# UnVAEling Network Anomalies: Detecting Network Attacks with Variational Autoencoders

**McManigal, Connor** mcmanigc@uci.edu
**Mui, Aaron** armuil@uci.edu
**Politewicz, Peyton** peyton.p@uci.edu
**Verma, Rishabh** rishabv1@uci.edu

## Abstract

We investigate the performance of two different variational autoencoder (VAE) approaches to the ever-expanding space of utilizating machine learning for network traffic monitoring and safety. We train a baseline VAE, and then compare its performance to a Mixed-loss VAE (MLVAE), testing if its loss structure of separating categorical and continuous features offer improved performance in this use case. We chose VAEs because the learned latent structure of the data is particularly useful when labels are sparse and difficult to obtain, which is often the case for network attacks. For the same reason, they often generalize well on unseen data and are scalable. We train these VAEs on normal, expected traffic, and then leverage the variety of attack data to validate and test our filtration approach. We utilize the RT-IoT2022 dataset, sourced from the UC Irvine Machine Learning Repository, whose contents are intended to simulate local area network communication between internet-connected 'smart' devices, packaged with simulated attack traffic. Compared to other related works, we add a second layer to classify specific varieties of network traffic, instead of detecting anomalies and sorting data into 'attack'/'normal' traffic patterns by reconstruction error.

## 1 Introduction

With technology evolving at breakneck speeds, network attacks will become increasingly sophisticated and prevalent. Pair this with the widespread adoption of internet-of-things devices into consumer homes, and it's easy to intuit how cybersecurity continues to serve as the frontier of IT practice. We can trace stark reminders of this back to 2013, where Graig Heffner presented shockingly easily-exploited vulnerabilities in the very hardware deployed to keep supposedly-secret government labs and manufacturing centers safe. (1)

With this problem under continuous assessment, there is a wealth of contemporary academic literature to understand its trajectory. There are a number of baseline datasets commonly employed: yahoo, CIDDS-001, UNSW-NB15, and ours, RT-IoT2022. Quick perusals of Kaggle or Hugging Face demonstrate a plethora of projects that seek to classify these datasets, with many settling on random forests and gradient boosting. These produce reasonable results at 94 percent accuracy (4), and tests of both standard and variational autoencoders boasts even greater success. (3) Fittingly, VAEs have higher performance peaks, but are more fragile in their implementation than standard AEs. (3). Under such heavy scrutiny, the applicability of machine learning for traffic filtering has niches focused on various individual metrics; in our case, we sourced this data and our project's genesis from a paper primarily concerned with resource utilization. AEs are computationally costly to to train; they compared quantized AEs to a standard AE baseline, and found a minor performance loss of approximately 2 percent accuracy, while analyzing data twice as quickly with less than a quarter of the memory and CPU utilization. (2).

This quick perusal of the state-of-the-art reveals that high accuracy with AE solutions in this space is taken as a given; truly, it's already looked at something that might be worth compromising for

Preprint. Under review.

flexibility in other areas, so where does our project enter the fray? Across our reviewed literature, the AE filtering method began and ended with binary classification; is a traffic record normal, allowable traffic, or is it attack data? If it's the latter, toss it out. Our aim here is to iterate and see how multinomial classification fares, especially considering practical aims for cybersecurity. While the 'exclusive filter' methodology might be highly effective for maintaining a clear network, given the amount of data generated by telecommunication, there's treasure on the cutting room floor: why not set the stage for the discovery of new attacks by working at higher resolution?

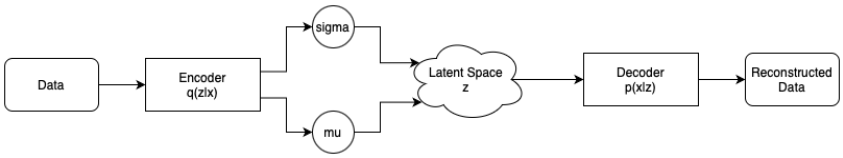


Figure 1: Variational Auto Encoder Architecture

## 2 Methodology

We approach this project by training a VAE and an MLVAE with normal network traffic, then implementing validation and test steps that fold in the attack traffic to ensure our models' reconstruction error specificity satisfactorily separates data at high accuracy. We leverage Bayesian optimization to tune VAE hyperparameters as well as the classification threshold theta. The following subsections describe our machine learning pipeline and present the details of each modeling stage.

### 2.1 Data Pipeline

The dataset consists of approximately 123,000 rows corresponding to network traces. Basic categorical information is recorded for each trace such as source port, destination port, protocol (e.g. TCP, UDP), and service (e.g. HTTP, FTP). The dataset includes approximately 80 continuous variables that characterizes each trace's associated set of packets through aggregate statistics. Characterizing variables include packet header size, packet payload size, packet inter-arrival time, and packet flag. Aggregate statistics include functions like minimum, maximum, count, average, and standard deviation. The target value column contains 13 categorical normal and attack types: 4 normal patterns and 9 attack patterns. Within the normal traces, there are Wipro Bulb, MQTT, Thing Speak, and Amazon Alexa. Attack patterns include DOS SYN Hping, ARP Poisoning, NMAP UDP Scan, NMAP XMAS Tree Scan, NMAP OS Detection, NMAP TCP Scan, DDOS Slowloris, Metasploit Brute Force SSH, and NMAP Fin Scan.

To ensure satisfactory training of the VAEs, we parceled out our data as such, with each subset stratified as evenly as possible across the variety of traffic pattern categories:

- **Training**: 70 percent of normal traffic data
- **Validation**: 15 percent of normal traffic data, and 50 percent of attack traffic data
- **Test**: 15 percent of normal traffic data, and 50 percent of attack traffic data

### 2.2 Mathematical Formulas

In our two models, we use two different loss functions. Below are the equations for the ELBO and loss functions.

**General ELBO Loss Formulas:**

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))]\end{aligned} \tag{1}$$

$$\mathcal{D}_{\text{KL}} = \frac{1}{2}\cdot\sum_{i=1}^{D}\left(1 + 2\cdot\log(\sigma_i) - \mu_i^2 - \sigma_i^2\right) \tag{2}$$

Because $p_\theta(\mathrm{z}) = N(\mathrm{z}; 0, \mathrm{I})$ and approximate posterior are both Normal Gaussians, we can solve the KL-Divergence in closed form. This is because Gaussian probability density functions are relatively simple and analytically tractable. Gaussian distributions are fully described by their mean and variance (sufficient statistic), which also helps in calculating the KL-Divergence because the calculations are based on log-likelihoods.

**Baseline VAE Formulas:**

$$\mathcal{L}_{\text{ELBO}} = \mathcal{L}_{\text{ReconMSE}} - \mathcal{D}_{\text{KL}} \tag{3}$$

$$\mathcal{L}_{\text{ReconMSE}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{d=1}^{D_n} (x_{n,d} - \hat{x}_{n,d})^2 \tag{4}$$

For our VAE loss, we use MSE, following the general PyTorch implementation. With our true data and reconstructed data both being of size Batch Size $\times$ Features. We sum over the features and then take the average over the batch to get a reconstruction loss.

**Mixed-Loss VAE (MLVAE) Formulas:**

$$\mathcal{L}_{\text{ELBO}} = \mathcal{L}_{\text{Recon}} - \mathcal{D}_{\text{KL}} \tag{5}$$

$$\mathcal{L}_{\text{Recon}} = \mathcal{L}_{\text{SmoothL1}} + \mathcal{L}_{\text{BCELogits}} \tag{6}$$

$$\mathcal{L}_{\text{SmoothL1}} = \begin{cases} \frac{0.5(x_n - y_n)^2}{\beta}, & \text{if } |x_n - y_n < \beta \\ |x_n - y_n| - 0.5\beta, & \text{otherwise} \end{cases} \tag{7}$$

$$\mathcal{L}_{\text{BCELogits}} = [y_n \cdot \log\sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))] \tag{8}$$

For the mixed-loss we split the features into two separate groups and calculate two reconstruction errors. The Smooth L1 loss is used for continuous features. This loss, which combines the MSE and MAE, is more robust to outliers in training, ensuring that our model is will not be tricked by large variations within the normal data. In our implementation we use $\beta = 1.0$. The binary cross entropy with logits is used with the categorical features. This combines the sigmoid function and binary cross entropy (BCE), first by turning the logits into probabilities then computing the BCE. Doing this in one step provides numeric stability and more efficient computation, leading to better performance in classification tasks. This extra loss component helps the MLVAE deal with binary categorical features better, because the baseline utilizes MSE for all features.

### 2.3 VAE and MLVAE Training

Both of these models were trained to minimize reconstruction error across the 4 variants of normal traffic data. The exclusion of any attack data at this step is purposeful: these filtering approaches rely upon AEs that are only trained on the data they should permit to pass through; anomalous data naturally bears a much higher reconstruction error, on average, so baseline accuracy numbers are expected to be quite high. It is peculiar, and was novel to work through, given that we are not actually trying to define the 'anomaly' - instead, we simply seek to filter out any signal that is appreciably different from the accepted standard. This has the added benefit of being intuitively robust against many kinds of attacks, and even novel, new attacks, so long as they don't sufficiently masquerade as normal traffic. As shown in Figure 2, ELBO training loss converged rather quickly; baseline VAEs took a couple more epochs to converge than the MLVAEs. This speedy convergence can be attributed to the relatively consistent nature of the normal trace patterns used to train.

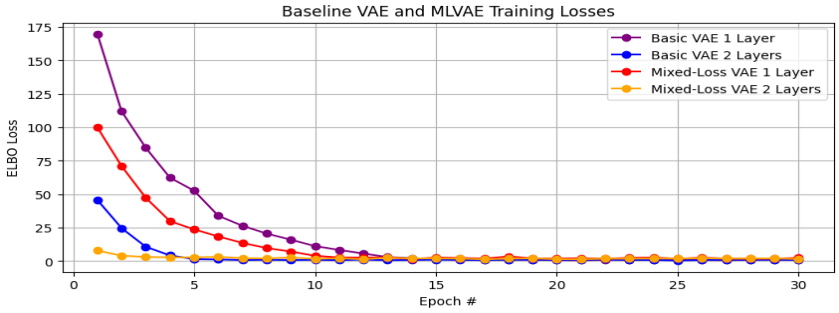


Figure 2: ELBO loss of baseline and mixed-loss VAEs over training epochs.

### 2.4 One Anomaly at a Time Validation

There is a bit of a baton pass from the training step to the validation and testing steps in this process. The remaining data will be run through the VAEs produced by the training step, and our objective from here is to define reconstruction loss thresholds that maximize separation between normal and attack traffic samples.

We employ one-left-out classification instead of traditional multinomial methods because we do not want to have to consider a softmax scenario where we are evaluating a sample's fit across all classes. Instead, we utilize Bayesian optimization to check error thresholds across a series of nine binary classification models, where we test the VAEs' ability to reconstruct one category of signal per model. We optimize such that the accuracy of the models for normal traffic categories is maximized, and the accuracy of the models for attack traffic are minimized. From here, we extract one value by taking the median of all threshold values, ensuring that at least half our data classes achieve similar one-left-out AUC, and utilize that for the testing step.

| | One Layer VAE | One Layer MLVAE | Two Layer VAE | Two Layer MLVAE |
|---|---|---|---|---|
| Training Error | 1.008 | 2.382 | 0.923 | 1.597 |
| Valid. AUC | 0.835 | 0.778 | 0.814 | 0.844 |
| $H_0$ | 173 | 361 | 279 | 396 |
| $Z_0$ | 27 | 21 | 21 | 5 |
| $H_1$ | NA | NA | 343 | 323 |

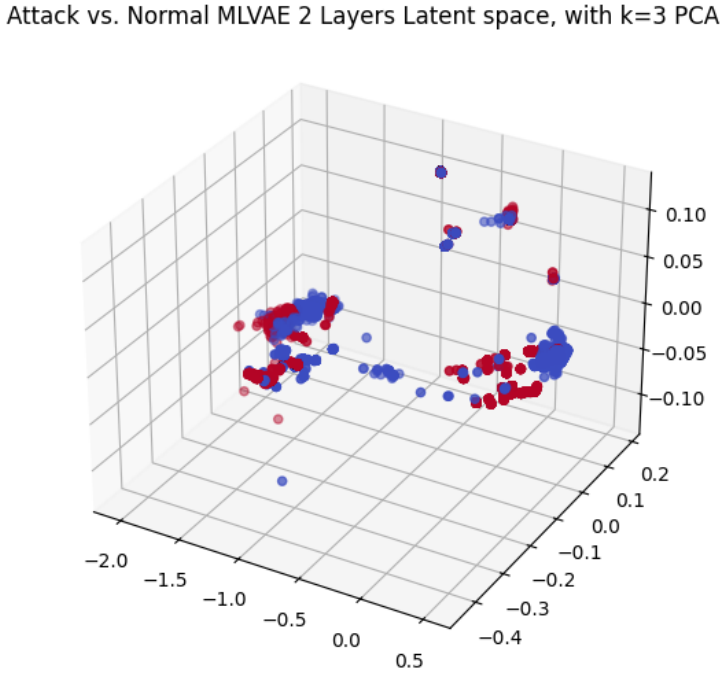


Figure 3: Latent space visualization (Red is anomalous, Blue is normal).

## 3 Results

After the VAE hyperparameters are tuned, we test each of the four VAEs on the remaining held out data to generate a practical assessment of filtering accuracy. Our results provide valuable insight into the comparison between and efficacy of VAEs and MLVAEs for identifying specific types of attack traffic.

| Model | DOS | ARP | UDP | XMAS | OS | TCP | DDOS | SSH | FIN |
|---|---|---|---|---|---|---|---|---|---|
| VAE1 | 0.662 | 0.923 | 0.821 | 0.740 | 0.458 | 0.848 | 0.921 | 0.939 | 0.480 |
| VAE2 | 0.694 | 0.922 | 0.806 | 0.767 | 0.534 | 0.847 | 0.913 | 0.931 | 0.519 |
| MLVAE1 | 0.647 | 0.669 | 0.917 | 0.999 | 0.500 | 0.777 | 0.896 | 0.934 | 0.439 |
| MLVAE2 | 0.705 | 0.766 | 0.926 | 0.999 | 0.577 | 0.819 | 0.919 | 0.958 | 0.614 |

We rate model success primarily via their mean AUC, and from these results we find a mean of 0.7534 in the worst case with MLVAE1 (with one layer), and a mean of 0.8092 in the best case with MLVAE2 (two layers). While these accuracy figures are lower than others reported on this dataset in similar studies, we note that these are size-agnostic and significantly differentiable by virtue of their multinomial aim, contrasted with the usual binary classification methodology.

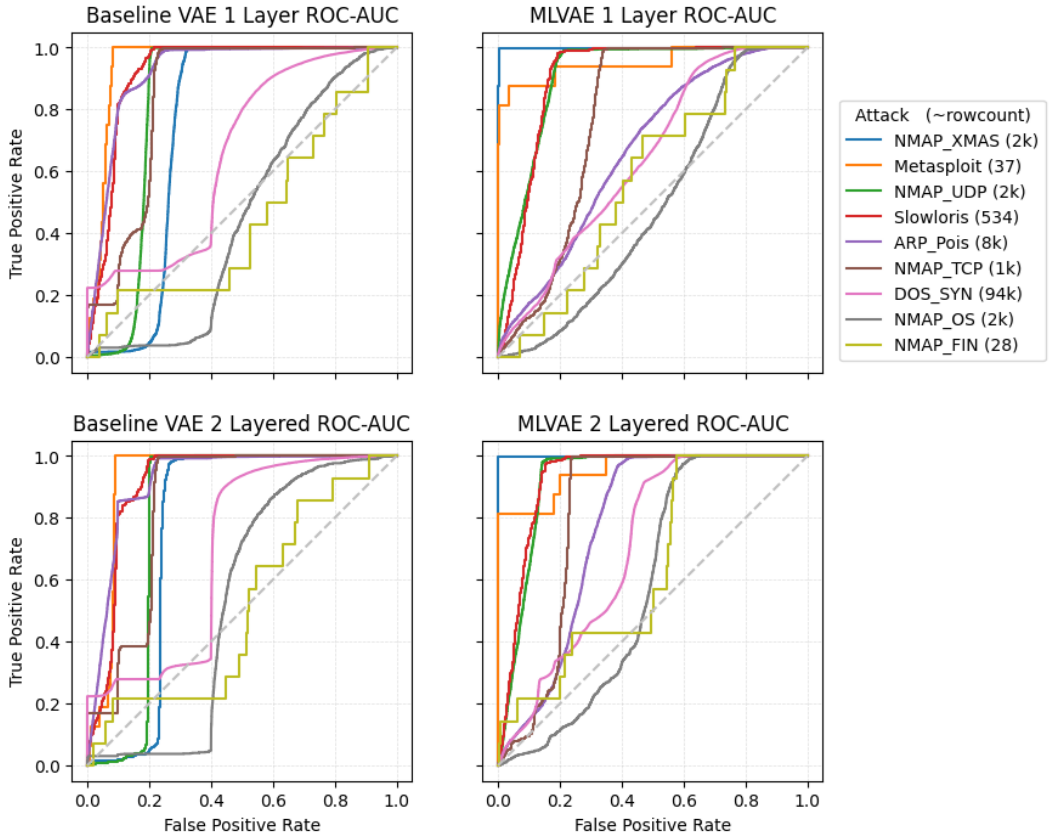


Figure 4: ROC plots for each model. Legend is annotated with corresponding observation count in dataset.

## 4 Discussion

Our results demonstrate that there is some clear performance tradeoff between VAEs and MLVAEs. MLVAEs seem to improve more quickly than VAEs with the inclusion of additional hidden layers. The attack types themselves can be sorted into a number of subcategories depending on their responsiveness to different model approaches:

- **Slowloris DDOS, Brute Force SSH**: Classification accuracy for these is consistent across all four models.

- **DOS HPing**: Classification accuracy benefits distinctly from the inclusion of an additional hidden layer.
- **ARP Poisoning, TCP Scan**: VAE outperforms MLVAE, but MLVAE appears to be closing in with the inclusion of an additional hidden layer.
- **UDP Scan**: VAE outperforms MLVAE, and there is no improvement for any model from an additional hidden layer.
- **XMAS TREE Scan**: VAE outperforms MLVAE, and there is no improvement for any model from an additional hidden layer.
- **OS Detection**: MLVAE outperforms VAE, and there is similar improvement across both with an additional hidden layer.
- **FIN Scan**: VAE outperforms MLVAE with one hidden layer. With an additional hidden layer, MLVAE leapfrogs with a drastic performance increase.

MLVAE models demonstrate an interesting relationship with classification utilizing categorical variables. Consider the various scan attacks: TCP, FIN, UDP and XMAS TREE, whose properties center around tricking networks with into responding and sending valuable, decryptable packets that reveal ports open for more intrusive attack trajectories. The MLVAE absolutely clears any XMAS TREE attack and outperforms on UDP scan detection, while seeing notable improvements in performance from an additional hidden layer for TCP and FIN scans.

This highlights an interesting, niche application for this model in particular: parsing these stealthy, seemingly-innocuous sniffing attack vectors that are more often used to explore and determine vulnerabilities. This avenue of further exploration into 'sniffing' detection is supported by MLVAE's performance with OS Detection attacks as well.

One potential area for expansion could be implementing a supervised VAE where the target labels are incorporated in the training process. This approach may be able to better learn representations of the data with a loss function that would include reconstruction error, KL divergence, and an added classification loss, such as cross entropy loss. Another possible expansion could include training a Generative Adversarial Network (GAN), where a generator is trained to produce synthetic network trace data and a discriminator that uses reconstruction error to evaluate whether the data is real or not. A few other ideas for improvement include penalizing false negatives by modifying the loss equations to include weights for these errors, or using a weighted F-score that prioritizes high recall.

## 5 Implementation Notes

Model training and evaluation is implemented using PyTorch. All exploratory data analysis and data visualization is performed using numpy, sklearn, pandas, and matplotlib. Model cross-validation loop is managed through the Python bayesian-optimization package, and no other external dependencies outside the Python data science ecosystem are used in our project.

Our code implements the functionality to train, assess, and optimize a VAE regardless of its loss function or its set of hyperparameters. Function wrappers were used to provide an interface between the model training/validating process and a bayesian optimizer searching over the hyperparameters latent dimensions and hidden layers. We restricted our latent search space to [5, 50] to ensure our models reduce the dimensionality of our data. Our hidden layers were restricted to [100, 400]. Our reasoning for this was that models that are too complex may easily overfit, and models that are too small may have difficulty reconstructing the non-anomalous data.

## 6 Contributions

- Connor: created baseline VAEs, EDA plotting, data splitting/stratification, trained/tested best models, mathematical formulas, and results tables.
- Aaron: Model architecture, MLVAE, training/testing, mathematical formulations, report writing, some bayesian optimization code.
- Peyton: Primary scribing on introduction/methodology/results/discussion sections, train-validation-test pipeline layout, EDA for stratification strategy and minor code troubleshooting.
- Rishabh: devised the class stratified testing scheme and implemented within the CV inner loop, orchestrated the Bayesian optimizer in CV outer loop, and unified all models to have a consistent interface.

## References


[1] Craig Heffner, *Exploiting Network Surveillance Cameras Like a Hollywood Hacker*. BlackHat USA 2013, July 2013.

[2] B. S. Sharmila, Rohini Nagapadma, *Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset*. Cybersecurity, Volume 6, Issue 41, September 2023.

[3] Torabi, H., Mirtaheri, S.L., Greco, S., *Practical autoencoder based anomaly detection by using vector reconstruction error*. Cybersecurity, Volume 6, Issue 1, January 2023.

[4] Oliveira, Nuno, Isabel Praça, Eva Maia, Orlando Sousa, *Intelligent Cyber Attack Detection and Classification for Network-Based Intrusion Detection Systems*. Applied Sciences, Volume 11, Issue 4, January 2021.