

The top-left corner features a white line-art leaf inside a tan circle, with a white line curving around it. The top-right corner shows a white line-art branch with two buds and leaves, set against a dark blue circle.

GROUP: BUMBLEBEE

Anomaly Detection in Network Traffic
using a Transformer Autoencoder

The bottom-left corner has a white line-art branch with two buds and leaves, set against a dark blue circle. The bottom-right corner features a white line-art leaf inside a tan circle, with a white line curving around it.



TEAM MEMBERS

Alexios C Papazoglou

Geetika Khanna

Mohit Saluru



Pragati Rao





PROBLEM & MOTIVATION





- **Challenge:** DDoS attacks threaten online services; traditional supervised methods fail against zero-day attacks
 - **Innovation:** Reframe DDoS detection as sequence reconstruction problem using Transformer autoencoders
 - **Advantage:** Detect both known and novel attacks by learning only from benign traffic patterns
- 
- 





DATASET & SCALE



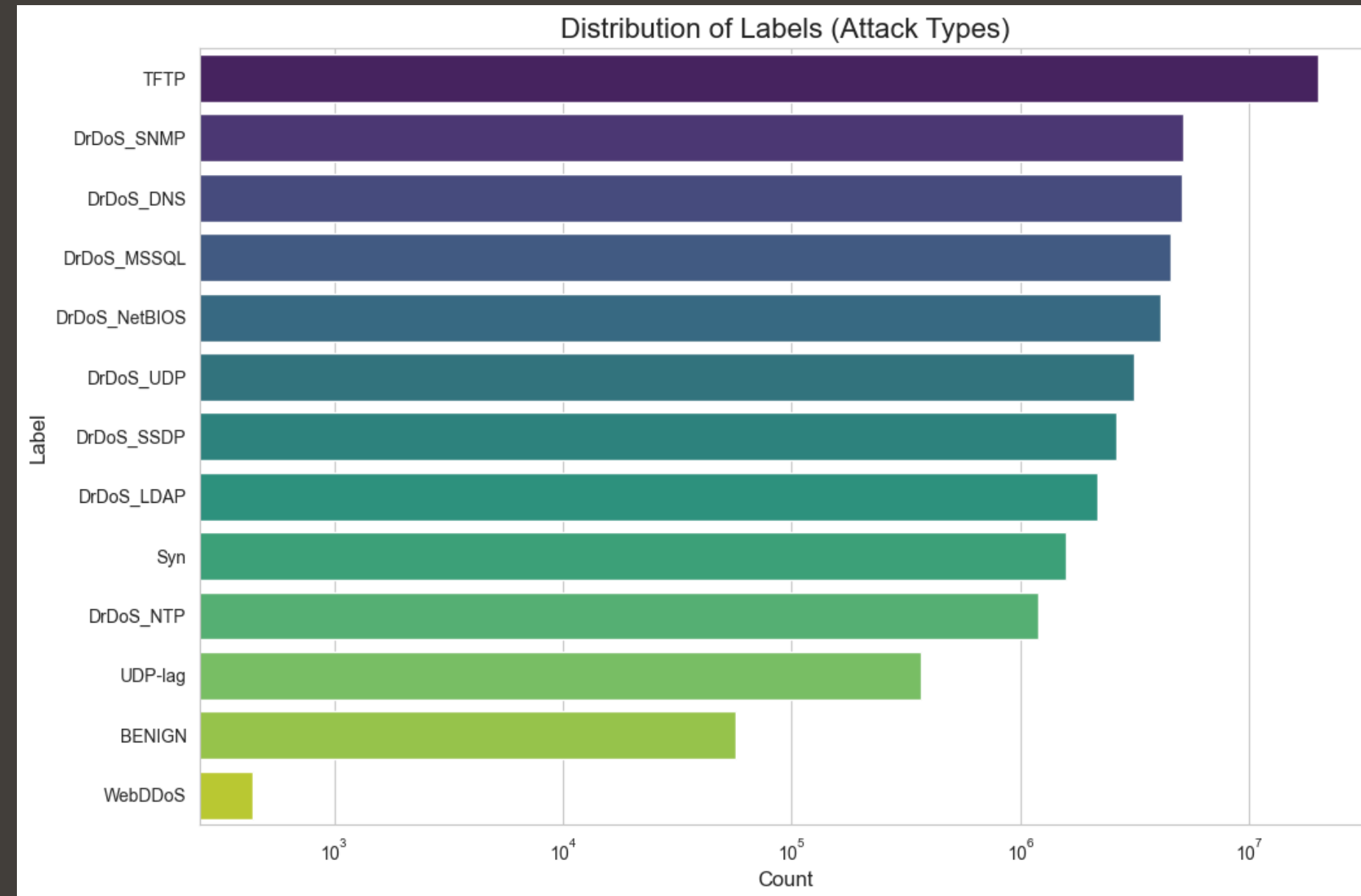
- **CIC-DDoS2019 Dataset:** 50+ million network flow records with 84 features
 - **Extreme Imbalance:** Only 0.11% benign traffic (56,863 samples) vs 99.89% attacks
 - **Processing:** Transformed into 56,326 sequences of 100 timesteps each for training and testing
- 
- 



DATA PREPROCESSING

- To handle the dataset's massive size, we developed a memory-safe, chunk-based processing pipeline.
 - Extracted only benign flows for autoencoder training
 - Dropped rows with missing or infinite values to ensure data integrity
 - Timestamp normalized to seconds since start of capture
 - Created overlapping sequences of 100 time steps (sliding window)
 - Final dataset shape: 56,326 samples \times 100 timesteps \times 84 features
 - Preprocessed data stored as .npy for fast loading during training
 - Ensured consistent scale across features using StandardScaler normalization.
 - Additionally, created different .npy files for 12 different attack vectors.
- 
- 

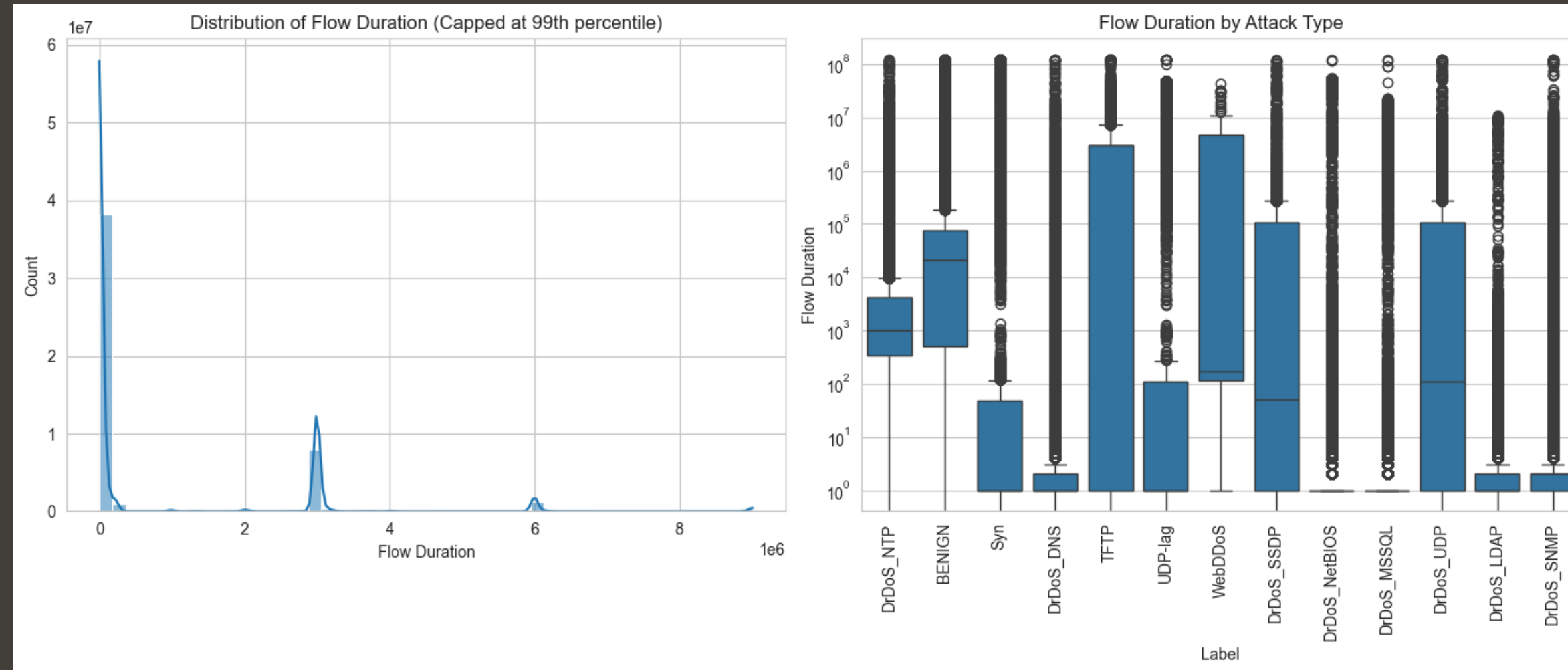
DATA VISUALIZATIONS



Severe Class Imbalance

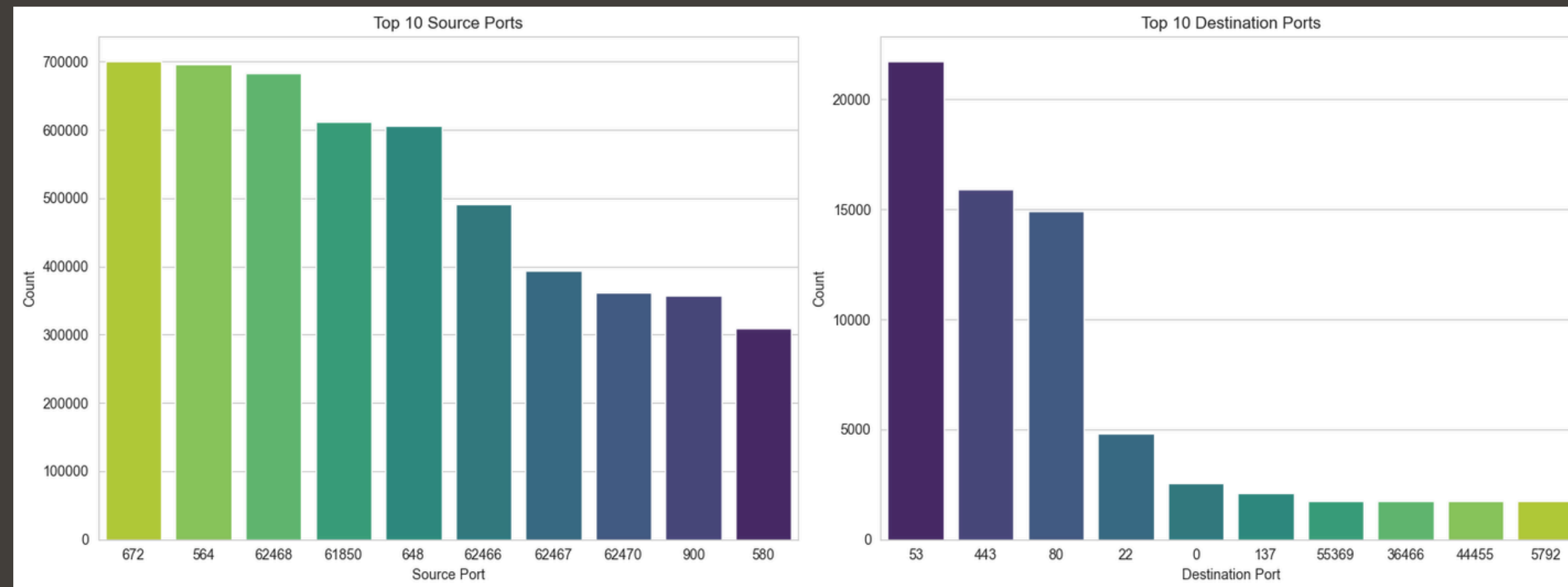
Benign traffic is only 0.11% of the dataset, validating our unsupervised approach.

DATA VISUALIZATIONS



Distribution of Flow Duration

DATA VISUALIZATIONS



Top 10 Source and Destination Ports



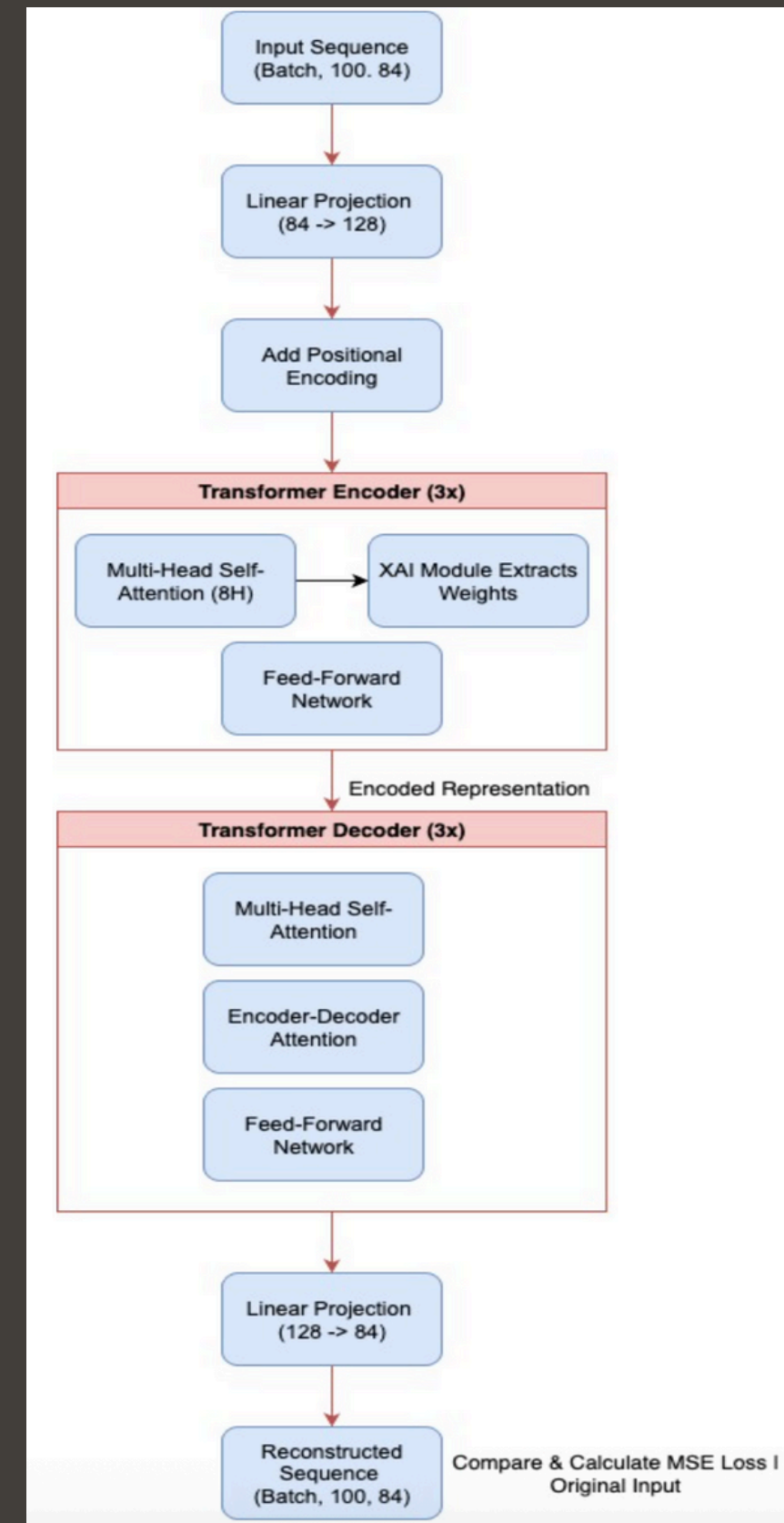
OUR SOLUTION : A THREE-PART STRATEGY

We tackled this with a robust engineering and modeling pipeline:

1. **Part 1 - Scalable Data Processing:** We engineered a memory-safe, chunk-based pipeline that can process terabyte-scale datasets on a standard machine. (This was a major challenge we solved.)
2. **Part 2 - Unsupervised Learning:** We trained a Transformer Autoencoder to become an expert on "normal" behavior, allowing it to spot anything that deviates.
3. **Part 3 - Explainable AI (XAI):** Our model doesn't just raise an alarm; it shows a security analyst where to look via attention heatmaps.

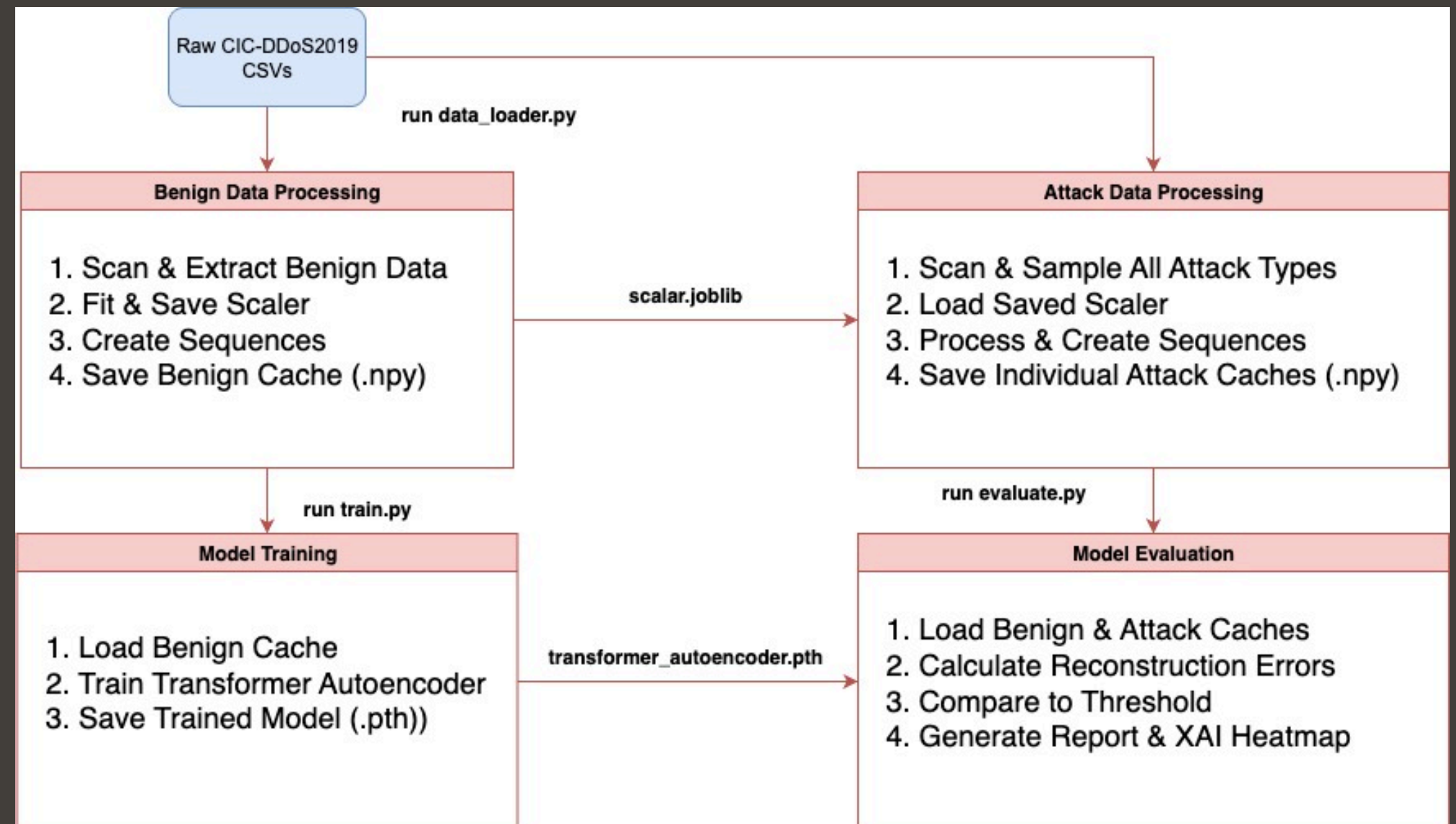
MODEL OVERVIEW & CORE INNOVATION

- **Input Projection:** A linear layer projects the input features of each time step to the model's internal dimension ($d_{model}=128$).
- **Positional Encoding:** Sinusoidal positional encodings are added to the input to give the model information about the order of time steps in a sequence.
- **Encoder:** A stack of 3 Transformer Encoder layers, each containing a Multi-Head Self-Attention module (8 heads) and a Position-wise Feed-Forward Network.
- **Decoder:** A stack of 3 Transformer Decoder layers that takes the encoder's output (the compressed representation) and attempts to reconstruct the original input sequence.
- **Output Projection:** A final linear layer projects the decoder's output back to the original feature dimension.



ARCHITECTURE DATA FLOW

- **Data Processing Parameters:**
 - SEQUENCE_LENGTH = 100
- **Model Hyperparameters:**
 - INPUT_FEATURES = 84
 - MODEL_DIM = 128
 - NUM_HEADS = 8
 - NUM_ENCODER_LAYERS = 3
 - DIM_FEEDFORWARD = 512
 - DROPOUT = 0.1
- **Training Hyperparameters:**
 - LEARNING_RATE = 0.0001
 - BATCH_SIZE = 64
 - NUM_EPOCHS = 20






DEMO



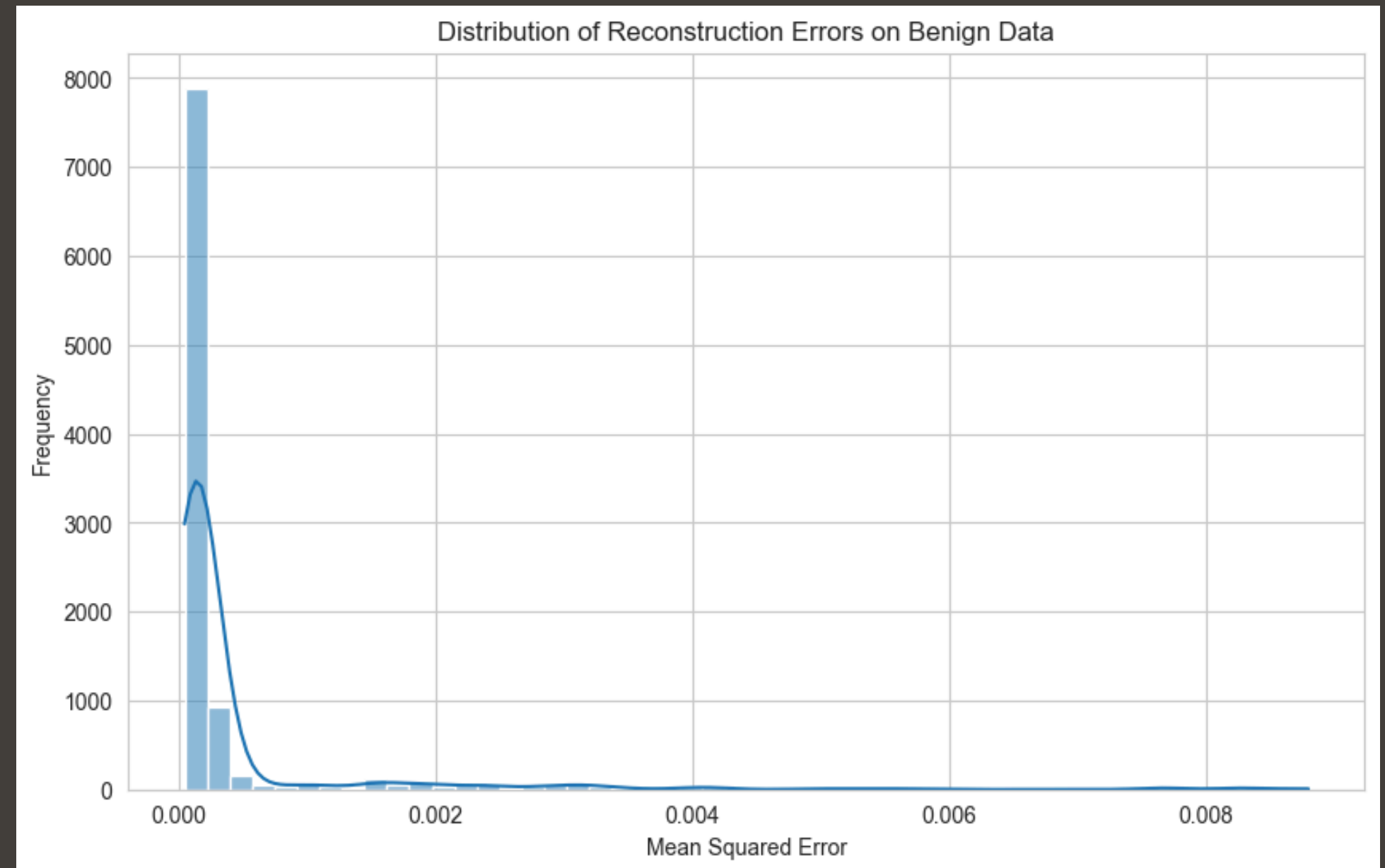


TOOLS AND TECHNOLOGIES USED

- 
- **Programming Language:** Python
 - **Deep Learning Framework:** PyTorch
 - **Data Manipulation:** Pandas, NumPy, PyArrow
 - **Machine Learning & Preprocessing:** Scikit-learn, Joblib
 - **Visualization:** Matplotlib, Seaborn

EVALUATION

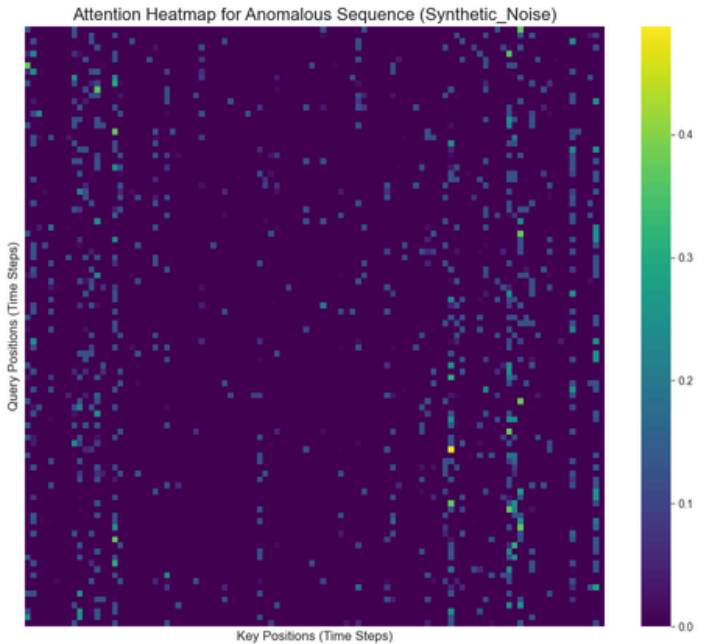
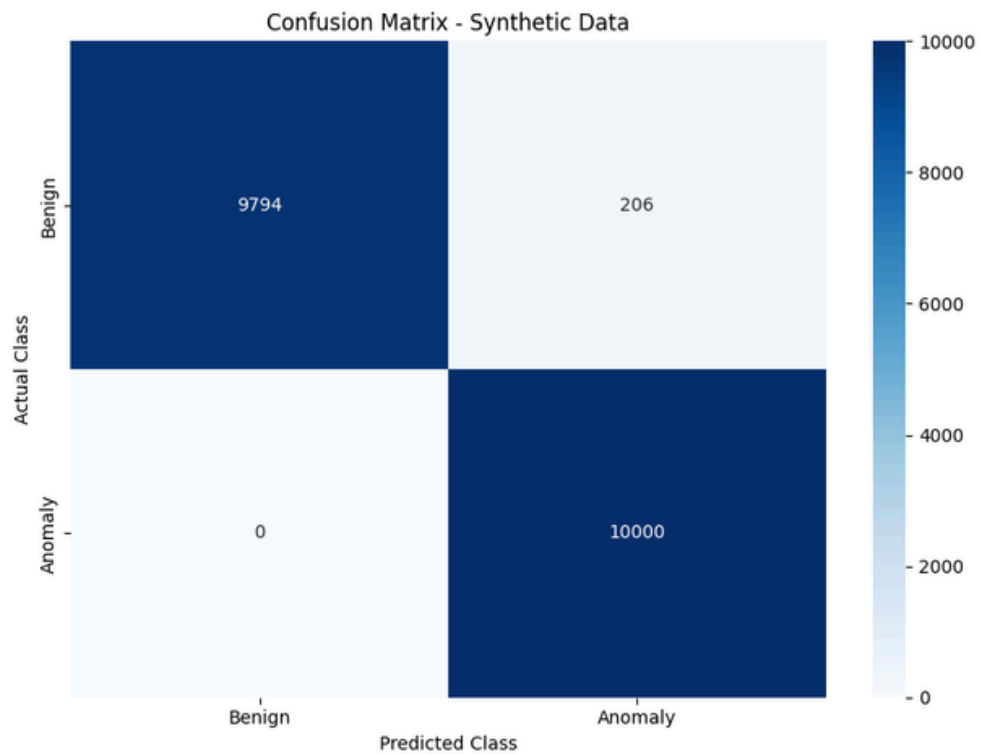
The model was evaluated using two distinct test sets. The anomaly threshold was set at the 99th percentile of reconstruction errors on a benign validation set.



EVALUATION

Test1: Synthetic Data (Noise)

- **Overall Accuracy: 98.97%**
- Metrics for 'Anomaly' class:
 - Precision: 97.98%
 - Recall: 100.00%
 - F1-Score: 98.98%
- **Insight:** The near-perfect score confirms the model is excellent at detecting simple statistical deviations but is not a true measure of its ability to detect structured attacks.



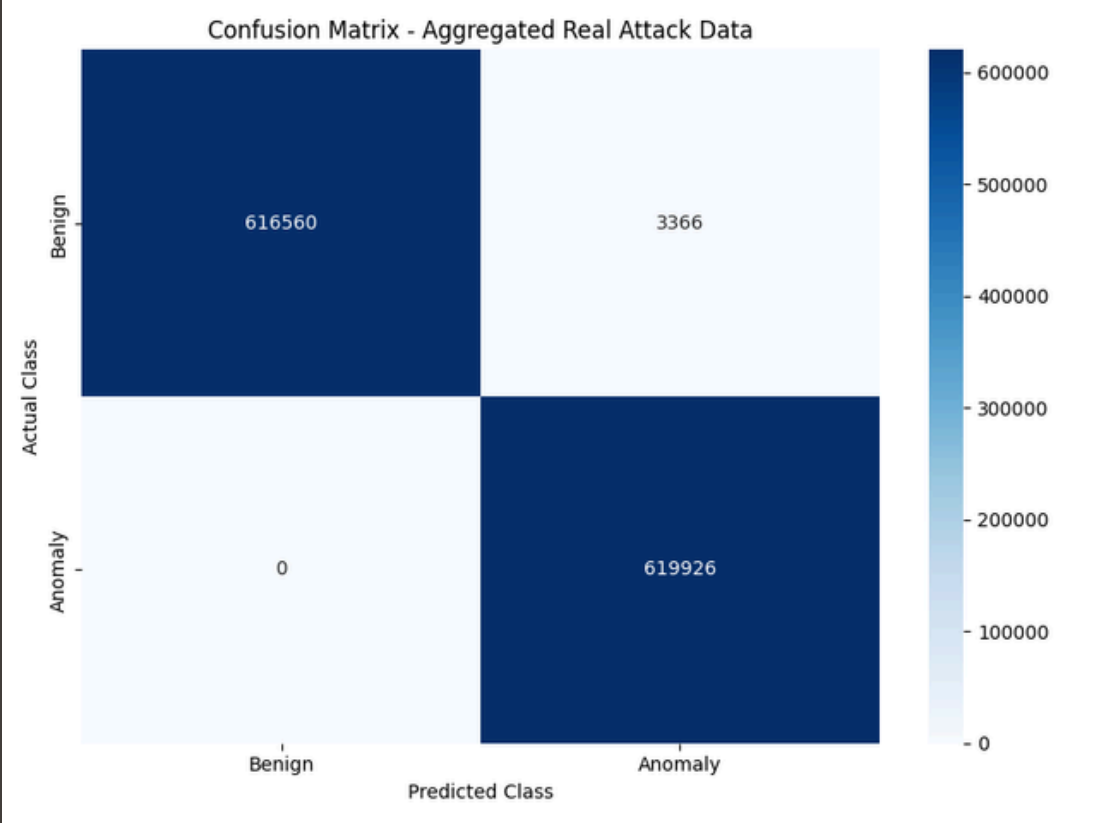
	Precision	Recall	F1 Score	Support
Benign	1.00	0.98	0.99	10000
Anomaly	0.98	1.00	0.99	10000
Accuracy			0.99	20000
Macro Avg.	0.99	0.99	0.99	20000
Weighted Avg.	0.99	0.99	0.99	20000

CLASSIFICATION REPORT

EVALUATION

Test 2: Combined Real Attack Data This test used a combined dataset containing samples from all 12 real attack types found in the dataset.

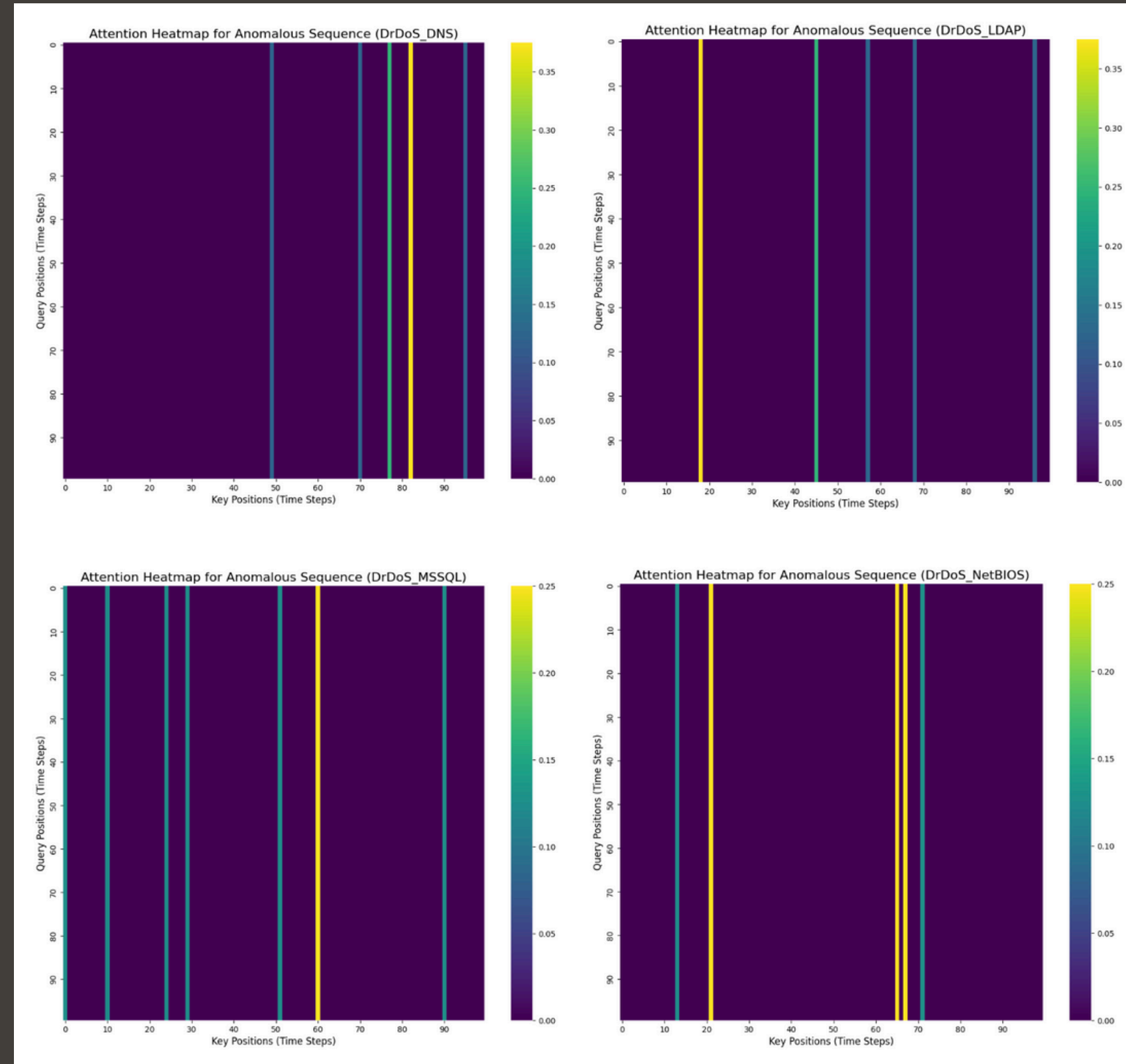
- **Overall Accuracy: 99.73%**
- **Metrics for 'Anomaly' class:**
 - **Precision: 99.46%**
 - **Recall: 100.00%**
 - **F1-Score: 99.73%**



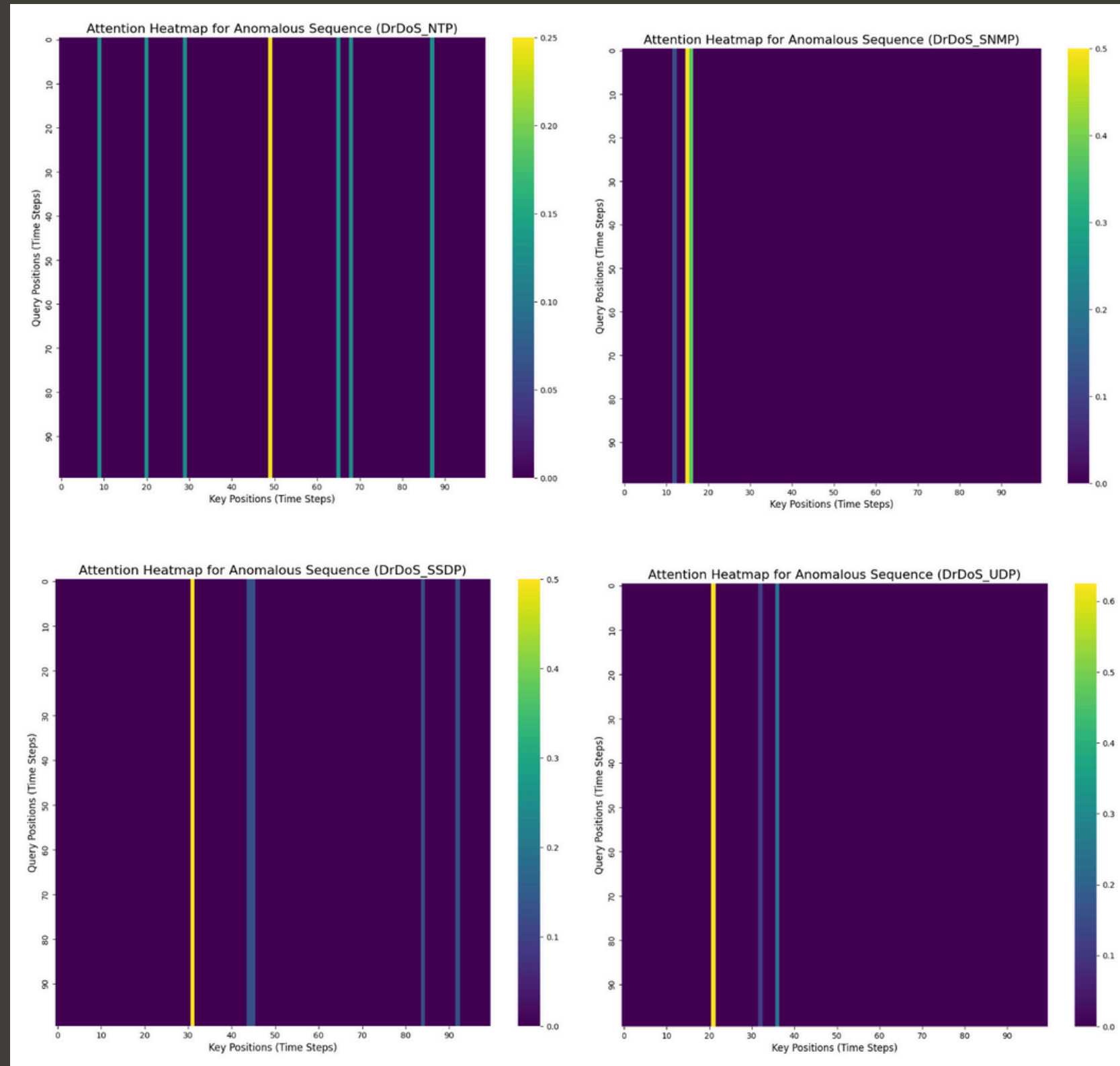
	Precision	Recall	F1 Score	Support
Benign	1.00	0.99	1.00	619926
Anomaly	0.99	1.00	1.00	619926
Accuracy			1.00	1239852
Macro Avg.	1.00	1.00	1.00	1239852
Weighted Avg.	1.00	1.00	1.00	1239852

CLASSIFICATION REPORT

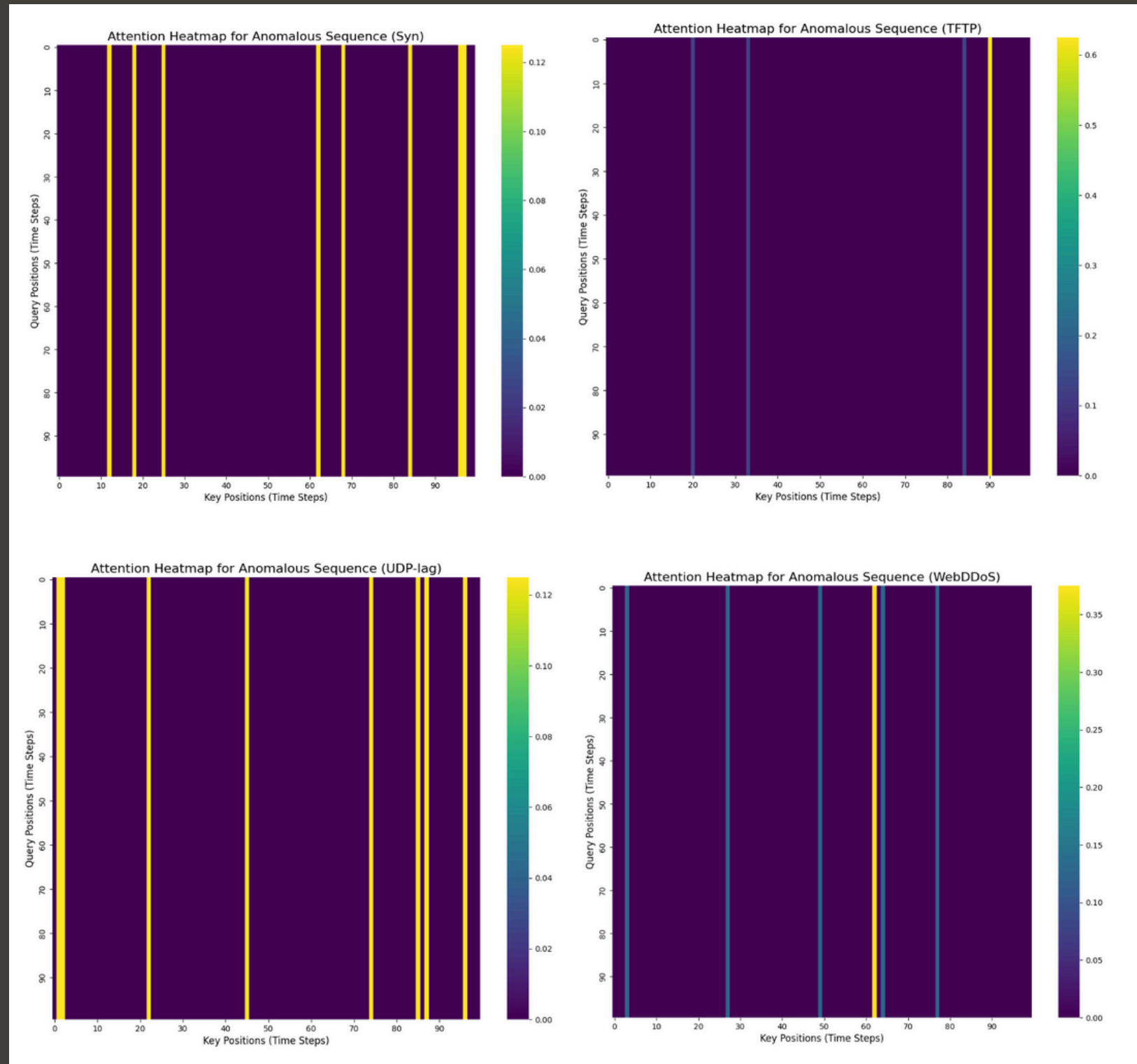
EVALUATION - ATTACK HEATMAPS



EVALUATION - ATTACK HEATMAPS



EVALUATION - ATTACK HEATMAPS





ADVANTAGES & APPLICATIONS

Key Advantages:

- **Handles Zero-Day Attacks with Unsupervised Learning**
- **Scalable to Real-World Data**
- **Explainable: Attention heatmaps show decision rationale**
- **Extensible: Support for multi-scale attention and VAE uncertainty modeling**
- **Flexible: Plug-and-play decoder options (Transformer/GRU)**

Applications:

- **Network intrusion detection**
- **IoT sensor anomaly detection**
- **Financial fraud detection**
- **Industrial equipment monitoring**



COMPARISON TO SOTA

Feature	Our Model (Transformer Autoencoder)	SOTA Model (Anomaly Transformer)
Core Mechanics	Learns to reconstruct normal (benign) time-series data. High reconstruction error indicates an anomaly.	Learns prior associations and series associations in time-series data to calculate an "association discrepancy."
Primary Goal	Effective detection AND practical explainability for security operations.	Highest possible detection accuracy on academic benchmarks.
Time Series Data Handling	Yes (Native). The entire architecture is built around learning temporal patterns in sequences.	Yes (Native). Specifically designed for time-series data.
Explainability AI (XAI)	Key Advantage: Integrated. Generates attention heatmaps to show why a sequence is anomalous.	Limited. While attention weights can be inspected, it's not a primary feature.
Scalability	Key Advantage: Proven. The data pipeline is explicitly designed to be memory-safe for massive datasets.	High. Transformers are generally scalable but the implementation may not be optimized for out-of-core data.
Novelty/ Key Advantages	Practicality & Insight. Combines high accuracy with a robust data pipeline and actionable XAI results.	Theoretical Performance. A novel "association discrepancy" mechanism designed to push accuracy limits.
FI-Score	99.73%	94.96%



INSIGHTS & LIMITATIONS

- **Insights:**

- **Realistic Testing is Crucial:** The high accuracy of $>99\%$ on real attack data demonstrates that simple noise injection is an inadequate proxy for real-world threats.
- **Scalability Solved:** The chunk-based data processing pipeline is a critical contribution, making it feasible to work with datasets that far exceed available RAM.
- **XAI Provides Value:** The attention heatmaps successfully provide a first step towards understanding the model's reasoning, moving it from a "black box" to an interpretable security tool.



- **Limitations:**

- **Static Threshold:** The fixed anomaly threshold may not adapt well to natural, long-term shifts in network behavior (concept drift).
- **Computational Cost:** The one-time caching of attack data is still a time-consuming (though memory-safe) process.



CONCLUSION & FUTURE WORK



1. **Summary:** We successfully built a highly accurate, unsupervised anomaly detection system and, just as importantly, developed a scalable data pipeline that makes working with massive security datasets feasible.
 2. **Key Takeaway:** Our work highlights that realistic evaluation is critical for building trustworthy security models.
 3. **Future Work:**
 - a. **Dynamic Thresholding:** Adapt the anomaly threshold to changing network conditions.
 - b. **Attack Classification:** Extend the model to identify the type of attack.
 - c. **Real-Time Deployment:** Integrate the system for live network monitoring.
- 
- 



CITATIONS



Papers:

- Xu, J., et al. (2022). "Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy." International Conference on Learning Representations (ICLR).
- Vaswani, A., et al. (2017). "Attention Is All You Need." *NeurIPS*.

Datasets:

- Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2019). "Intrusion Detection in the Era of Big Data: A CIC-DDoS2019 Dataset." Canadian Institute for Cybersecurity (CIC). Retrieved from <https://www.unb.ca/cic/datasets/ddos-2019.html>.

Tools & Libraries:

- Paszke, A., et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." Advances in Neural Information Processing Systems.
- The pandas development team. (2020). "pandas-dev/pandas: Pandas." Zenodo.
- Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research.



THANK
YOU

