A Novel Application of Variational Autoencoders to Network Intrusion Detection Systems (NIDS)

1. Introduction

The purpose of this project is to produce a novel approach to anomaly detection in network traffic to produce a hypothetically effective network intrusion detection system designed to detect both commonly known malicious network traffic and previously unknown malicious network traffic. This problem is an anomaly detection problem for which we will be leveraging supervised learning and the variational autoencoder (VAE) architecture.

An autoencoder is typically a neural network that uses a series of hidden layers of decreasing numbers of nodes down until some final smallest hidden layer, called an encoder and then hidden layers of increasing numbers of nodes until eventually return to the same dimensionality of the inputs, called a decoder which produces the compressed representation also called the latent representation of the input. The model is trained by using the input to the model as the ground truth to provide the model signal to train on. As such we are training the model to simply reconstruct the output which is equivalent to minimizing the reconstruction loss which is effectively the difference between the input and what the model generated. Where the VAE differs from an autoencoder is in the final smallest layers of the encoder the model generates two vectors encoding the mean and variance of distribution. By sampling from this distribution we introduce some stochasticity into the model which enables it to learn more meaningful and interpretable latent representations than a standard auto encoder.

The objective of this project is to leverage a threshold of the reconstruction loss of a VAE to build a predictive model for the nature of network traffic.

1.1. Current Practices and Challenges

Machine learning has been leveraged in this area of network security for the task classification and analysis of network traffic. However, historically there is a precedent for reliance on what are called signature based network intrusion detection systems also referred to as misuse detection. “Misuse detection typically leverage rule based string search and pattern matching algorithms to detect attacks by searching for specific strings in network activity. A substantial benefit of these kinds of traditional misuse detection approaches is a generally low false positive rate (FPR). This is an issue that has plagued other camps of intrusion detection systems chiefly anomaly detection based systems in which our model falls [1].” “In addition, high FPR models face other challenges such as data imbalance, computational complexity, adversarial attacks, computational overhead, and the ever evolving nature of cyberattacks [2].” Additionally the domain of network activity itself poses substantial challenges to the practical applications of machine learning. The primary issue being the highly dynamic nature of networks such that what an anomaly detection model should perceive as anomalous and normal can vary greatly by organization and by time of day.

1.2. Impact

This approach would perhaps be novel and effective enough to justify the deployment of such a system in a home or workplace environment as an additional network security measure. This approach will hopefully produce results that marginally improve upon some of the shortcomings of previous approaches primarily, FPR, and the evolution of cyberattacks, in a way that makes it practical as a lightweight alternative or a harmless additional layer of security.

1.3. Approach

I thought that due to the nature of network traffic containing a lot of what can be superfluous information it would be to my benefit to utilize a model that is designed to compress and remove superfluous noise in data through compression.

The following is an implementation of a VAE approach to network intrusion detection on the NSL-KDD dataset which is a shrunken and easier version of the KDD.

“The VAE was adapted from reference implementation from the keras website and provided a clean implementation of a sampling layer and framework to make the necessary modifications to be used with this dataset [5].”

“As per the classical VAE architecture this model

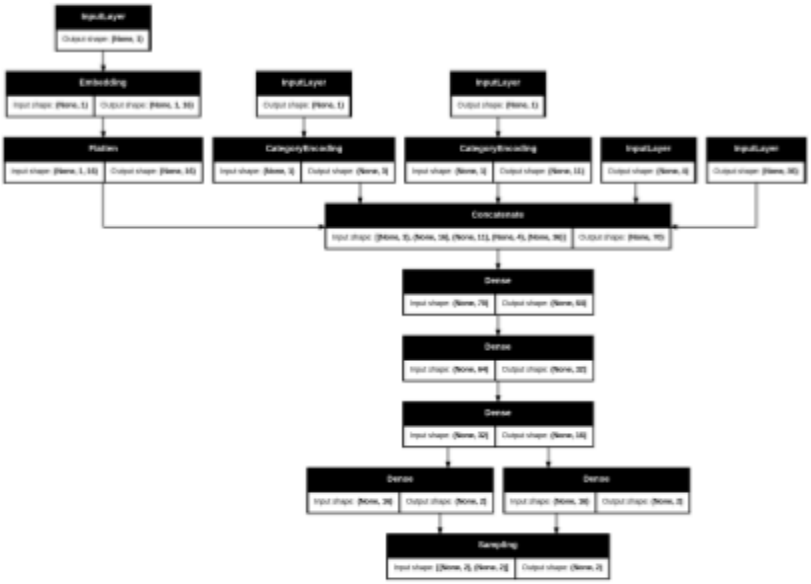

Figure 1: Encoder portion of the VAE.

consists of an encoder and decoder model [3].” The NSL-KDD dataset consists of many discrete and continuous fields that need to be handled with care for our model to be able to derive interpretable meaning from them. The first three important fields were those of service, protocol, and flag each of which is a discrete field from one of a finite number of labels as such we label encode each discrete field and pass them into keras input layers which subsequently run to one hot encoding layers or for the service class due to its high number of unique values an embedding layer to keep input to the later layers slim. The remaining fields contain either binary values 0 or 1, rates represented as numbers between 0.00 and 1.00 inclusive, or counts of integers greater than or equal to 0. The binary values were passed to input layers unchanged. However, after statistical analysis the counts of the integers were highly skewed based upon their mean/median difference as well as an unbalanced IQRs. As such I applied some preprocessing to the integer data through a log transformation and subsequent standardization before passing them to input layers.

The multi-input architecture with separate processing paths (embeddings for categorical, standardization for continuous) reflects the heterogeneous nature of network traffic data. The multi-head decoder reconstructing each feature type independently allows feature-specific loss functions matching the original data types.

Then all of the previous layers up until this point are concatenated and fed into an encoder consisting of dense layers of sizes 64, 32, 16 nodes. Then the model feeds the output of the layer of 16 nodes to two other dense layers of 2 nodes to produce our latent space mean and variance vectors. These vectors are then passed to a custom sampling layer to produce a stochastically sampled output.

Next our decoder takes our stochastically sampled output from the encoder and passes it through an architecture mirroring the encoder, of dense layers of 16 , 32, 64 nodes. Then our decoder splits into 5 heads one for

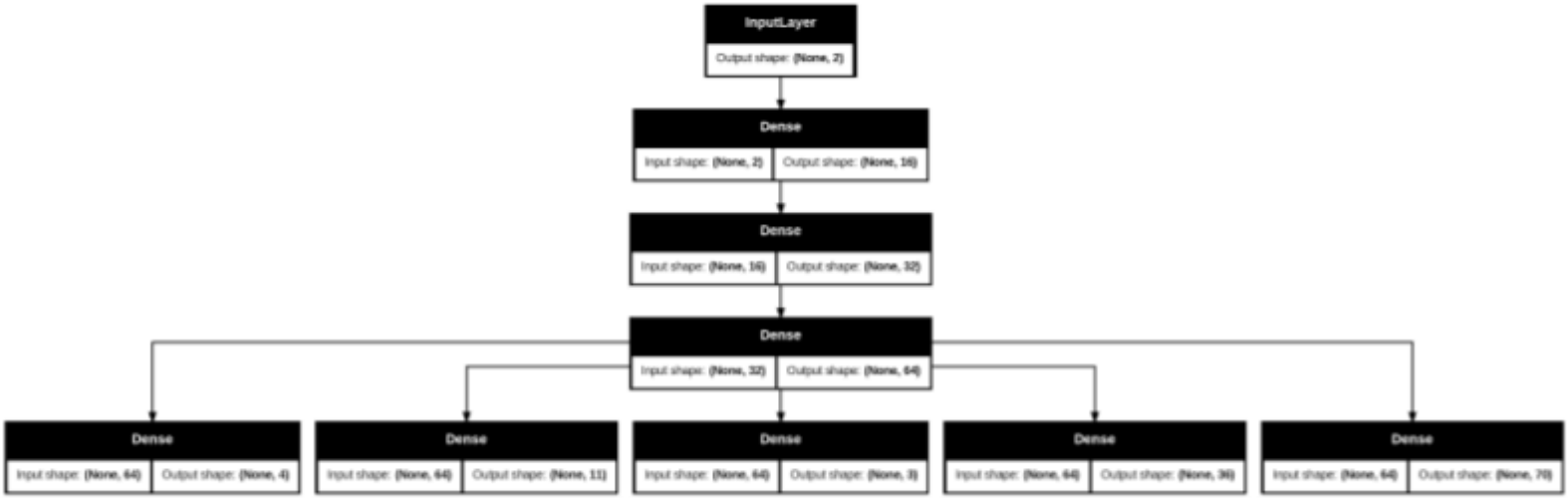

Figure 2: Decoder portion of the VAE.

each of the classes of input for reconstruction. The model will reconstruct the service, flag, and protocol encodings using sparse cross entropy, the binary inputs using binary cross entropy and the integer fields using mean squared error for the count fields. As such we needed to use a multi-head weighted sum loss including KL divergence to actually quantify the reconstruction loss in order to be able to train this model which was done using the default keras Adam optimizer. All neural network weights (encoder, decoder, embeddings) totaling ~16,000 parameters were learned during training. Non-learned components include the one-hot encoding transformations, StandardScaler parameters (fitted on training data then frozen), and the detection threshold (empirically selected post-training).

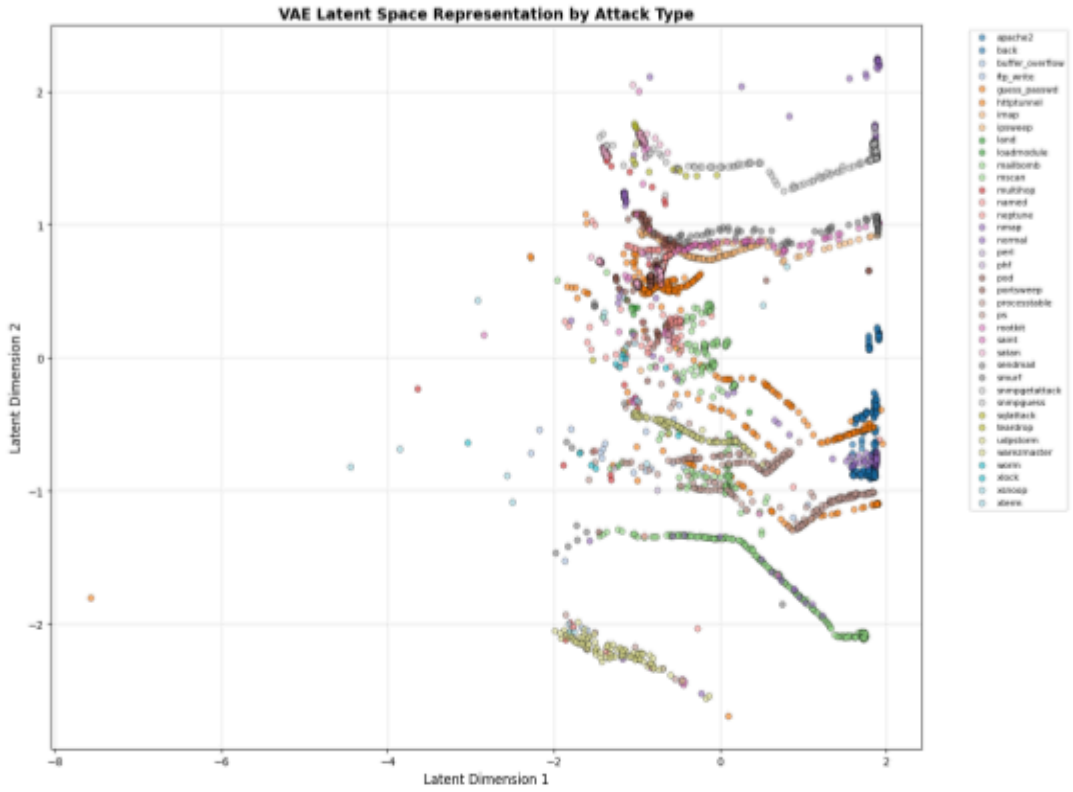

Figure 3: A visual representation of the latent space embeddings produced as the means of the variational autoencoder encoder network

1.4. Challenges

My initial intentions were to leverage the latent embeddings produced by the encoder, but the interpretability of the latent spaces of VAEs is quite a challenging problem for certain inputs as many forms of latent space interpretability collapse under training. As such, I found the latent space interpretability for this problem to be especially brittle and moved forward with an alternative approach.

The approach being that the VAE should only effectively learn to reconstruct the data upon which it was trained. As such, through training the model on benign traffic, when abnormal traffic was reconstructed, it would result in orders of magnitude higher reconstruction error. Using this key idea I constructed a VAE design to produce an error rate as an output. This approach, while much less sophisticated, produced far more robust and desirable results.

The most significant challenges with this project arose in the training of the model for this approach to anomaly detection. Training a model for anomaly detection as we were differs substantially from standard supervised learning. The loss of the model suddenly became almost a non-factor in the ability of our model to produce different reconstruction results for anomalous data. As such we

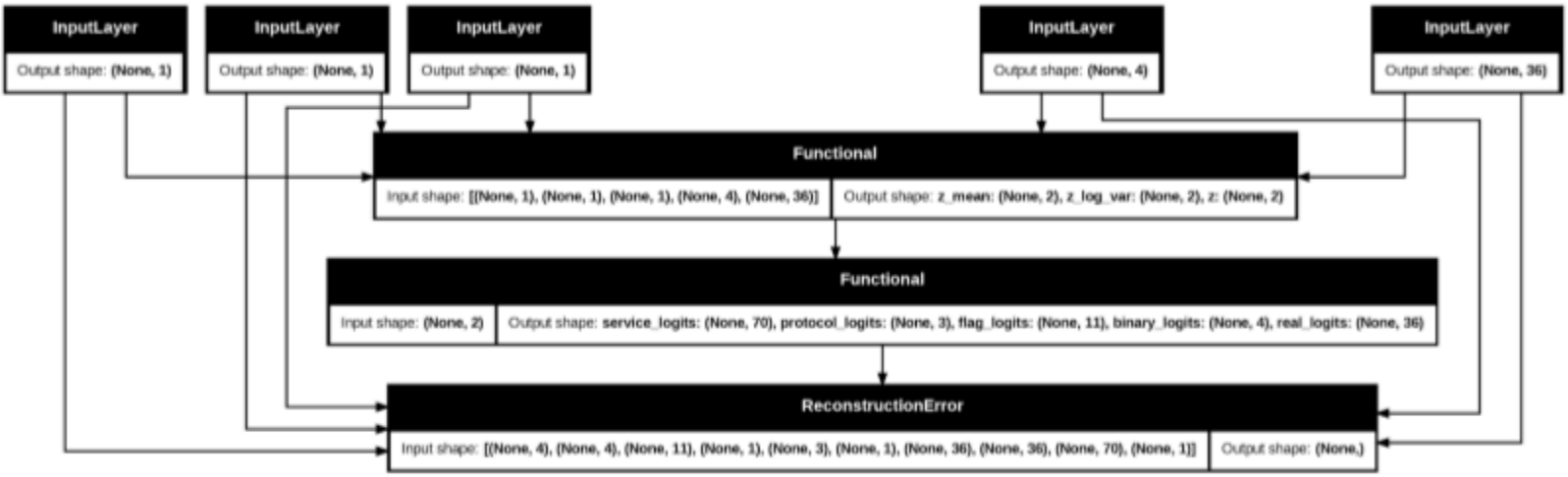

Figure 4: Full VAE showing the encoder and decoder functional layers as well as model inputs and outputs.

needed a new metric to quantify the performance of the model for our desired task and control how much training we should actually perform. "For this new metric the classical anomaly detection approach suggests the use of Area Under Receiver Operating Characteristic Curve (AUROC). [4]" This metric quantifies the quality of all possible thresholds for binary classification by seeking to maximize the area under the Receiver Operator Characteristic Curve (ROC) corresponding to the model that has thresholds that best minimize FPR and maximize true positive rate (TPR). This will inform us as to whether the model produces reconstruction errors that created more separability in the output which is our goal in this anomaly detection problem. So we implemented a call back function that calculates the AUROC on a validation set after every epoch to determine if any actual improvement in the separability of data actually occurred as a result of additional training. This is used as a metric to halt training early if improvements in the AUROC are not present after 5 epochs of training. The quality of the separation in our data on the validation set improved drastically with the use of the AUROC training metric as opposed to blindly iterating for a certain number of epochs or until the convergence of the validation loss.

Figure 5: A graph measuring the AUROC against the number of iterations of training.

1.5. Results

I measured success for this project through a few metrics. One of these metrics being the previously mentioned AUROC as it effectively quantifies the ability of our model to enable good anomaly detection from our experimentation we attained an AUROC of ~0.97.

Figure 5 shows validation AUROC peaked at epoch 4 (~0.97) before declining, indicating the model began overfitting. Early stopping at 5 epochs patience preserved the best generalization performance. The model achieved similar AUROC on validation (~0.97) and final test evaluation (~0.964), demonstrating good generalization. Training solely on normal traffic while successfully detecting diverse attack types indicates the model learned generalizable patterns rather than memorizing specific examples.

Due to the nature of our anomaly detection task and early stopping our final model shows no signs of overfitting as overfitting would cause the reconstruction error of normal traffic in our validation set to rise and lower the AUROC triggering early stopping.

AUROC does not carry much practical statistical significance. So additionally I chose the optimal threshold for reconstruction error empirically through the validation set, that balances minimizing FPR with the recall and accuracy for balanced performance.

The 95th percentile threshold balanced detection rate (83.9% recall) with acceptable false alarm rate (4.1% FPR); the 90th percentile achieved higher recall (87%) but at the cost of 7% FPR, while the 99th percentile reduced FPR to 1% but missed too many attacks (67% recall).

The confusion matrix created through validation testing is my other means of validation for the models performance. The model threshold produced an accuracy of 88.8%, precision of 95.9%, recall of 83.9% and an F1 of 0.895.

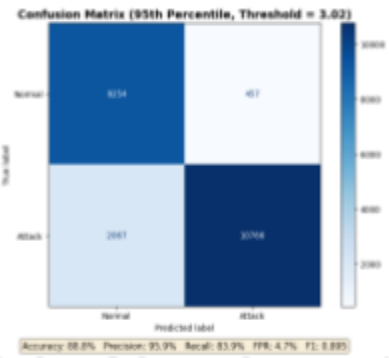

Figure 6: A confusion matrix generated through validation testing the prediction results of an error threshold of the 95th percentile of the reconstruction error rate of normal traffic.

Additionally, I could qualitatively observe the distributions of reconstruction error rates by plotting the error rates on their attack type.

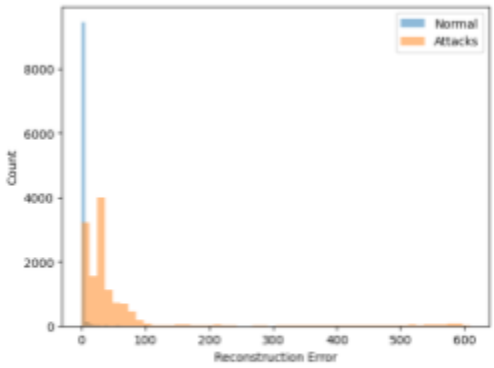

Figure 7: A graph measuring the distribution of the reconstruction errors in the validation set..

This work demonstrates that VAEs can effectively detect network intrusions with 83.9% recall and only 4.1% false positive rate. While this represents a viable complementary detection layer to traditional signature-based systems, the model still suffers from many of the pitfalls of anomaly detection based network intrusion detection.

1.6. Availability

My code is available on github at the following link:

https://github.com/jasperhopkins/vae_nids

My findings and the code used to train the model and generate the graphics are made available in my github repository under an MIT license.

1.7. Reproducibility

The NSL-KDD dataset is free and openly available. I specifically used the KDDTrain+.txt as training data and KDDTest_.txt as validation data. I will include the specific testing and training data I used within the github repository in addition to the model parameters saved in the .keras model checkpoint format. The model results are fully reproducible using the pseudo random seed of 1337 with the pretrained model within the repository.

I trained my model with the default keras Adam optimizer and the following hyperparameters for the multi-head weighted sum loss:

service: 2.0
protocol: 1.5
flag: 1.5
binary features: 3.0
real features: 1.0
KL divergence: 0.1

Training environment:

Python	3.10
TensorFlow	2.19.0
Keras	3.10.0
NumPy	2.0.2
Pandas	2.2.2
Scikit-learn	1.6.1
Matplotlib	3.10.0

Model architecture: - Encoder layers: [64, 32, 16] nodes - Latent dimensions: 2 - Decoder layers: [16, 32, 64] nodes - Batch size: 32 - Learning rate: 0.001 (Adam default) - Max epochs: 100 (early stopped at ~7-12)

Hyperparameters were selected through empirical validation. Loss weights prioritized features by importance: binary flags (3.0) received highest weight as connection state is critical for attack detection, while continuous statistics (1.0) served as baseline. The KL divergence weight (0.1) was reduced from the standard β=1.0 after observing posterior collapse at higher values during initial experiments. The 2-dimensional latent space was chosen for interpretability.

References

[1] Nazim Uddin Sheikh, Hasina Rahman, Shashwat Vikram, and Hamed AlQahtani. A lightweight signature-based IDS for IoT environment. arXiv preprint arXiv:1811.04582, 2018

[2] Kazeem M. Olagunju, Ayodele A. Adebiyi, Timothy T. Adeliyi, Jonathan P. Oguntoye, Emmanuel O. Igbekele, and Seyitanfunmi Osunade. State-of-the-art research in intrusion detection systems (IDS) for network attack detection: A review. NIPES Journal of Science and Technology Research, 7:974–981, 2025.

[3] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.
[4] Vít Škvara, Tomáš Pevný, and Václav Šmídl. Is AUC the best measure for practical comparison of anomaly detectors? Data Mining and Knowledge Discovery, 2023.
[5] Keras Team. Variational autoencoder example. https://keras.io/examples/generative/vae/, 2020.