# Anomaly Detection in System Logs Using Generative Adversarial Networks

Kelvin Benedict[1], Aidan Call[1], Jack Feller[1], Lance Ma[2], Kora Gwartney[3], Lingyu Li[2], Jacob Jannotta[2], Christopher Brantley[2], Katie O'Daniel[2], Austen Suqi[2]

1 FORT LEWIS COLLEGE 2 PURDUE UNIVERSITY 3 UCCS

LOCKHEED MARTIN

## Introduction

- Modern networks generate massive **volumes** of system logs and traffic data at high **velocity** and with great **variety** in format and content. This makes manual monitoring impractical.
- Cyber threats and **anomalies** often hide within this complexity.
- Traditional rule-based systems cannot adapt fast enough to detect new or evolving threats.
- **Machine learning offers a dynamic solution**, able to learn patterns and detect unusual behavior automatically.
- Our project investigates the use of **Generative Adversarial Networks (GANs)** for anomaly detection in system logs.
- We include **LSTM-based Autoencoders (LSTM AEs)** as a performance baseline.

## Research Questions

- Can GANs effectively learn patterns of normal system behavior and highlight deviations that may represent anomalies?
- How do GAN-based models compare to AEs in terms of performance and detection accuracy?
- How can knowledge graphs be integrated into this process to provide structure and context that may enhance the interpretability and precision of anomaly detection?

## Research Methodology

An **experiment** was used to evaluate the use of GANs for anomaly detection based on the metrics of **accuracy, precision, recall, F1 score, specificity, and false positive rate**. An Autoencoder was used as the baseline for comparison as it is the current industry standard.

Utilized the Anvil Supercomputer and Python for training and testing

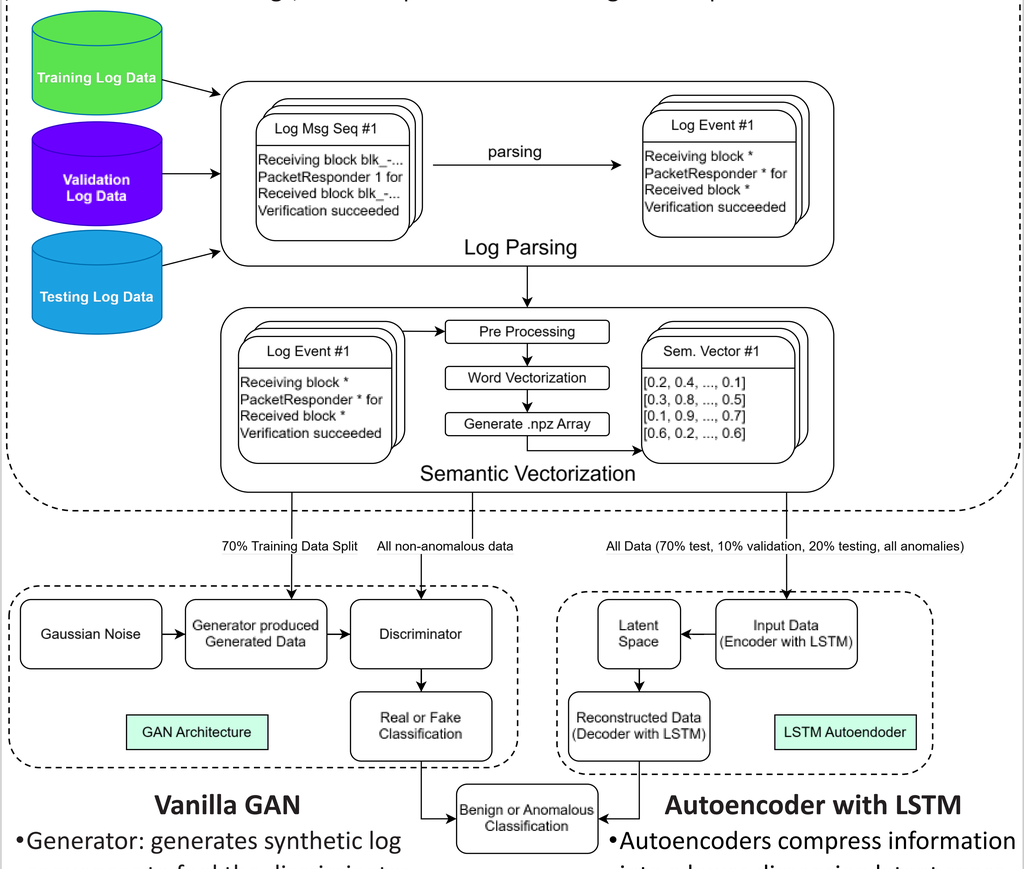
Measure all metrics on both models to standardize the experiment

Compare metrics to evaluate performance

## Research Process

### Data Preprocessing

- Convert raw, unstructured logs into structured templates using Drain.
- Partition logs into meaningful sequences based on identifiers or temporal windows or sliding windows for systems without identifiers.
- Create embeddings, vector representations of logs that represent their context




### Vanilla GAN

- Generator: generates synthetic log sequences to fool the discriminator
- Discriminator: distinguishes between real and fake logs event to identify deviations that indicate anomalies

### Autoencoder with LSTM

- Autoencoders compress information into a lower dimension latent space and then recreate them from the simplified representation
- Errors in the reconstruction can be indications of anomalies

### Transformers

- Transformers are excellent at recognizing relationships within data, usually natural language.

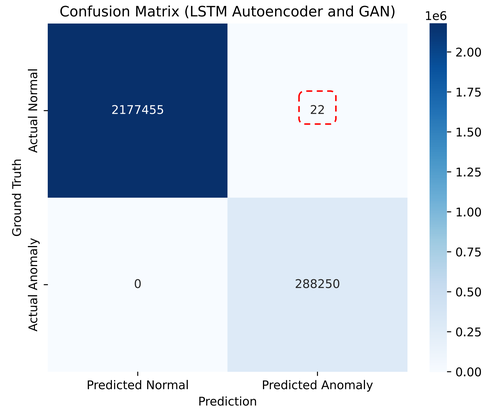
### Knowledge Graph Integration

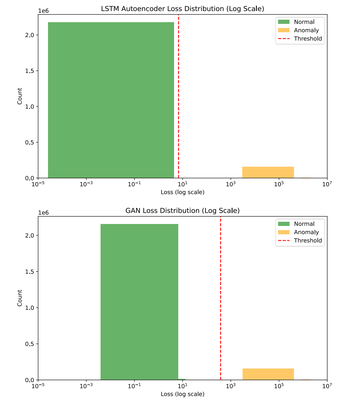
- Knowledge Graphs are often used to find patterns in relationships between entities in a network.

### Transformer-Based GAN Development

- Based on the benefits of Transformers and Knowledge Graphs a more complex GAN was built to determine if it would enhance pattern recognition within logs (Transformer) and between entities in many logs (Knowledge Graph)

## Results and Conclusions




Both Models had **identical** performance. They misclassified the same 22 logs.

The **Autoencoder** loss shows slightly more separation between normal and anomalous data.

| Metric | Vanilla Gan | LSTM AE |
|---|---|---|
| Accuracy | 99.9991% | 99.9991% |
| Precision | 99.9924% | 99.9924% |
| Recall | 100.00% | 100.00% |
| F1 Score | 99.9962% | 99.9962% |
| Specificity | 99.9990% | 99.9990% |
| False Positive Rate | 0.0010% | 0.0010% |

### Conclusion

- Both models have **identical metrics**, proving legitimacy in both GANs and autoencoders
- However, the **GAN ran faster in training and inference**
- While knowledge graphs are powerful tools for machine learning models, in this case, we saw an **increase** in training and test time and a **decrease in accuracy**
- Standard GANs or LSTM-based autoencoders will work well for this use case

## Future Goals

- Test this model against other data sets to determine performance across systems.
- Test the model with a smaller set of training data to evaluate requirements to efficiently train a model.
- Test methods to improve model inference time.

## Acknowledgements

We thank Jonathan Conover for his mentorship, Lockheed Martin Corp, and Purdue Data Mine for making this project possible.

**Data Mine of the Rockies Spring 2025**