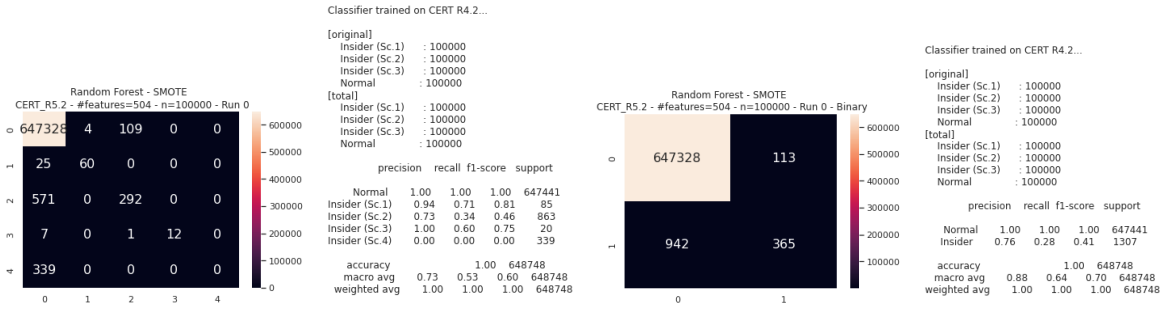
(g) WCGAN-GP DeepV1 - Multi-class (h) WCGAN-GP DeepV1 - Binary-class

Figure A.3: CERT R6.2, 100 features: Sample confusion matrices for random forest classification



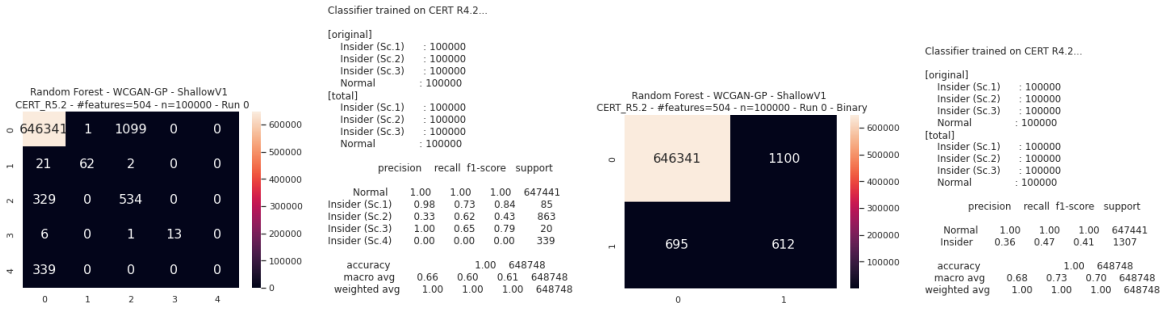
(a) Baseline - Multi-class

(b) Baseline - Binary-class



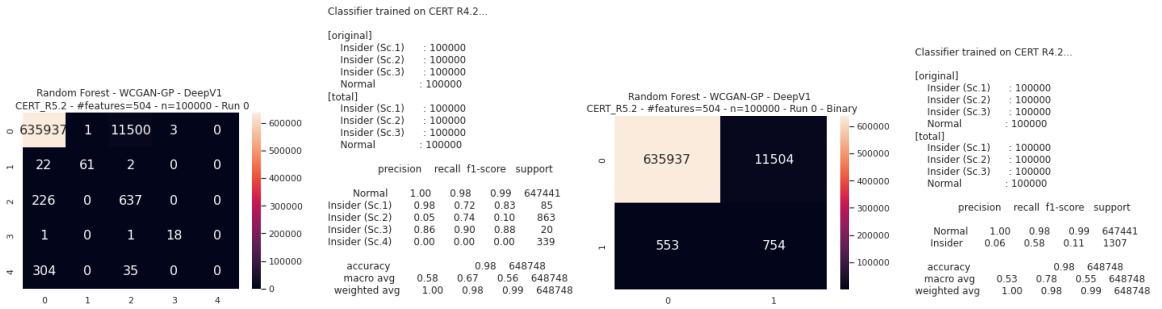
(c) Smote - Multi-class

(d) Smote - Binary-class



(e) WCGAN-GP ShallowV1 - Multi-class

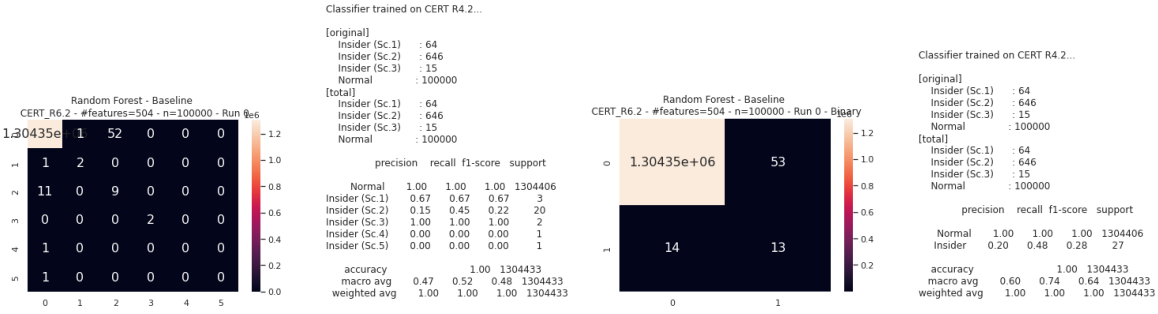
(f) WCGAN-GP ShallowV1 - Binary-class



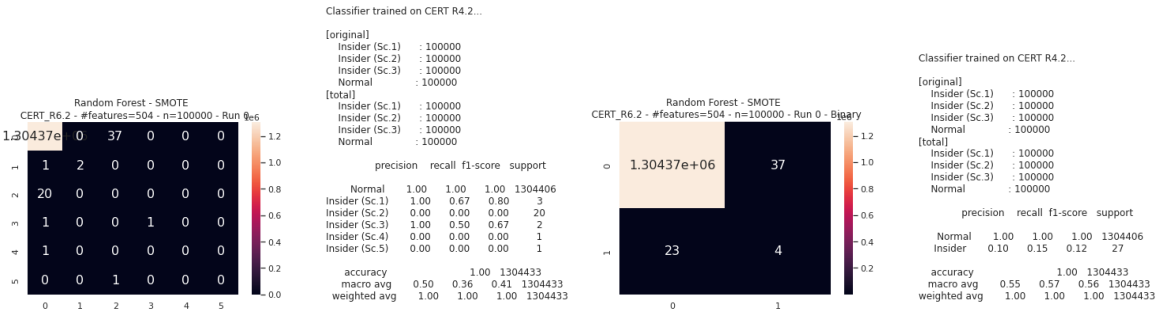
(g) WCGAN-GP DeepV1 - Multi-class

(h) WCGAN-GP DeepV1 - Binary-class

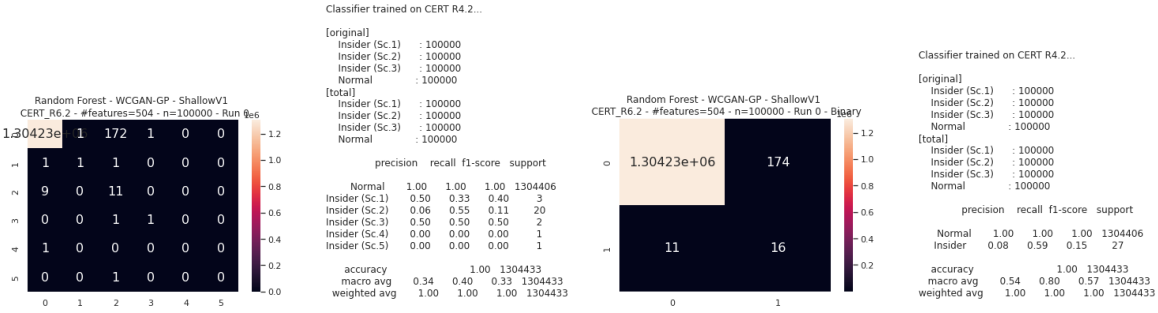
Figure A.4: CERT R5.2, 504 features: Sample confusion matrices for random forest classification



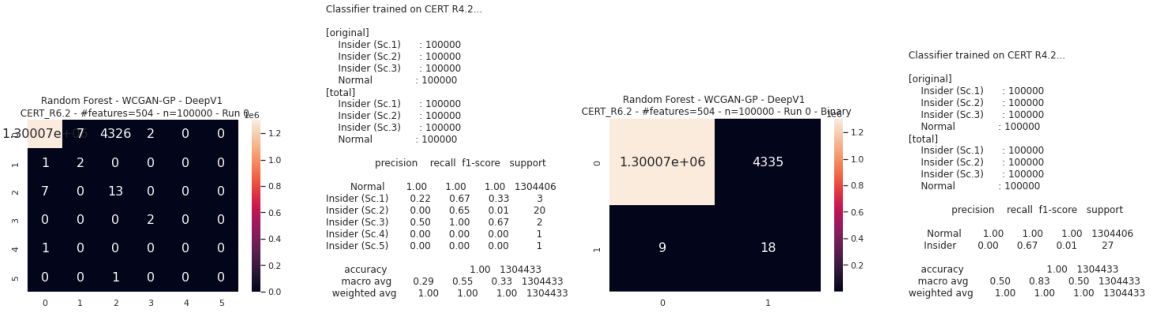
(a) Baseline - Multi-class (b) Baseline - Binary-class



(c) Smote - Multi-class (d) Smote - Binary-class



(e) WCGAN-GP ShallowV1 - Multi-class (f) WCGAN-GP ShallowV1 - Binary-class



(g) WCGAN-GP DeepV1 - Multi-class (h) WCGAN-GP DeepV1 - Binary-class

Figure A.5: CERT R6.2, 504 features: Sample confusion matrices for random forest classification

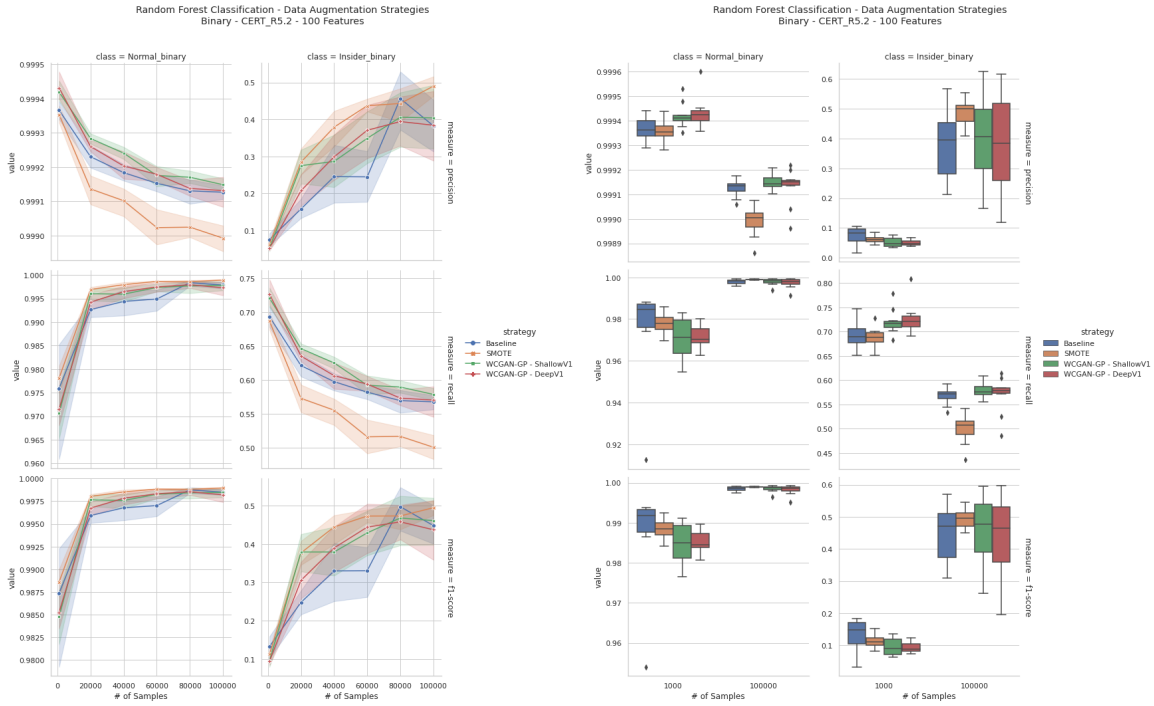


Figure A.6: CERT R5.2, 100 features: Binary weight-averaged classification plots for all augmentation strategies

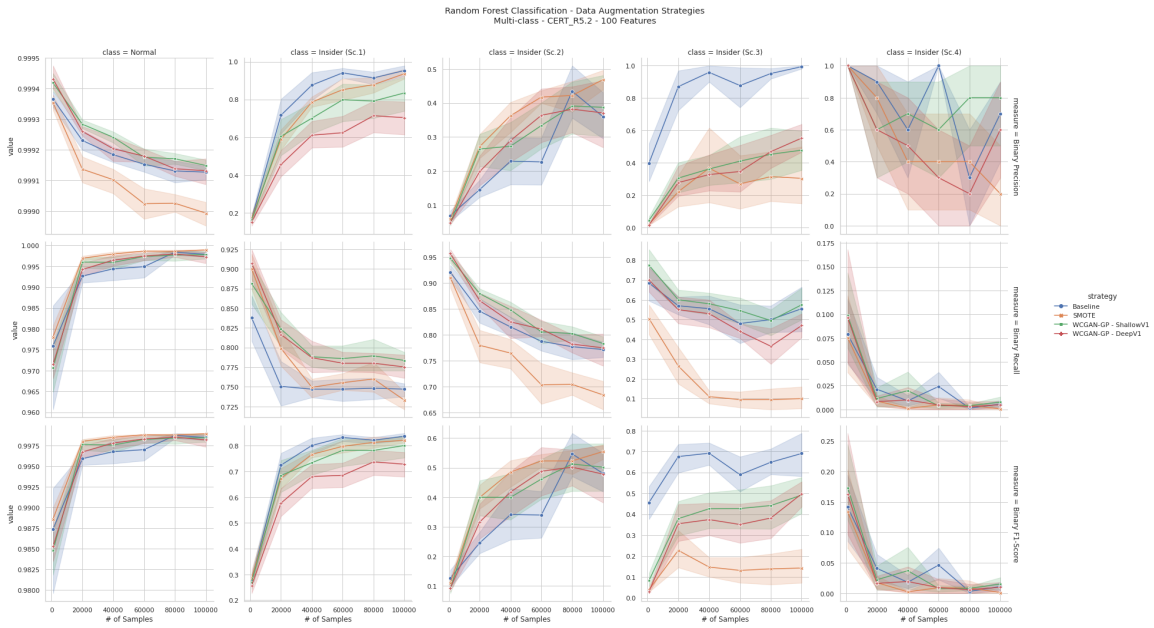


Figure A.7: CERT R5.2, 100 features: Weight-averaged classification metrics as n_samples increases for all augmentation strategies

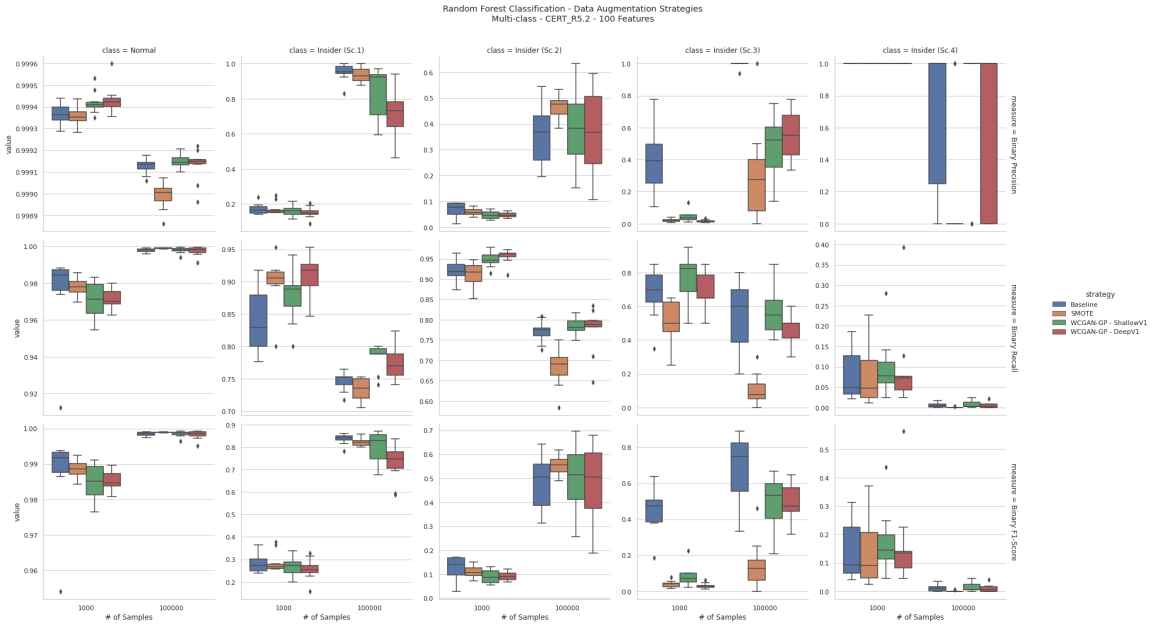


Figure A.8: CERT R5.2, 100 features: Weight-averaged classification metrics at 1000 and 100000 samples per class for all augmentation strategies

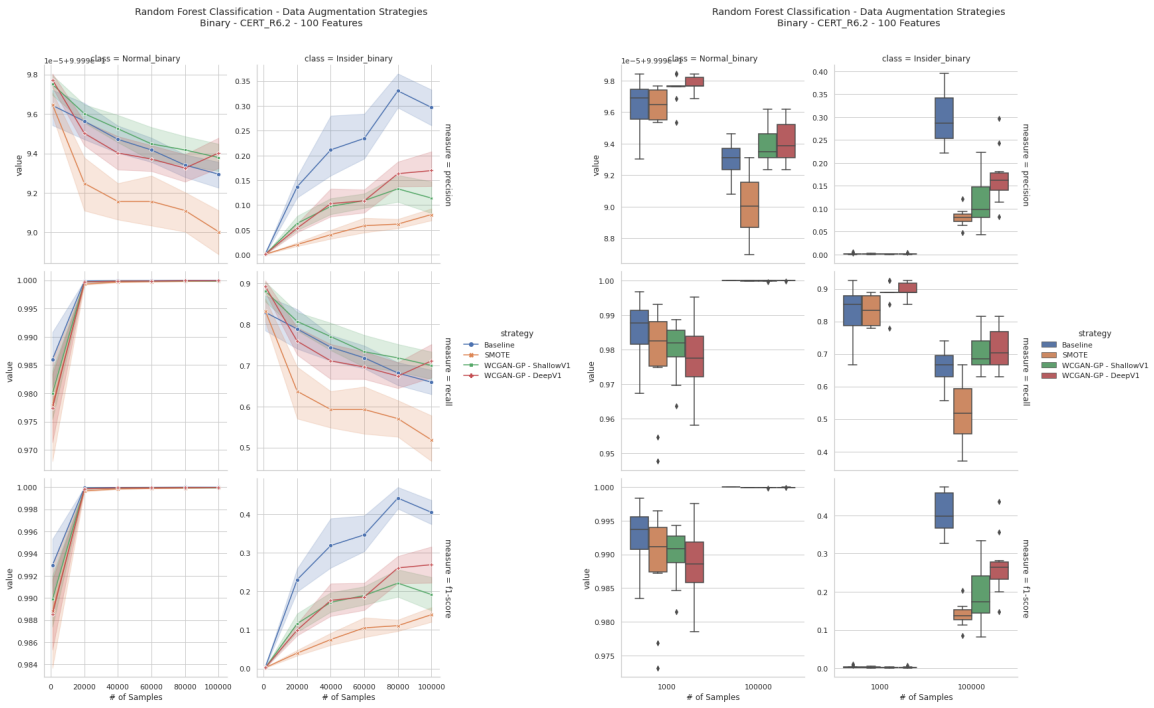


Figure A.9: CERT R6.2, 100 features: Binary weight-averaged classification plots for all augmentation strategies

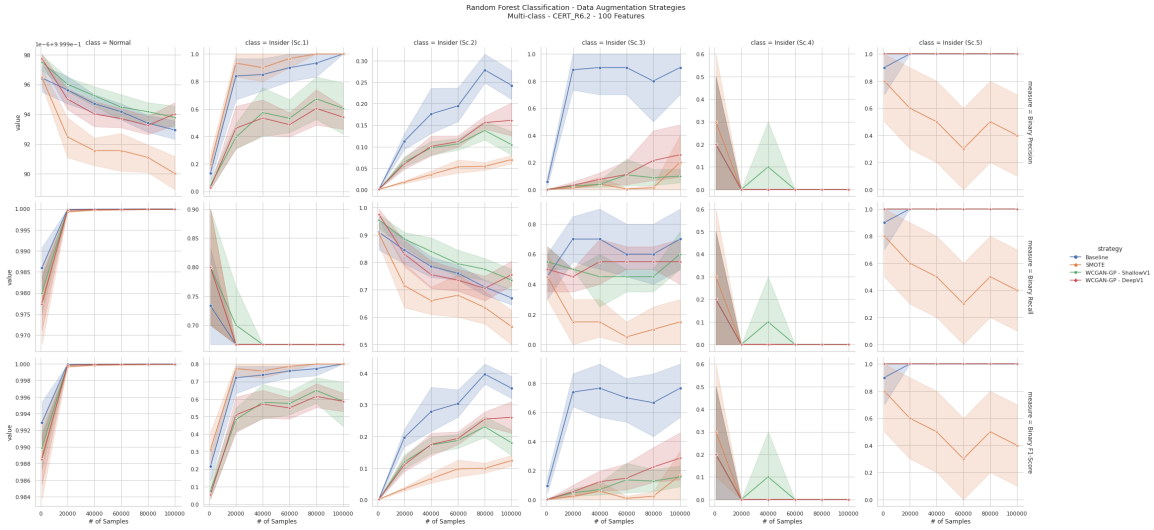


Figure A.10: CERT R6.2, 100 features: Weight-averaged classification metrics as n_samples increases for all augmentation strategies

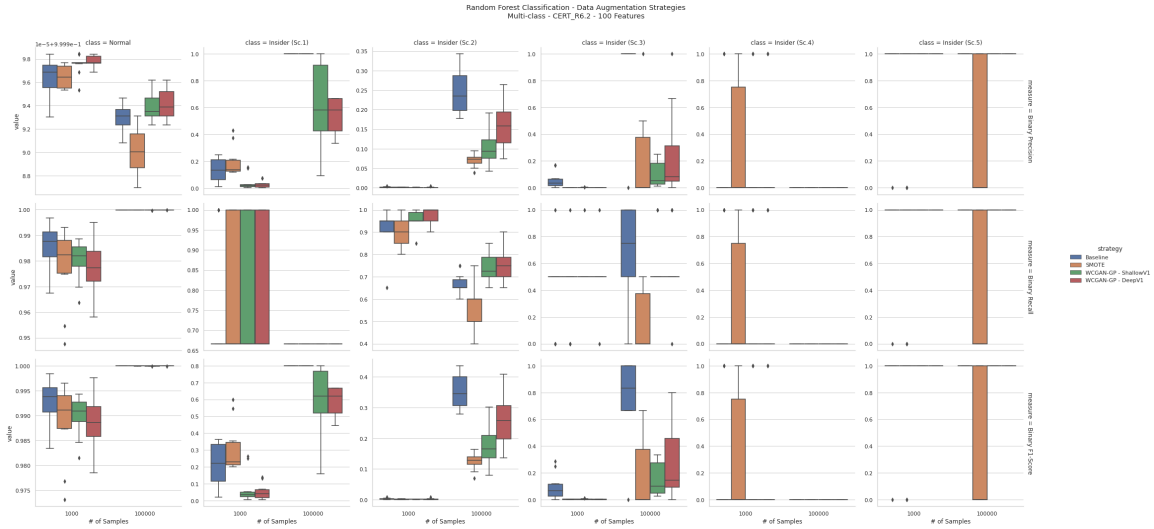


Figure A.11: CERT R6.2, 100 features: Weight-averaged classification metrics at 1000 and 100000 samples per class for all augmentation strategies

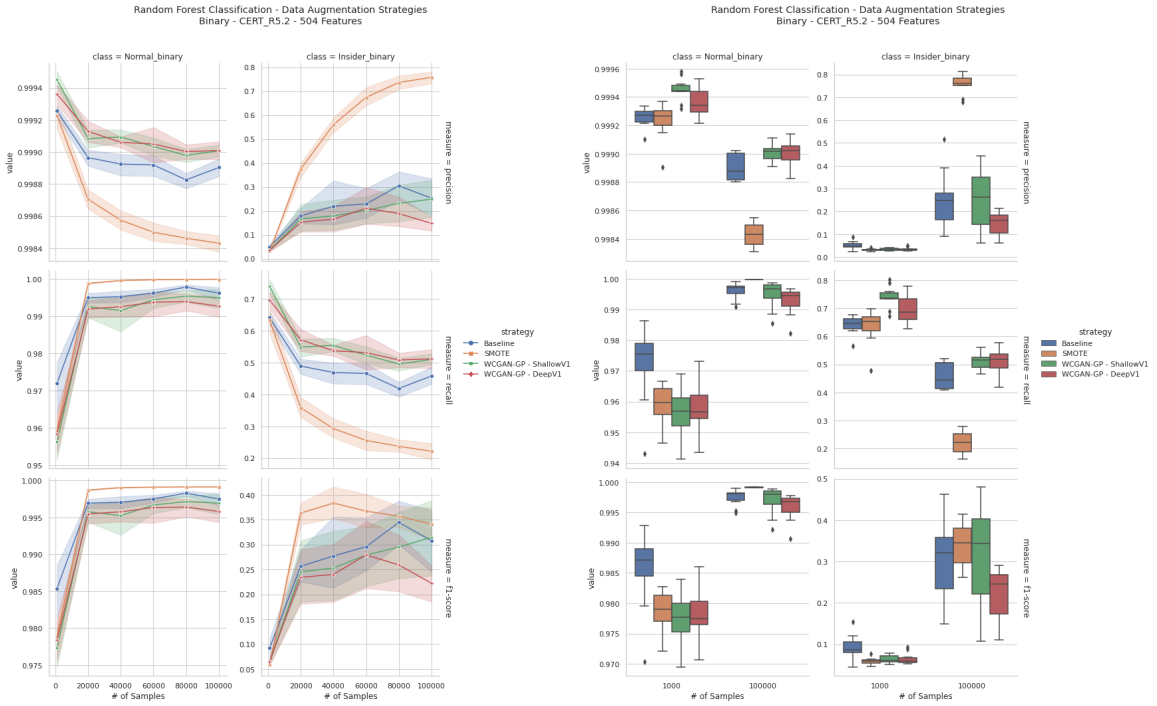


Figure A.12: CERT R5.2, 504 features: Binary weight-averaged classification plots for all augmentation strategies

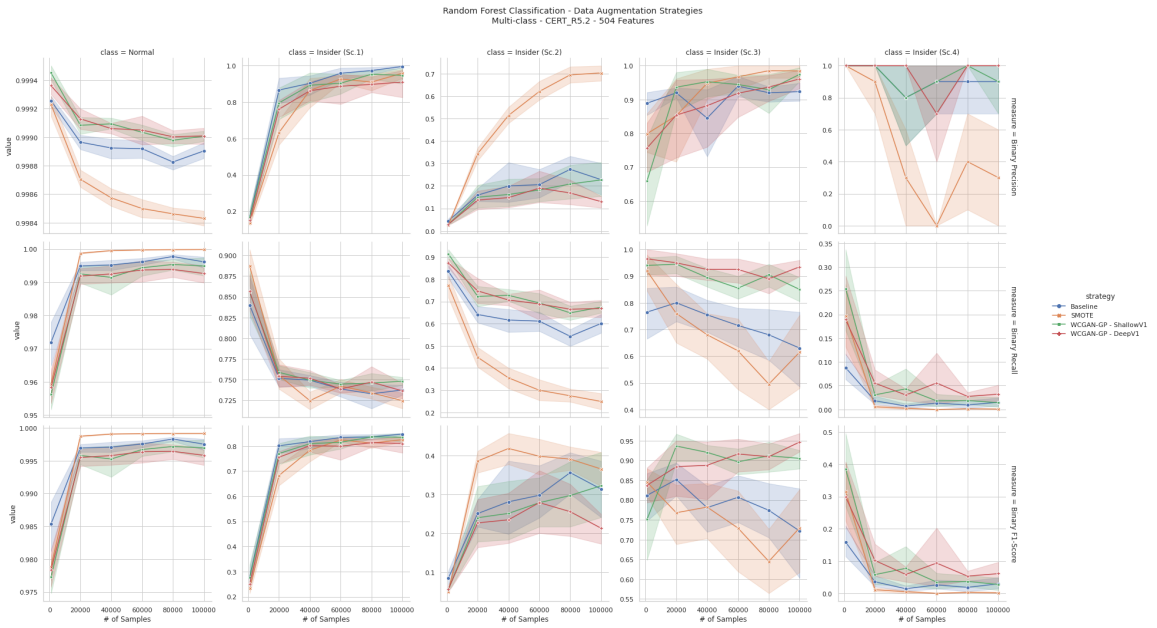


Figure A.13: CERT R5.2, 504 features: Weight-averaged classification metrics as n_samples increases for all augmentation strategies

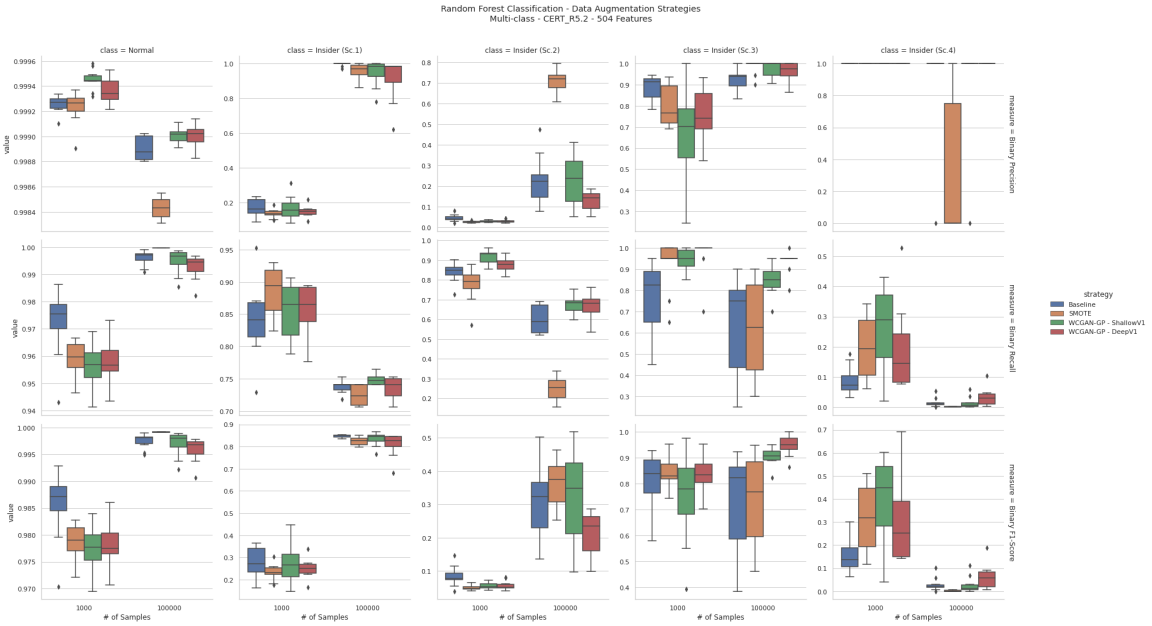


Figure A.14: CERT R5.2, 504 features: Weight-averaged classification metrics at 1000 and 100000 samples per class for all augmentation strategies

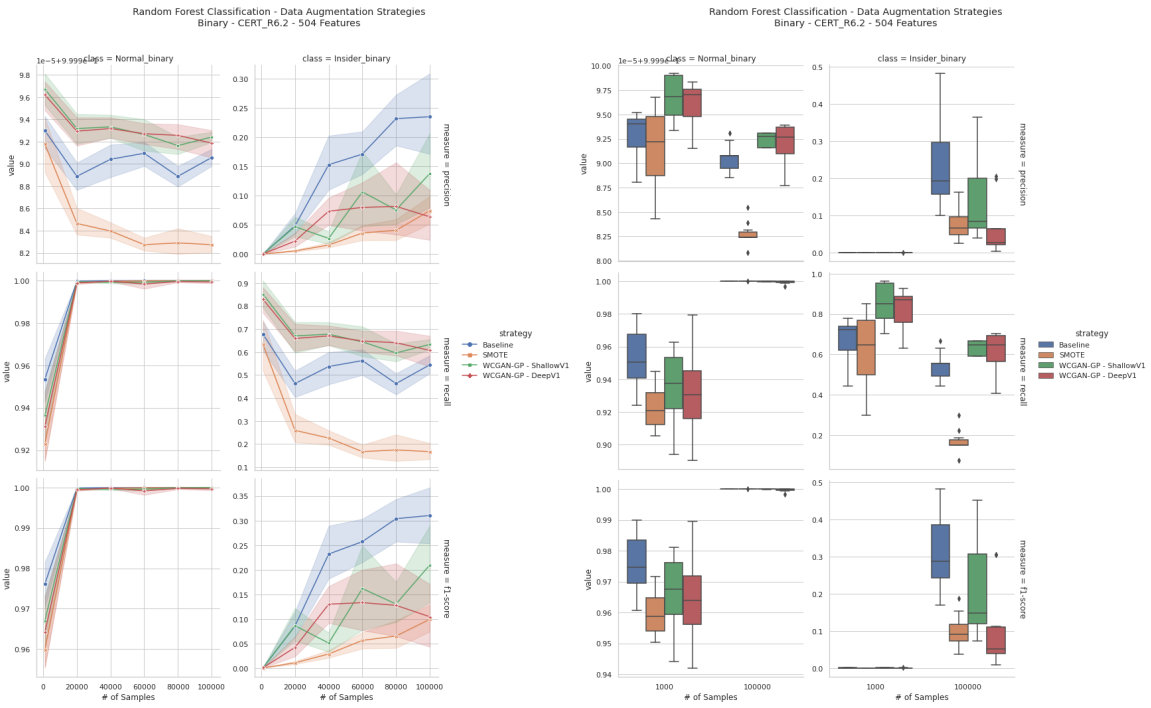


Figure A.15: CERT R6.2, 504 features: Binary weight-averaged classification plots for all augmentation strategies

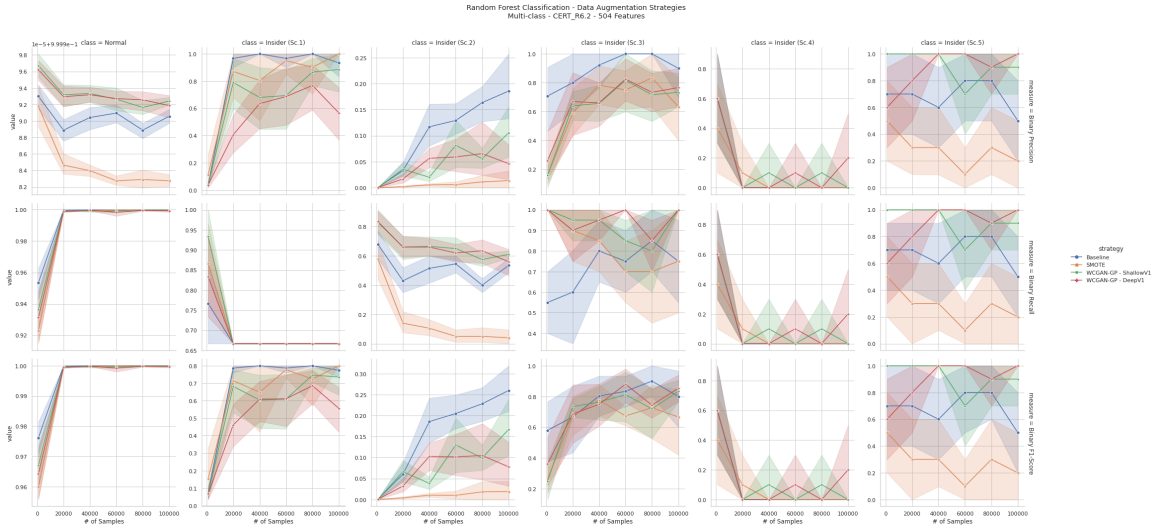


Figure A.16: CERT R6.2, 504 features: Weight-averaged classification metrics as n_samples increases for all augmentation strategies

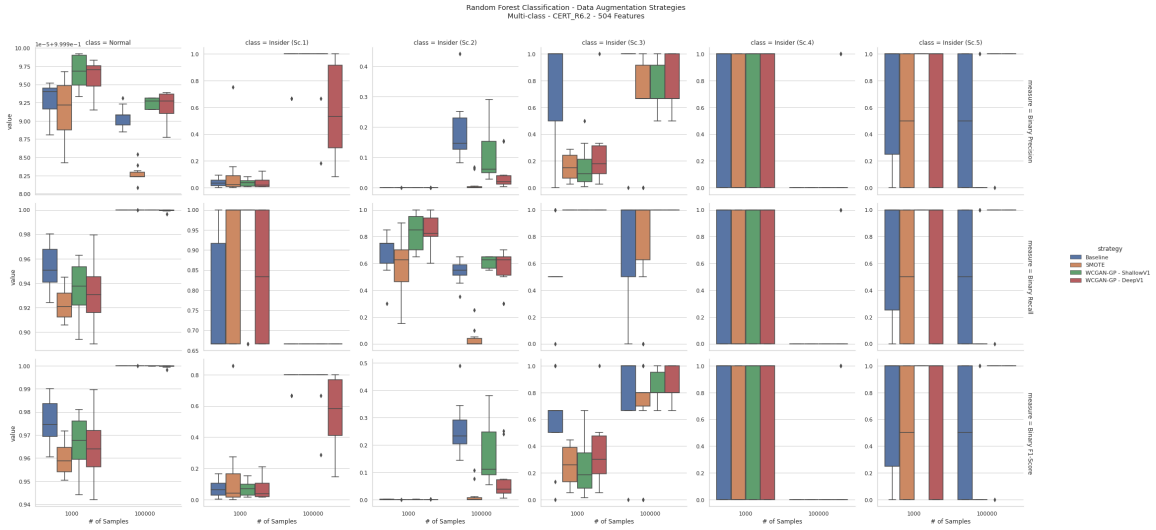


Figure A.17: CERT R6.2, 504 features: Weight-averaged classification metrics at 1000 and 100000 samples per class for all augmentation strategies