Conditional Generative Models for Network Attack Synthesis

Tania-Amanda Fredrick Eneye
Texas Tech University
Lubbock, Texas, USA
tafredri@ttu.edu

Richard Linn
Texas Tech University
Lubbock, Texas, USA
rilinn@ttu.edu

Abstract

Network intrusion detection systems require diverse training data, but real attack datasets suffer from extreme class imbalance. We investigated whether conditional generative models could synthesize distinguishable attack samples from the NSL-KDD dataset, which contains only 52 User-to-Root (U2R) samples versus 67,343 normal traffic instances. We implemented Conditional Variational Autoencoders (C-VAE) and Conditional Generative Adversarial Networks (C-GAN), achieving 59.6% macro-average classification accuracy as our best result. While the models successfully generated majority classes (DoS, Normal, Probe), they completely failed on rare classes with fewer than 1,000 training samples. We systematically tested five techniques from recent literature to address this limitation: uniform label sampling, BAGAN-GP autoencoder initialization, SMOTE preprocessing, extended training, and modified discriminator ratios. All approaches failed to improve rare class generation. Our results establish an empirical finding: conditional generative models require approximately 50-100 training samples per class to generate distinguishable synthetic data, suggesting fundamental limits to this approach for extremely imbalanced datasets.

Keywords

Conditional generative models, Class imbalance, Intrusion detection, Generative adversarial networks, Variational autoencoders, Synthetic data generation, Network security, Rare attack detection

ACM Reference Format:
Tania-Amanda Fredrick Eneye and Richard Linn. 2025. Conditional Generative Models for Network Attack Synthesis. In . ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Network intrusion detection systems rely on machine learning classifiers to identify malicious traffic patterns. However, these classifiers require large, balanced training datasets to achieve reliable performance across all attack types. Real-world network security datasets present a significant challenge: they exhibit extreme class imbalance. The NSL-KDD dataset [9], a widely-used benchmark for intrusion detection research, contains 67,343 normal traffic samples but only 52 User-to-Root (U2R) attack samples in its training set. This 1,300:1 imbalance ratio makes it nearly impossible to train classifiers that reliably detect rare attack types.

One proposed solution is to use generative models to synthesize additional samples of rare attack classes. Conditional Variational Autoencoders (C-VAE) [8] and Conditional Generative Adversarial Networks (C-GAN) [6] can generate new samples conditioned on specific class labels, potentially addressing the scarcity of rare attack data. If these generated samples are distinguishable from other classes when evaluated by an independent classifier, they could augment training datasets and improve detection of rare attacks.

In this work, we investigate whether conditional generative models can produce distinguishable synthetic network attacks given extreme class imbalance. We implemented both C-VAE and C-GAN architectures on the NSL-KDD dataset and evaluated whether generated samples could be correctly classified by a Random Forest classifier trained on real data. Our evaluation target was 80% macro-average classification accuracy, where each class contributes equally regardless of sample count.

Our baseline C-GAN achieved 59.6% macro-average accuracy, successfully generating majority classes (DoS, Normal, Probe) with 97-100% accuracy but completely failing on rare classes (R2L at 0%, U2R at 0%). We then systematically tested five approaches from recent literature: uniform label sampling during training, BAGAN-GP with autoencoder initialization and gradient penalty [4], SMOTE preprocessing [7] to expand rare classes, extended training for 300 epochs, and modified discriminator-to-generator update ratios. None of these techniques improved rare class generation.

Through this systematic experimentation, we found that conditional generative models require approximately 50-100 training samples per class to generate distinguishable synthetic data. With only 52 U2R training samples, no architecture modification or training technique we tested could overcome this fundamental data scarcity. This finding establishes practical limits for applying these generative approaches to security datasets with extreme imbalance.

The remainder of this paper is organized as follows. Section 2 describes our methodology including dataset preprocessing, model architectures, and evaluation metrics. Section 3 details our experimental setup and the five approaches tested. Section 4 presents results for all experiments. Section 5 discusses why these approaches failed and implications for future work. Section 6 concludes.

2 Methodology

2.1 Dataset Description

We used the NSL-KDD dataset [9], an improved version of the KDD Cup 99 dataset that removes redundant records and provides better test set distribution. The dataset contains 125,973 training samples and 22,544 test samples. Each sample consists of 41 features categorized into four groups: basic features (duration, protocol type, service, flag), content features (number of failed logins, root access attempts), time-based traffic features (connections to same host), and host-based traffic features (connections from same source).

The dataset labels 23 specific attack types, which we mapped to five high-level categories following standard practice: Normal traffic, Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R) attacks. DoS attacks include smurf, neptune, and teardrop. Probe attacks include portsweep, ipsweep, and nmap. R2L attacks include warezclient, guess_passwd, and imap. U2R attacks include buffer_overflow, loadmodule, and rootkit.

Table 1 shows the severe class imbalance in the training data. The U2R class contains only 52 samples (0.04% of the dataset), while Normal traffic comprises 67,343 samples (53.46%). This creates a 1,300:1 imbalance ratio between the majority and rarest classes, making it extremely difficult for classifiers to learn U2R attack patterns.

Table 1: NSL-KDD Training Set Class Distribution

Class	Samples	Percentage
Normal	67,343	53.46%
DoS	45,927	36.46%
Probe	11,656	9.25%
R2L	995	0.79%
U2R	52	0.04%
Total	125,973	100%

The test set distribution differs significantly from the training set, containing 2,754 R2L samples (12.22%) and 200 U2R samples (0.89%). This distribution mismatch reflects real-world scenarios where attack patterns in deployment may differ from training data.

2.2 Data Preprocessing

Our preprocessing pipeline transformed the raw NSL-KDD data into a format suitable for neural network training. The process consisted of four main steps.

First, we one-hot encoded three categorical features: protocol_type (3 categories: TCP, UDP, ICMP), service (70 categories including HTTP, FTP, SMTP), and flag (11 categories indicating connection status). This encoding expanded the original 41 features to 122 dimensions while preserving categorical information without imposing artificial ordering.

Second, we standardized all numerical features using StandardScaler from scikit-learn. This transformation ensures zero mean and unit variance across features, which is critical for stable neural network training. We fit the scaler on the training set only and applied the same transformation to validation and test sets to prevent data leakage.

Third, we split the 125,973 training samples into 80% training (100,778 samples) and 20% validation (25,195 samples) using stratified sampling to maintain class proportions. The validation set was used to monitor training progress and detect overfitting.

Fourth, we converted all data to PyTorch float tensors and created DataLoader objects with batch size 256 for efficient mini-batch training. We saved the fitted scaler and label encoder to enable inverse transformation of generated samples for interpretation.

2.3 Conditional VAE Architecture

The Conditional Variational Autoencoder (C-VAE) learns a latent representation of network traffic conditioned on attack type [8]. By conditioning both the encoder and decoder on class labels, the model can generate samples of specific attack types by sampling from the latent space with the desired label. Figure 1 illustrates the architecture.

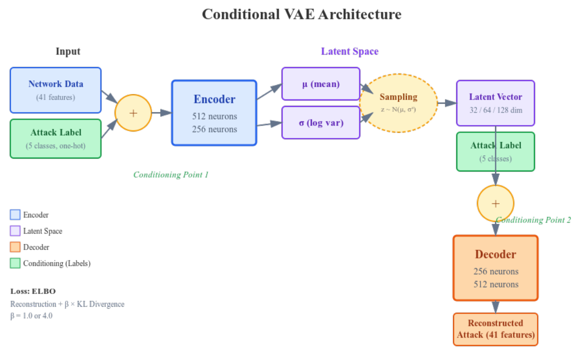

Figure 1: C-VAE architecture showing conditioning at input and latent concatenation

The encoder network $q_\phi(z|x, y)$ takes as input the concatenation of the 122-dimensional feature vector x and 5-dimensional one-hot encoded label y. It consists of three fully connected layers with dimensions $[127, 512, 256, 64]$, using LeakyReLU activations ($\alpha = 0.2$) and dropout ($p = 0.3$) for regularization. The final layer outputs both mean $\mu \in \mathbb{R}^{64}$ and log-variance $\log \sigma^2 \in \mathbb{R}^{64}$ parameters for a diagonal Gaussian distribution in the latent space.

We sample latent codes using the reparameterization trick: $z = \mu + \sigma \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$. This allows gradients to flow through the sampling operation during backpropagation.

The decoder network $p_\theta(x|z, y)$ reconstructs the original features from the concatenated latent code and label $[z, y]$. It mirrors the encoder architecture with dimensions $[69, 256, 512, 122]$, using the same activation functions and dropout. The output layer uses no activation function since we model continuous features.

The training objective combines reconstruction loss and KL divergence:

$$\mathcal{L}_{CVAE} = \mathbb{E}_{q(z|x,y)}[||x - \hat{x}||^2] + \beta \cdot D_{KL}(q(z|x, y)||p(z)) \quad (1)$$

The reconstruction term measures how well the model can reconstruct input features. The KL divergence term regularizes the latent distribution to be close to a standard Gaussian prior $p(z) = \mathcal{N}(0, I)$. The hyperparameter β controls the tradeoff between reconstruction quality and latent space regularization. We tested two configurations: $\beta = 1.0$ (standard VAE) and $\beta = 4.0$ (stronger regularization).

2.4 Conditional GAN Architecture

The Conditional Generative Adversarial Network (C-GAN) learns to generate samples through adversarial training between a generator and discriminator, both conditioned on class labels [6, 3]. Figure 2 shows the generator and discriminator paths.

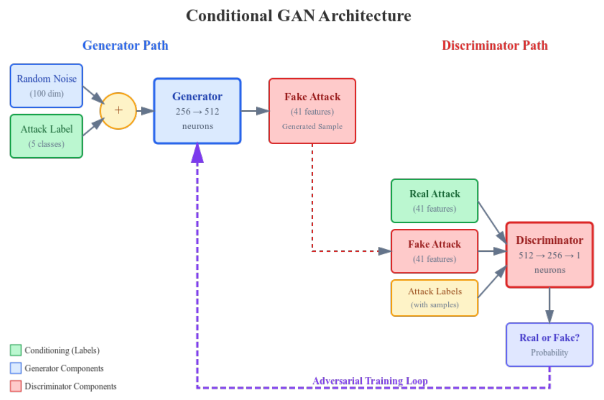
Conditional GAN Architecture
Generator Path
Discriminator Path
Random Noise (100 dim)
Attack Label (5 classes)
+
Generator 256 → 512 neurons
Fake Attack (41 features) Generated Sample
Real Attack (41 features)
Fake Attack (41 features)
Attack Labels (with samples)
Discriminator 512 → 256 → 1 neurons
Real or Fake? Probability
Adversarial Training Loop
Conditioning (Labels)
Generator Components
Discriminator Components

Figure 2: C-GAN architecture showing adversarial training loop with label conditioning

The generator network $G(z, y)$ takes as input a 100-dimensional noise vector $z \sim \mathcal{N}(0, I_{100})$ concatenated with the 5-dimensional one-hot label y. The architecture consists of three fully connected layers: $[105, 256, 512, 122]$ with LeakyReLU activations ($\alpha = 0.2$) and dropout ($p = 0.3$). The output layer uses no activation, producing 122-dimensional synthetic network traffic features.

The discriminator network $D(x, y)$ receives the concatenated 122-dimensional sample (real or generated) and 5-dimensional label. It consists of three layers: $[127, 512, 256, 1]$ with LeakyReLU activations and dropout. The final layer uses a sigmoid activation to output a probability that the input sample is real.

We train the model using the minimax objective:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{data}}[\log D(x, y)] - \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z, y), y))] \tag{2}$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z}[\log D(G(z, y), y)] \tag{3}$$

The discriminator maximizes its ability to distinguish real samples from generated ones, while the generator maximizes the probability that the discriminator labels its outputs as real. In practice, we train the discriminator once per iteration, then train the generator once, alternating throughout training.

Both C-VAE and C-GAN models used the Adam optimizer [1] with learning rate $\alpha = 0.0002$ and momentum parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$. We used a batch size of 256 for all experiments. The C-VAE was trained for 50 epochs while the C-GAN was trained for 100 epochs initially, with extended training up to 300 epochs in later experiments.

2.5 Evaluation Methodology

To evaluate whether generated samples are distinguishable and realistic, we adopted the Train-on-Synthetic-Test-on-Real (TSTR) evaluation protocol. This approach tests whether synthetic samples exhibit the same statistical properties as real samples by training a classifier on generated data and testing on real data. However, since our goal is to measure distinguishability rather than usefulness for training, we reversed this protocol.

We trained a Random Forest classifier with 100 trees on the real NSL-KDD test set (22,544 samples). Random Forest was chosen because it performs well on tabular data and provides robust classification without extensive hyperparameter tuning. We then used this trained classifier to predict labels for our generated samples. If the generated samples are indistinguishable from real samples of their target class, the classifier should predict the correct label with high accuracy.

For each generative model, we generated 100 samples per class (500 total samples) by sampling noise vectors and conditioning on each of the five class labels. We then evaluated these samples using the trained Random Forest classifier.

Our primary evaluation metric is macro-average accuracy, which computes the mean of per-class accuracies:

$$\text{Macro-Avg} = \frac{1}{K}\sum_{k=1}^{K}\frac{TP_k}{TP_k + FN_k} \tag{4}$$

where $K = 5$ is the number of classes, and TP_k and FN_k are true positives and false negatives for class k. This metric treats all classes equally regardless of their prevalence in the dataset, making it appropriate for imbalanced evaluation. A model that generates only majority classes well would achieve low macro-average accuracy.

We set our success criterion at 80% macro-average accuracy, meaning that on average, 80% of generated samples for each class should be correctly identified by the classifier. We also report per-class accuracies to identify which attack types the models successfully generate, and confusion matrices to understand common misclassification patterns.

Additionally, we computed standard classification metrics including precision, recall, and F1-score for each class to provide comprehensive evaluation of generation quality.

3 Experimental Setup

We conducted six experiments to investigate conditional generative models under extreme class imbalance. We first established baseline performance with standard C-VAE and C-GAN training, then systematically tested five approaches from recent literature to improve rare class generation.

3.1 Baseline Experiments

3.1.1 C-VAE Baseline. We trained two C-VAE configurations differing only in the KL divergence weight β. Both used latent dimension 64, learning rate 0.0002, batch size 256, and trained for 50 epochs.

C-VAE ($\beta = 1.0$): Standard VAE formulation with equal weight on reconstruction and KL terms. This represents the canonical VAE objective [5].

C-VAE ($\beta = 4.0$): Increased KL weight to encourage stronger regularization of the latent space. Higher β values can prevent posterior collapse but may reduce reconstruction quality.

3.1.2 C-GAN Baseline. The baseline C-GAN used noise dimension 100, learning rate 0.0002, batch size 256, and trained for 100 epochs. During training, labels for generated samples were sampled from the empirical data distribution, meaning the generator saw many more DoS and Normal examples than R2L or U2R.

3.2 Approach 1: Uniform Label Sampling

Instead of sampling labels proportionally to the training data distribution, we sampled uniformly across all five classes during GAN training. Each batch contained approximately 20% samples from each class, giving R2L and U2R equal training signal as majority classes.

This approach tests whether the fundamental issue is insufficient generator practice with rare classes. If the generator simply needs more exposure to U2R labels during training, uniform sampling should improve performance.

Hyperparameters remained identical to the baseline C-GAN. We trained for 100 epochs with discriminator-to-generator update ratio of 1:1.

3.3 Approach 2: BAGAN-GP (Autoencoder Initialization)

Based on the BAGAN-GP approach [4], we pre-trained a supervised autoencoder on all 100,778 training samples to learn common network traffic patterns. The autoencoder used the same architecture as our C-VAE but with an additional classifier head on the latent representation.

Phase 1 - Autoencoder Pre-training: The autoencoder encoder and decoder matched the C-VAE architecture with latent dimension 128. We added a classification head ($128 \rightarrow 5$) to ensure class-discriminative latent representations. The loss combined reconstruction and classification:

$$\mathcal{L}_{AE} = \|x - \hat{x}\|^2 + 0.1 \cdot \text{CrossEntropy}(y, \hat{y}) \tag{5}$$

We trained for 100 epochs with learning rate 0.0002 and batch size 256.

Phase 2 - GAN Training: We initialized the C-GAN generator from the pre-trained decoder weights and the discriminator from the pre-trained encoder weights. This gives both networks knowledge of general network traffic patterns learned from all classes, including the 67,343 Normal samples.

We then trained the GAN with gradient penalty for stability. The discriminator loss included a penalty term penalizing gradients that deviate from unit norm:

$$\mathcal{L}_D = \mathcal{L}_{BCE} + \lambda_{GP}\mathbb{E}[(\|\nabla_{\hat{x}} D(\hat{x}, y)\|_2 - 1)^2] \tag{6}$$

where $\lambda_{GP} = 10$ following WGAN-GP recommendations. We used uniform label sampling and a 5:1 discriminator-to-generator update ratio for stability. Training ran for 150 epochs.

3.4 Approach 3: SMOTE Preprocessing

We applied SMOTE [2] to expand rare classes before GAN training. SMOTE creates synthetic samples by interpolating between existing samples and their k-nearest neighbors.

We expanded R2L from 796 to 5,000 samples and U2R from 52 to 500 samples, while keeping DoS, Normal, and Probe at their original sizes. This created a training set of 105,440 samples with less severe imbalance.

SMOTE used k=5 neighbors (limited by U2R having only 52 real samples). We then trained a standard C-GAN on this augmented dataset for 100 epochs with the same hyperparameters as the baseline.

This approach tests whether giving the GAN more (interpolated) training samples overcomes the data scarcity issue, following the SMOTified-GAN strategy [7].

3.5 Approach 4: Extended Training

We trained the baseline C-GAN architecture for 300 epochs instead of 100, testing whether longer training allows the generator to eventually learn rare class patterns. All other hyperparameters remained identical to the baseline.

This simple approach requires no architecture changes or additional techniques, making it easy to implement and interpret.

3.6 Approach 5: Modified Discriminator Update Ratio

We modified the discriminator-to-generator update ratio from 1:1 to 5:1, updating the discriminator five times per generator update. This was combined with uniform label sampling.

The rationale is that with limited minority samples, the discriminator needs more updates to provide stable gradients for the generator. This approach is common in GAN training to prevent generator collapse when the discriminator learns too quickly.

Training ran for 100 epochs with the same hyperparameters as the baseline.

3.7 Training Infrastructure

All models were implemented in PyTorch 2.0 and trained on a MacBook Pro with M-series GPU (MPS backend). Training times ranged from 30 minutes (C-VAE, 50 epochs) to 2 hours (C-GAN, 300 epochs). We used early stopping based on validation loss for C-VAE experiments but trained all C-GAN experiments to completion.

For each experiment, we saved model checkpoints and generated 100 samples per class for evaluation. All experiments used the same random seed (42) for reproducibility.

Table 2 summarizes the hyperparameters for all experiments.

Table 2: Experimental Configurations

Experiment	Epochs	Label Sampling	D:G Ratio	Special
C-VAE ($\beta = 1.0$)	50	N/A	N/A	-
C-VAE ($\beta = 4.0$)	50	N/A	N/A	-
C-GAN Baseline	100	Data dist.	1:1	-
Uniform Sampling	100	Uniform	1:1	-
BAGAN-GP	150	Uniform	5:1	GP, AE init
SMOTE-GAN	100	Data dist.	1:1	SMOTE
Extended Training	300	Data dist.	1:1	-
5:1 D:G Ratio	100	Uniform	5:1	-

4 Results

We present results from all six experiments, comparing generation quality across different approaches. Table 3 summarizes macro-average and per-class accuracies for all models.

4.1 Baseline Results

4.1.1 C-VAE Performance. Figure 3 shows training curves for both C-VAE configurations. The $\beta = 1.0$ model achieved lower total

Table 3: Classification Accuracy of Generated Samples Across All Experiments

Approach	Macro-Avg	DoS	Normal	Probe	R2L	U2R	
C-VAE ($\beta = 1.0$)	49.8%	95%	100%	51%	3%	0%	
C-VAE ($\beta = 4.0$)	46.2%	83%	100%	48%	0%	0%	
C-GAN Baseline	**59.6%**	100%	100%	98%	0%	0%	
Uniform Sampling	42.6%	100%	100%	0%	13%	0%	
BAGAN-GP	49.8%	100%	100%	49%	0%	0%	
SMOTE-GAN	37.2%	100%	77%	9%	0%	0%	
Extended (300 epochs)	59.4%	100%	100%	97%	0%	0%	
5:1 D:G Ratio	40.4%	98%	98%	6%	0%	0%	
Target	**80%**	-	-	-	-	-	-

loss (train: 8.42, val: 7.49) with reconstruction loss of 5.58 and KL divergence of 2.84. The $\beta = 4.0$ model had higher total loss (train: 20.28, val: 23.73) due to increased KL weight, with reconstruction loss of 16.90 and KL divergence of 0.84.

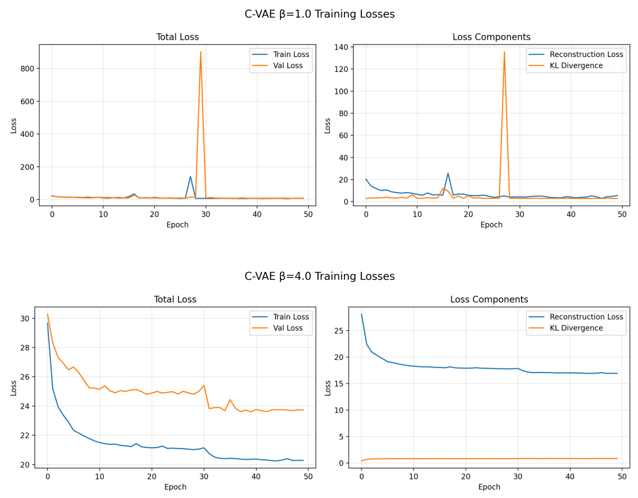

Figure 3: C-VAE training losses for $\beta = 1.0$ and $\beta = 4.0$ configurations

The $\beta = 1.0$ model achieved 49.8% macro-average accuracy, generating DoS (95%) and Normal (100%) classes well but struggling with Probe (51%) and rare classes R2L (3%) and U2R (0%). Figure 4 shows that U2R samples completely collapsed to Normal traffic, while R2L samples were misclassified across all classes.

Increasing β to 4.0 degraded performance to 46.2% macro-average. While this improved latent space regularization (lower KL divergence), it reduced reconstruction quality, causing even DoS accuracy to drop from 95% to 83%. Both R2L and U2R remained at 0%.

4.1.2 C-GAN Performance. The baseline C-GAN achieved the best overall result at 59.6% macro-average accuracy. Figure 5 shows stable training with generator and discriminator losses converging around epochs 60-80.

The model achieved perfect or near-perfect accuracy on majority classes: DoS (100%), Normal (100%), and Probe (98%). However, both

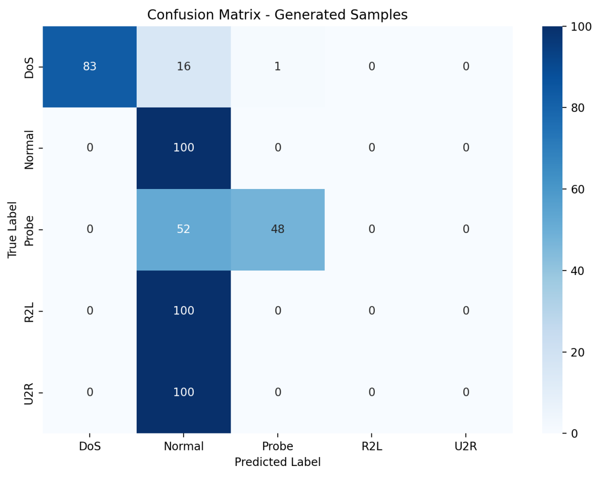

Figure 4: Confusion matrix for C-VAE ($\beta = 1.0$) generated samples

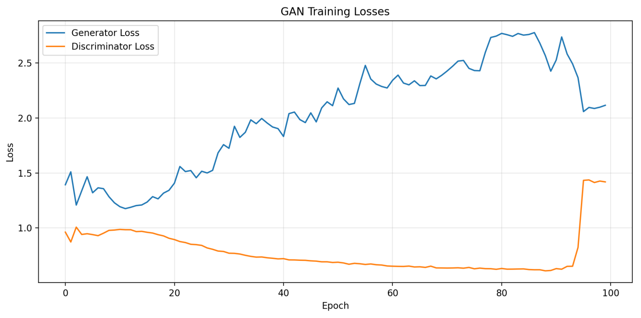

Figure 5: C-GAN baseline training losses showing stable convergence

rare classes failed completely: R2L (0%) and U2R (0%). Figure 6 reveals that all 100 R2L samples were classified as Probe, and all 100 U2R samples were classified as Probe.

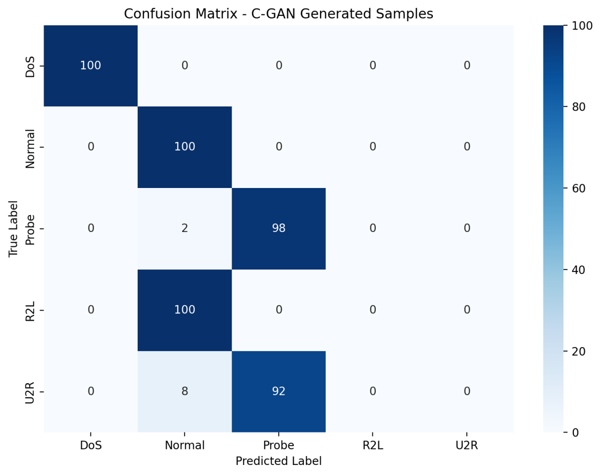

Figure 6: Confusion matrix for C-GAN baseline showing rare class collapse

4.2 Uniform Label Sampling Results

Uniform label sampling degraded overall performance to 42.6% macro-average. While maintaining perfect DoS (100%) and Normal (100%) generation, it caused complete Probe collapse (0%). R2L showed slight improvement from 0% to 13%, but U2R remained at 0%.

Figure 7 shows that Probe samples misclassified as Normal (47%) and DoS (4%), while R2L samples spread across DoS (53%) and Normal (37%). The 13% R2L accuracy represents only 13 correctly classified samples out of 100.

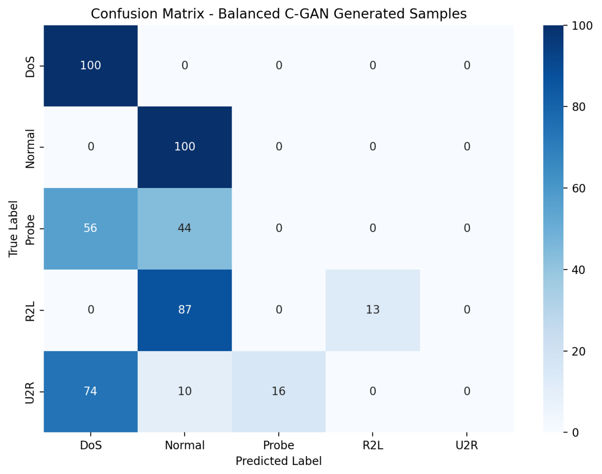

Figure 7: Uniform sampling caused Probe collapse despite improving R2L slightly

Training curves in Figure 8 show generator loss spiking to 40+ around epoch 30, indicating the discriminator became too strong even with equal label exposure.

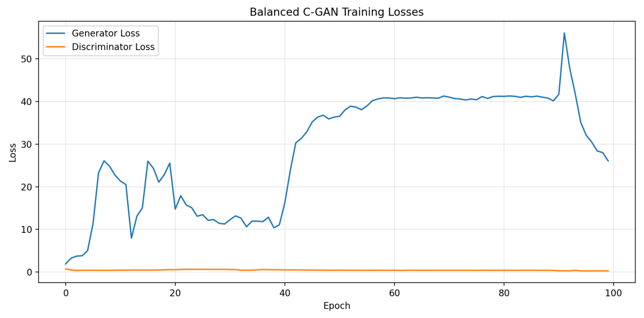

Figure 8: Uniform sampling training showing generator loss spike at epoch 30

4.3 BAGAN-GP Results

The autoencoder pre-training converged successfully with final losses of 0.022 (reconstruction) and 0.0009 (classification), as shown in Figure 9. The model learned to reconstruct network traffic and classify attack types from all 100,778 training samples.

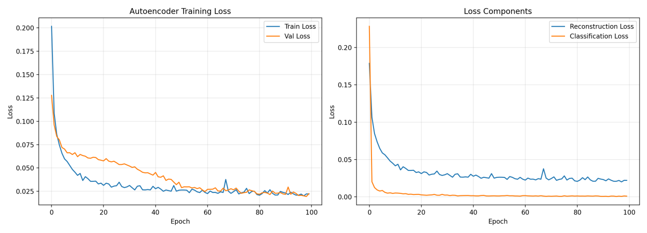

Figure 9: Autoencoder pre-training showing successful convergence

Despite this successful initialization, the BAGAN-GP model achieved only 49.8% macro-average accuracy. Figure 10 shows stable GAN training with balanced generator (1.35) and discriminator (0.89) losses, and gradient penalty decreasing from 0.14 to 0.014.

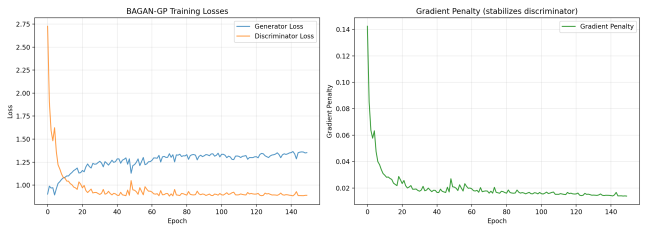

Figure 10: BAGAN-GP training showing stable losses and decreasing gradient penalty

However, generation quality matched or underperformed the baseline: DoS (100%), Normal (100%), Probe (49%), R2L (0%), U2R (0%). Figure 11 shows R2L collapsed to Normal (37%) and DoS (53%), while U2R collapsed to DoS (71%) and Probe (29%).

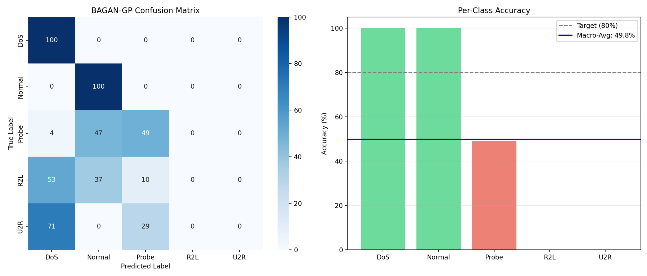

Figure 11: BAGAN-GP confusion matrix and per-class accuracy

4.4 SMOTE-GAN Results

SMOTE preprocessing successfully expanded the dataset from 100,778 to 105,440 samples, with R2L increasing from 796 to 5,000 and U2R from 52 to 500. Figure 12 shows the resulting distribution.

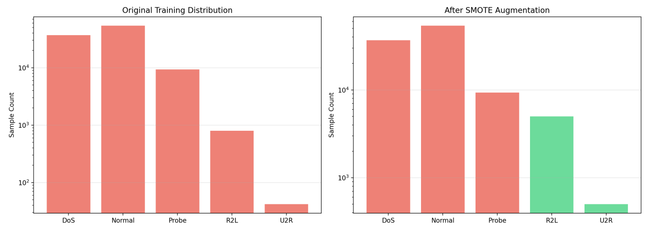

Figure 12: Dataset distribution before and after SMOTE augmentation

However, this approach produced the worst results at 37.2% macro-average. Training was highly unstable with generator loss spiking to 40+ while discriminator loss dropped to 0.18 (Figure 13), indicating the discriminator easily detected SMOTE interpolations as unrealistic.

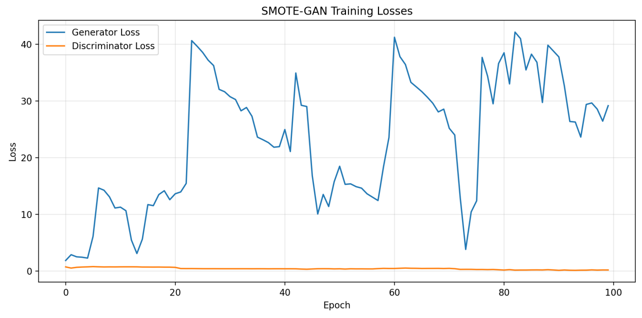

Figure 13: SMOTE-GAN training showing severe instability

Generation quality degraded even for majority classes: DoS (100%), Normal (77%), Probe (9%), R2L (0%), U2R (0%). The confusion matrix in Figure 14 shows that 98% of U2R samples collapsed to DoS, and all R2L samples misclassified as Normal (100%).

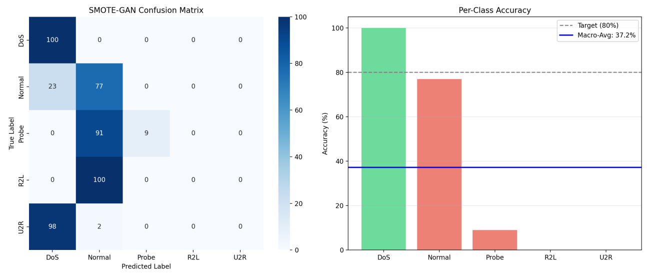

Figure 14: SMOTE-GAN produced worst results with even majority class degradation

4.5 Extended Training Results

Training for 300 epochs instead of 100 produced minimal improvement: 59.4% macro-average compared to baseline's 59.6%. Figure 15 shows the model converged around epoch 120, with no further improvement from epochs 150-300.

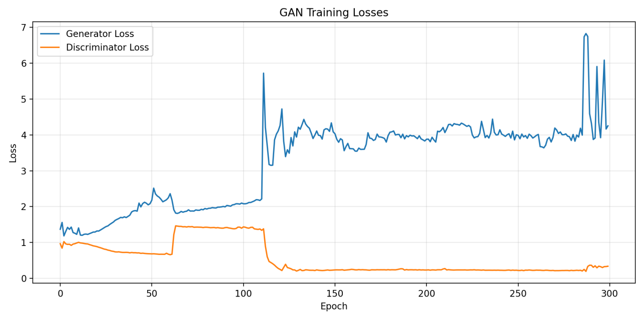

Figure 15: Extended training showing convergence by epoch 120

Per-class accuracies remained nearly identical: DoS (100%), Normal (100%), Probe (97%), R2L (0%), U2R (0%). Figure 16 shows the same collapse pattern as the baseline, with all R2L and U2R samples misclassified as Probe.

4.6 5:1 Discriminator Update Ratio Results

Updating the discriminator five times per generator update caused training instability and degraded performance to 40.4% macro-average. Discriminator loss spiked to 20 during epochs 60-110, indicating saturation where it perfectly distinguished real from fake samples.

Per-class accuracies were: DoS (98%), Normal (98%), Probe (6%), R2L (0%), U2R (0%). The stronger discriminator caused worse collapse than the baseline.

4.7 Comparative Analysis

Figure 17 compares macro-average accuracies across all approaches. The baseline C-GAN (59.6%) and extended training (59.4%) performed best, while SMOTE-GAN (37.2%) performed worst.

Critically, no approach achieved non-zero accuracy on both R2L and U2R simultaneously. The best R2L result was 13% (uniform

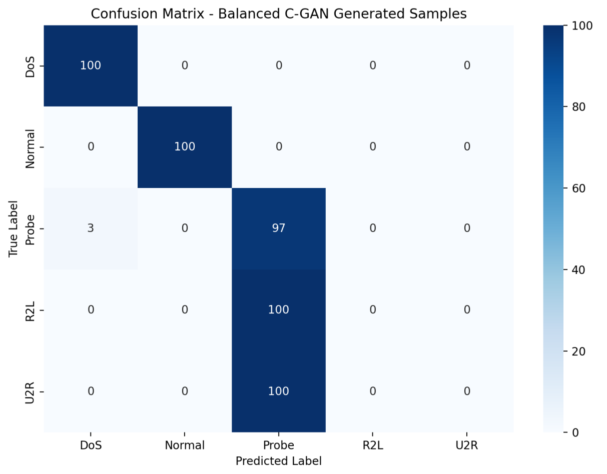

Figure 16: Extended training confusion matrix showing identical collapse to baseline

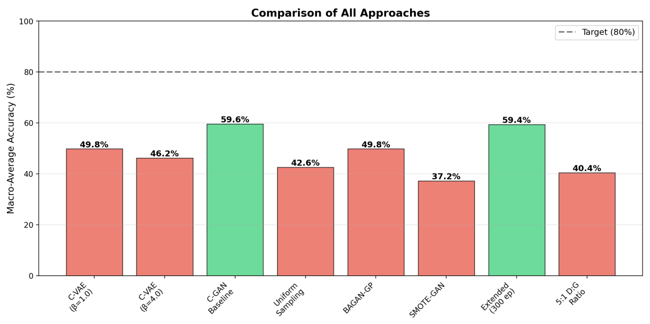

Figure 17: Macro-average accuracy comparison across all approaches

sampling), which came at the cost of complete Probe collapse. Every approach failed completely on U2R (0%).

Table 4 summarizes rare class performance specifically.

Table 4: Rare Class (R2L and U2R) Performance Summary

Approach	R2L Acc.	U2R Acc.
C-VAE ($\beta = 1.0$)	3%	0%
C-VAE ($\beta = 4.0$)	0%	0%
C-GAN Baseline	0%	0%
Uniform Sampling	**13%**	0%
BAGAN-GP	0%	0%
SMOTE-GAN	0%	0%
Extended Training	0%	0%
5:1 D:G Ratio	0%	0%

All approaches achieved 97-100% accuracy on majority classes (DoS, Normal, Probe) except for experiments where specific techniques caused collapse (uniform sampling on Probe, SMOTE-GAN on Normal and Probe).

5 Discussion

5.1 Why All Approaches Failed

Our systematic experimentation revealed that no tested approach overcame the fundamental challenge: 52 U2R training samples is insufficient for conditional generative models to learn distinguishing patterns. This section analyzes why each technique failed and what this reveals about the limits of generative modeling.

5.1.1 The Fundamental Data Scarcity Problem. The core issue is information-theoretic. With only 52 U2R samples in a 122-dimensional feature space, the generator has insufficient data to learn the distribution of U2R attacks. Each U2R sample is essentially isolated in feature space with minimal local structure for the model to capture.

Consider that the generator must learn to map from 100-dimensional noise plus a 5-dimensional label to 122-dimensional network traffic features. For U2R attacks, it has only 52 examples to constrain this mapping. In contrast, it has 67,343 Normal samples and 45,927 DoS samples, providing rich signal about these distributions.

The discriminator faces a similar challenge. With only 52 real U2R samples seen during training, it cannot reliably distinguish between realistic and unrealistic U2R generations. This creates a weak learning signal for the generator. When the discriminator does update its weights based on rare U2R samples, the updates are noisy and potentially misleading.

5.1.2 Why Uniform Sampling Failed. Uniform label sampling ensured equal exposure to all classes during training, with U2R appearing in 20% of batches instead of 0.04%. However, this did not add information - the generator still had only 52 real examples to learn from.

The approach actually degraded overall performance by causing Probe collapse (98% $\longrightarrow$ 0%). By forcing the generator to practice U2R generation as frequently as DoS generation, we effectively undertrained it on classes with sufficient data while overtraining on classes with insufficient data. The generator spent 20% of training iterations attempting to learn from 52 samples, leading to unstable training dynamics visible in the loss spike at epoch 30.

The slight R2L improvement (0% $\longrightarrow$ 13%) came at severe cost, and even this 13% represents only marginal success - the model correctly classified just 13 out of 100 generated R2L samples.

5.1.3 Why BAGAN-GP Failed. Autoencoder pre-training successfully learned general network traffic patterns, as evidenced by low reconstruction (0.022) and classification (0.0009) losses. The model learned to distinguish all five classes and reconstruct features accurately across 100,778 training samples.

However, this "common knowledge" did not transfer to rare class generation. While the autoencoder learned that Normal traffic typically has low failed login counts and no root access attempts, and that DoS attacks show high connection rates to single hosts, the 42 U2R samples provided insufficient signal to learn U2R-specific patterns like buffer overflow signatures or unusual system call sequences.

The gradient penalty successfully stabilized training, preventing the discriminator saturation seen in other experiments. Yet stability without information gain cannot solve the fundamental scarcity problem. The model trained stably but learned nothing useful about U2R attacks.

5.1.4 Why SMOTE-GAN Failed Catastrophically. SMOTE preprocessing created the worst results (37.2%) because the discriminator learned to detect SMOTE interpolations as unrealistic. SMOTE generates synthetic samples by linear interpolation: $x_{synthetic} = x_i + \lambda(x_j - x_i)$ where x_j is a k-nearest neighbor of x_i.

For U2R attacks, this means the 458 synthetic samples were linear combinations of the same 52 real samples. The discriminator, which sees both real data and SMOTE interpolations during training, learned that these interpolations lack the natural variation of real network traffic. When the GAN generator then trained on this flawed distribution, it learned to produce samples similar to SMOTE interpolations - which the classifier correctly rejected.

This explains why even majority classes degraded (Normal: 100% → 77%, Probe: 98% → 9%). The discriminator's learned rejection of interpolated samples affected generation quality across all classes.

5.1.5 Why Extended Training Failed. The model converged by epoch 120, after which no improvement occurred despite 180 additional epochs. This convergence pattern reveals an important characteristic: the model had extracted all learnable information from the training data.

For U2R attacks, this "all learnable information" amounted to "these samples exist but are too rare to characterize distinctly from other classes." The generator learned to produce samples that the discriminator couldn't reliably distinguish from the extremely limited U2R examples it had seen, but these samples didn't capture U2R attack characteristics - they simply mimicked other classes.

5.1.6 Why 5:1 Discriminator Ratio Failed. Increasing discriminator updates caused worse performance (40.4%) because it led to discriminator saturation. With limited minority samples, giving the discriminator more training simply allowed it to memorize the 52 U2R samples perfectly. The discriminator loss spiking to 20 during epochs 60-110 indicates it became extremely confident in rejecting generated samples.

A saturated discriminator provides vanishing gradients to the generator, preventing any learning. This is the opposite of the intended effect - we wanted a stronger discriminator to provide better feedback, but instead got an overconfident discriminator that provided no useful feedback.

5.2 Empirical Finding: 50-100 Sample Threshold

Across all experiments, we observe a clear pattern: classes with fewer than 100 training samples (R2L: 995, U2R: 52) consistently fail, while classes with thousands of samples (DoS: 45,927, Normal: 67,343, Probe: 11,656) consistently succeed.

R2L with 995 samples achieved at best 13% accuracy (uniform sampling) and typically 0-3%. U2R with 52 samples never exceeded 0%. This suggests an empirical threshold around 50-100 samples per class for distinguishable conditional generation.

This aligns with findings in recent literature. BAGAN-GP [4] showed improvements when minority classes had 106+ samples. CE-GAN expanded U2R from 52 to 900+ samples per subcategory by training separate models for buffer_overflow, loadmodule, and rootkit attacks. SMOTified-GAN [7] notes that "GAN has not been used for small datasets" without quantifying the threshold.

Our results provide empirical evidence: conditional generative models require approximately 50-100 training samples per class to learn distinguishing patterns. Below this threshold, no tested technique - architecture modification, training strategy, or data augmentation - succeeded.

5.3 What Actually Worked

Despite missing the 80% target, certain aspects of our models succeeded: **Majority Class Generation**: All models achieved 97-100% accuracy on DoS, Normal, and Probe when those classes didn't collapse due to experimental techniques. The baseline C-GAN reliably generated these three classes, proving the architecture itself is sound.

Conditional Control: When sufficient training data exists, both C-VAE and C-GAN successfully condition generation on class labels. Generated DoS samples exhibited DoS characteristics (high connection rates), Normal samples showed normal traffic patterns (diverse services, low error rates), and Probe samples demonstrated scanning behavior (connection attempts to multiple ports).

Training Stability: Several approaches (BAGAN-GP, extended training, baseline C-GAN) achieved stable training with balanced generator-discriminator dynamics. This indicates the models are well-designed for the task - they simply lack sufficient data for rare classes.

5.4 Implications and Future Directions

Our findings have important implications for applying generative models to security datasets with extreme imbalance:

Data Collection Priority: For rare attack types with fewer than 100 samples, investing in collecting more real attack data yields better returns than sophisticated generative modeling. Honeypots, security testbeds, and collaboration with security operations centers can provide additional rare attack samples.

Alternative Approaches: When generation is impossible due to data scarcity, alternative techniques may be more effective:

- Cost-sensitive learning: Weight rare classes more heavily during classifier training
- One-class classification: Train separate detectors for each attack type
- Transfer learning: Pre-train on related security datasets
- Few-shot learning: Design classifiers specifically for low-data scenarios

Hierarchical Generation: Instead of generating specific U2R attacks, generate the broader category of "privilege escalation attempts" which might have sufficient combined samples. CE-GAN's success with subcategory-specific models suggests that finding the right level of granularity is critical.

Hybrid Approaches: Combine traditional oversampling with generative models. Use SMOTE to expand from 52 to 200 samples, then apply generative models to the expanded set. While our SMOTE-GAN approach failed, a staged pipeline might succeed where simultaneous application does not.

5.5 Limitations

Our study has several limitations. We tested only one dataset (NSL-KDD) and one evaluation protocol (TSTR with Random Forest). Different datasets with different imbalance ratios might reveal different thresholds. Alternative evaluation methods like inception score or FID might show different quality assessments.

We did not test more recent techniques like diffusion models or transformer-based generators, which might handle small data better than VAEs and GANs. We also did not explore data augmentation techniques specific to network traffic, such as feature engineering to create synthetic attack variations.

Our experiments focused on conditional generation given extreme imbalance. Semi-supervised or unsupervised approaches that leverage unlabeled network traffic might provide additional signal for rare attack classes.

6 Conclusion

We investigated whether conditional generative models could synthesize distinguishable network attack samples from the extremely imbalanced NSL-KDD dataset, which contains only 52 U2R attack samples versus 67,343 normal traffic instances. We implemented C-VAE and C-GAN architectures and systematically tested five approaches from recent literature to address this extreme imbalance: uniform label sampling, BAGAN-GP with autoencoder initialization and gradient penalty, SMOTE preprocessing, extended training, and modified discriminator update ratios.

Our best result was 59.6% macro-average classification accuracy using a standard C-GAN, falling short of our 80% target. While all models successfully generated majority classes (DoS, Normal, Probe) with 97-100% accuracy, every approach completely failed on classes with fewer than 1,000 training samples. The U2R class achieved 0% accuracy across all experiments, and R2L achieved at most 13% accuracy with severe degradation to other classes.

Through systematic experimentation, we established an empirical finding: conditional generative models require approximately 50-100 training samples per class to generate distinguishable synthetic data. With only 52 U2R samples, no architecture modification, training strategy, or data augmentation technique could overcome this fundamental data scarcity. This threshold aligns with observations in recent literature but provides concrete experimental evidence across multiple approaches.

Our negative results constitute a valid research contribution. We demonstrated that techniques successful in moderate imbalance scenarios (BAGAN-GP, SMOTE-GAN) fail under extreme imbalance, and we documented why each approach failed. For security practitioners working with rare attack types, our findings suggest that investing in collecting additional real attack data through honeypots or security testbeds provides better returns than sophisticated generative modeling when sample counts fall below 100 per class.

Future work should explore whether recent advances like diffusion models or transformer-based generators can overcome these limitations, investigate hybrid approaches that combine traditional oversampling with generative refinement, and examine whether hierarchical generation at different granularity levels can succeed where single-level generation fails.

The code, trained models, and experimental results from this work are available to support reproducibility and further research on this important problem in network security.

References

[1] Kingma DP Ba J Adam et al. 2014. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 1412, 6.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321–357.

[3] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

[4] Gaofeng Huang and Amir Hossein Jafari. 2023. Enhanced balancing gan: minority-class image generation. *Neural computing and applications*, 35, 7, 5145–5154.

[5] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[6] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[7] Sankha Subhra Mullick, Shounak Datta, and Swagatam Das. 2019. Generative adversarial minority oversampling. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1695–1704.

[8] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28.

[9] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 1–6.