Intrusion Detection System

Unsupervised Machine Learning for IDS

CSCA 5632: Final Project

Ryan Lynch (lynchrl@colorado.edu)

The problem

Ubiquitous Threats

Threats to computer network security are ubiquitous and ever changing, with threat vectors surfacing internally and externally.

Extreme Volume

Today's networks transition many Mbps or Tbps, requiring network security solutions that can handle extremely high volumes of traffic metadata.

Real Time

Network security systems need to work at near real time to identify and stop malicious actors before they can cause damage.

Solution

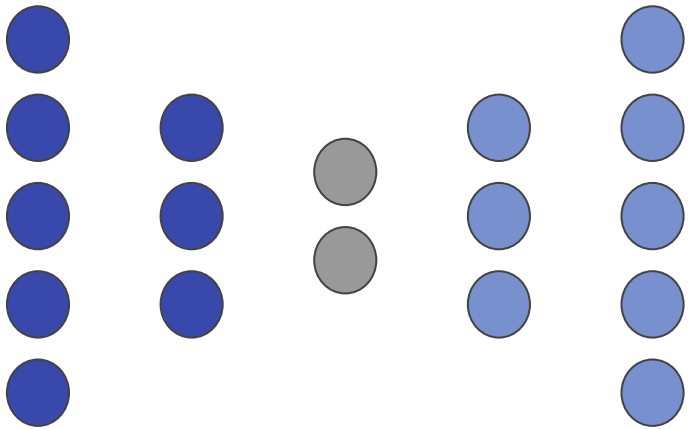
IDS via Anomaly Detection

Various types of machine learning models can be trained to perform anomaly detection. By training on “normal” traffic flows, anomalies (potential threats) can be identified by such models as outliers.

Implementation

Autoencoder for Anomaly Detection

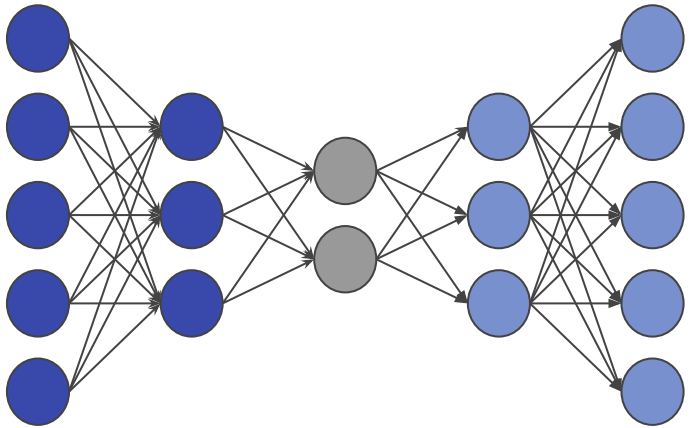
Our IDS uses a classic Autoencoder neural network for anomaly detection.



Autoencoder for Anomaly Detection

An Autoencoder is a fully connected feed forward neural network.

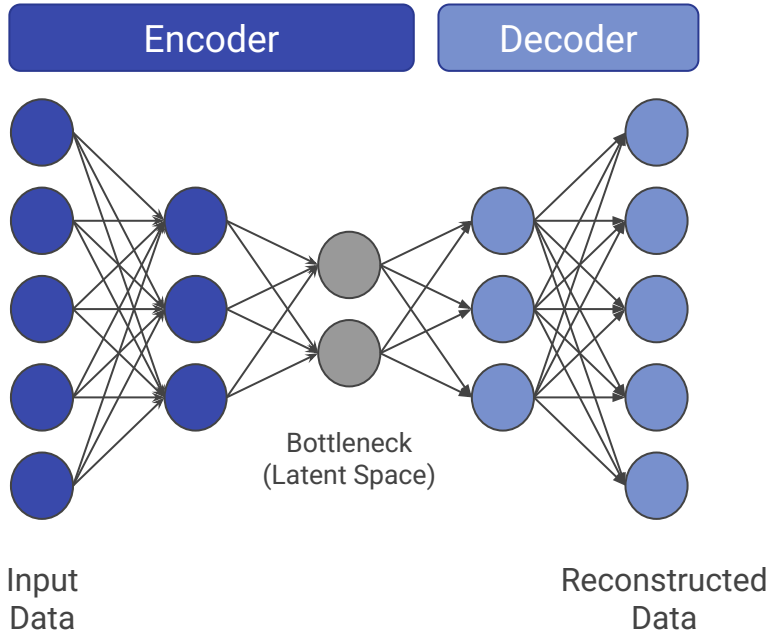
They typically look like an hourglass on its side with the number of neurons on the left/right (top/bottom) being equal.



Autoencoder for Anomaly Detection

The input side is the Encoder, which compresses the data down to the latent layer (bottleneck).

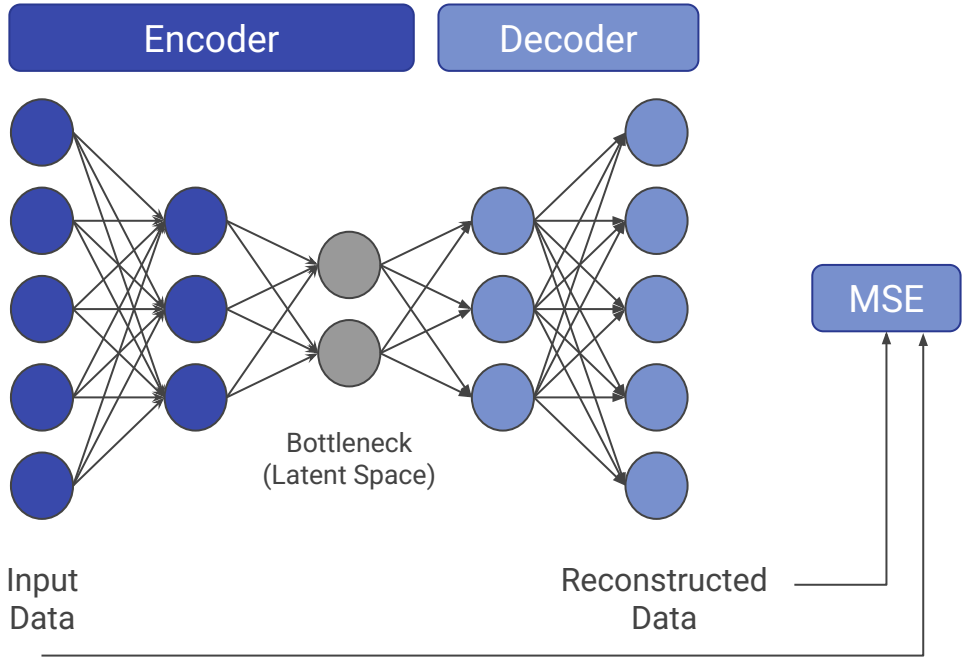
The output side is the Decoder, which attempts to reconstruct the latent representation back to the original input.



Autoencoder for Anomaly Detection

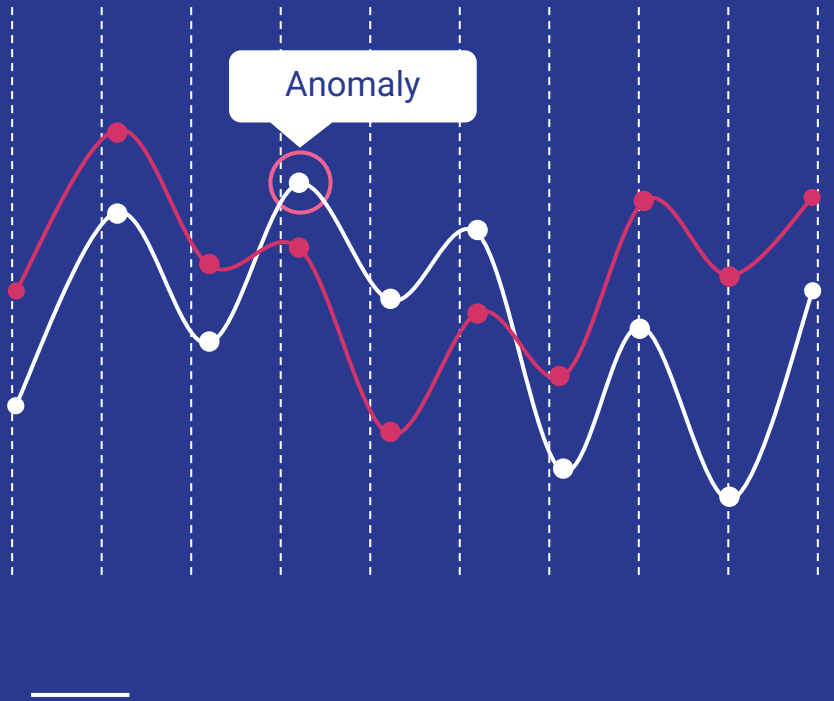
MSE is based on the difference between Input Data and Reconstructed Data. The objective function minimizes MSE.

The model is trained on normal data, and outliers are detected during inference based on an MSE threshold.



Results

98% Anomaly Recall



Results: Training/Evaluation Data

Training and evaluation used the Network Intrusion Detection dataset on Kaggle.

Characteristics:

- TCP and UDP
- Flow based
- 40 quantitative and qualitative features
- ~25k flow samples
- Labeled (normal or anomaly)

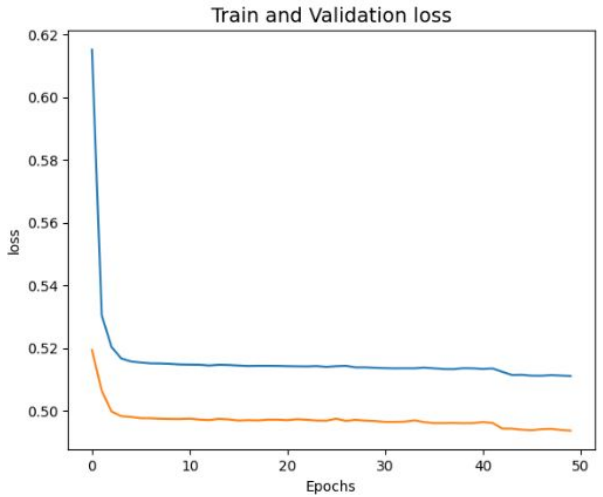
Results: Modeling and Training

Training is completed using only the **normal** data (i.e. exclude anomalies).

Labels are removed for training -> unsupervised training.

Encoding and normalization was completed on training data.

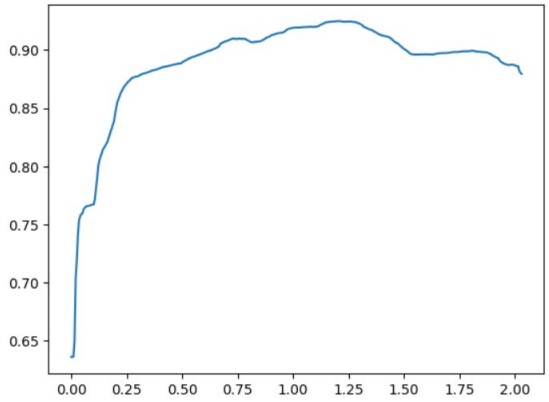
Architecture: A small 64->32->16->32->64 fully connected NN.



Results: Performance Analysis

	precision	recall	f1-score	support
normal	0.97	0.81	0.89	13449
anomaly	0.82	0.98	0.89	11743
accuracy			0.89	25192
macro avg	0.90	0.89	0.89	25192
weighted avg	0.90	0.89	0.89	25192

Baseline MSE (0.508)



	precision	recall	f1-score	support
normal	0.94	0.92	0.93	13449
anomaly	0.91	0.94	0.92	11743
accuracy			0.93	25192
macro avg	0.93	0.93	0.93	25192
weighted avg	0.93	0.93	0.93	25192

F1 Opt MSE (1.206)

Results: Benefits

- Simple model, converges quickly
- Fast inference (~1s for 25k flows)
- Flexible tuning without retraining
- Effective as one piece of a robust IDS

References

- Jupyter notebook on Kaggle:
  - https://www.kaggle.com/code/lynchrl/csca-5632-final-project
- Jupyter notebook on GitHub:
  - https://github.com/lynchrl/csca5632/blob/main/csca-5632-final-project.ipynb
- GitHub repo:
  - https://github.com/lynchrl/csca5632/

These slides are available in the GitHub repository.