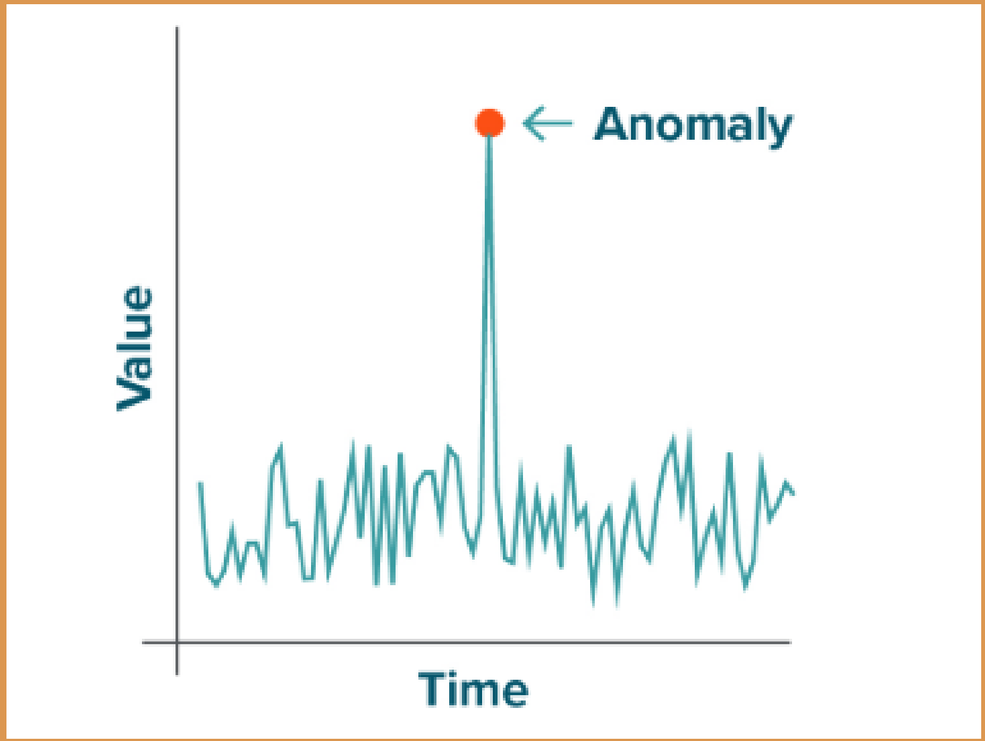
UNSUPERVISED ANOMALY DETECTION ON NASA BEARING DATA USING AUTOENCODERS



DATASET DETAILS

- Source: NASA IMS Bearing Dataset (Set 1)
- Sampling Rate: 20,480 samples per second per file
- Channels: 8 (4 bearings × 2 directions)
- Preprocessing:
  - Normalized all values between 0 and 1
  - Applied window slicing (100-sample windows)
  - Extracted over 3.5 million windows

TOOLS USED

Tools Used

- Python – scripting and data analysis
- TensorFlow / Keras – building and training the autoencoder
- NumPy, Pandas – numerical computations and data wrangling
- scikit-learn – evaluation metrics, t-SNE, ROC analysis
- Matplotlib, Seaborn – data visualization
- Streamlit – UI dashboard (planned, optional)
- Hugging Face Transformers (BLIP-2) – captioning plots for human-readable explanations

MODEL ARCHITECTURE: 1D AUTOENCODER

A symmetric autoencoder was used:

Input(100) → Dense(64) → Dense(32) → Dense(16) # Encoder

Input(100) → Dense(32) → Dense(64) → Output(100) # Decoder

It learns to compress each 100-sample vibration window into a 16-dimensional latent vector and reconstruct the original signal. A high reconstruction error indicates unusual input.

TRAINING STRATEGY

- No labels used during training (purely unsupervised)
- Trained to minimize Mean Squared Error (MSE)
- Used early stopping and batch training for speed

ANOMALY DETECTION LOGIC

- Predict reconstruction on all input windows
- Compute reconstruction error: MSE = mean((original - reconstructed)^2)
- Set threshold = 99th percentile of training errors
- Flag any window above the threshold as an anomaly
- This technique works because the autoencoder only learns normal behavior and fails to reconstruct unseen faulty patterns.

EVALUATION AND METRICS

For evaluation only (not training), labeled data was used.
A balanced dataset of 25,000 normal and 25,000 faulty samples was selected.

- Precision: % of predicted anomalies that are actually faulty
- Recall: % of real faults correctly detected
- F1-Score: Harmonic mean of precision and recall
- Confusion Matrix: Visual breakdown of TP, FP, TN, FN
- ROC Curve: Plots TPR vs. FPR across thresholds

Loaded shape: (3531528, 100)
Label shape : (3531528,)
Unique labels: (array([0, 1]), array([1310400, 2221128]))

Anomaly detection threshold: 0.65157
Total Anomalies detected: 30000 out of 50000

```
 Classification Metrics
Precision : 0.6304 – % of predicted anomalies that are truly faulty
Recall    : 0.6014 – % of actual faults correctly identified
F1-Score  : 0.6155   – Harmonic mean of precision and recall
Accuracy  : 0.5275 – Overall correct predictions (both normal and faulty)

Detailed Classification Report:
              precision    recall  f1-score   support

           0     0.3732    0.4023    0.3872    262080
           1     0.6304    0.6014    0.6155    444225

    accuracy                         0.5275    706305
   macro avg     0.5018    0.5019    0.5014    706305
weighted avg     0.5350    0.5275    0.5308    706305
```

Model Evaluation Summary

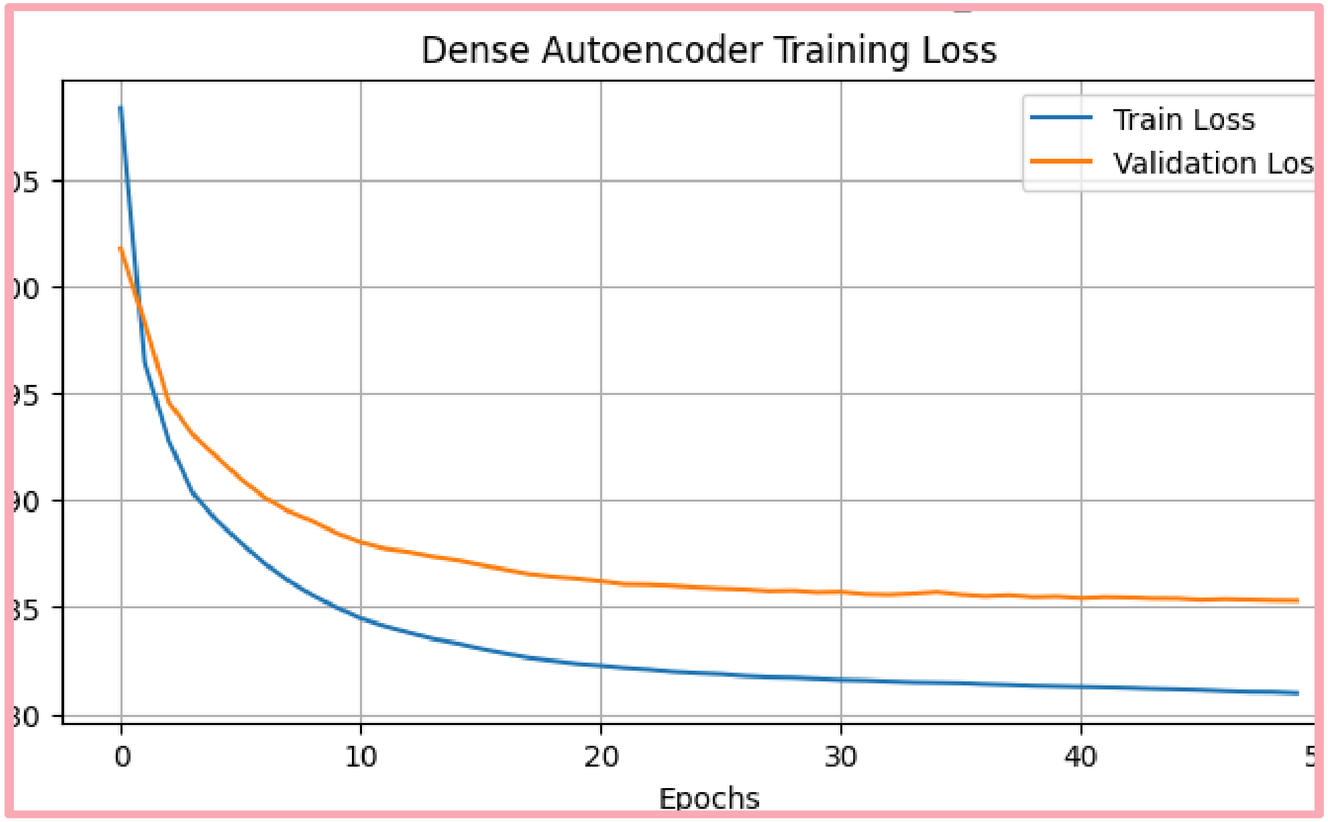
Input shape: 3.5 million signal windows (each of length 100)
Labels: 0 = Normal, 1 = Faulty
Threshold used: 0.65157
Anomalies detected: 30,000 out of 50,000 predictions

Performance Metrics:

Precision: 63% → Most predicted anomalies were truly faulty
Recall: 60% → Model found 60% of real faults
F1-Score: 61.5% → Balanced measure of accuracy
Accuracy: 52.8% → Overall prediction correctness

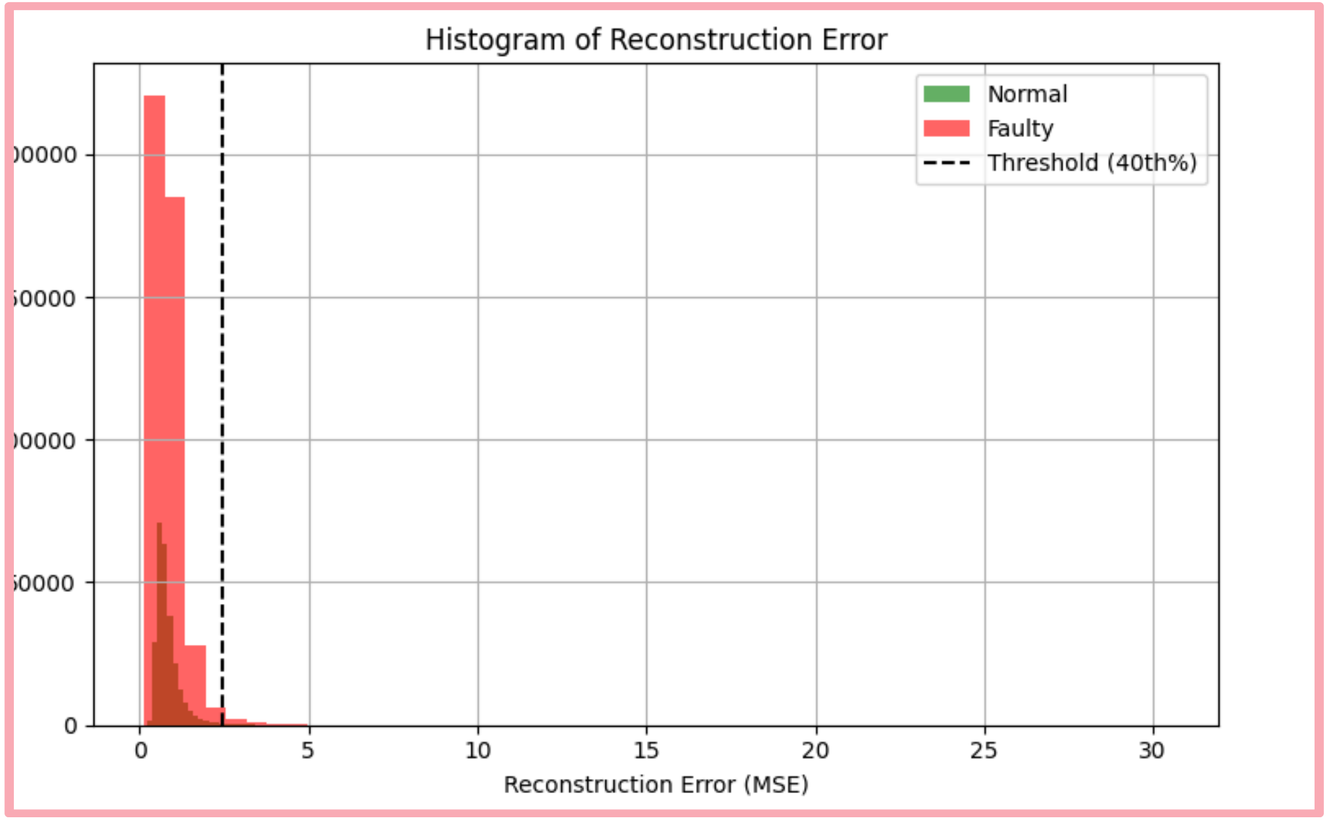
Co.

VISUALIZATION TECHNIQUES



- This plot shows the training and validation loss of the autoencoder.
- Both curves decrease smoothly and flatten, which means the model learned well without overfitting

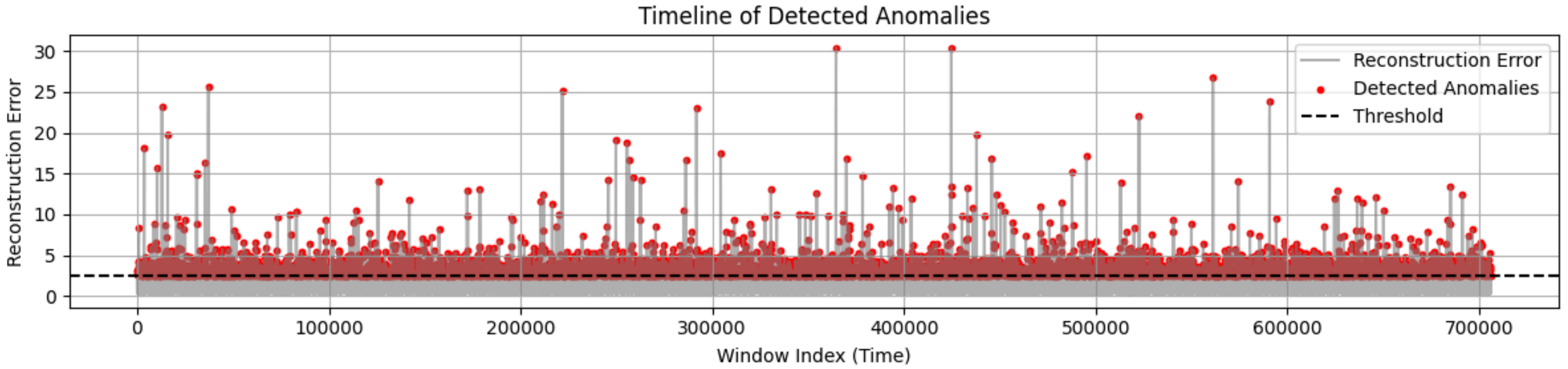
VISUALIZATION TECHNIQUES



Histogram of Reconstruction Errors

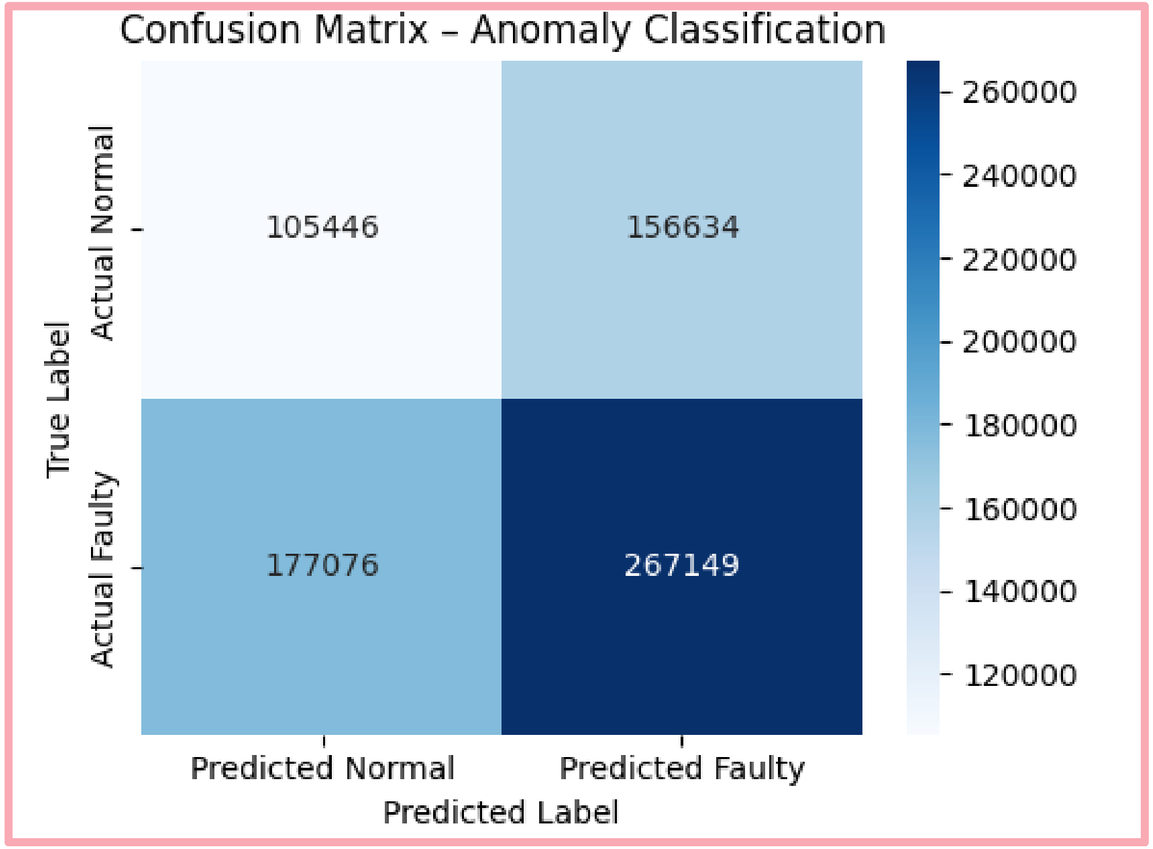
- This histogram shows how the reconstruction errors are distributed for normal (green) and faulty (red) samples.
- The X-axis represents the reconstruction error (Mean Squared Error), and the Y-axis indicates the number of samples (frequency).
- Faulty samples generally produce higher reconstruction errors compared to normal ones.
- The dashed line marks the anomaly detection threshold, set at the 40th percentile of the error distribution.
- While there is some overlap between normal and faulty distributions, the longer right tail in the faulty data suggests that reconstruction error is a meaningful signal for separating normal and abnormal behavior.

VISUALIZATION TECHNIQUES



Timeline of Detected Anomalies

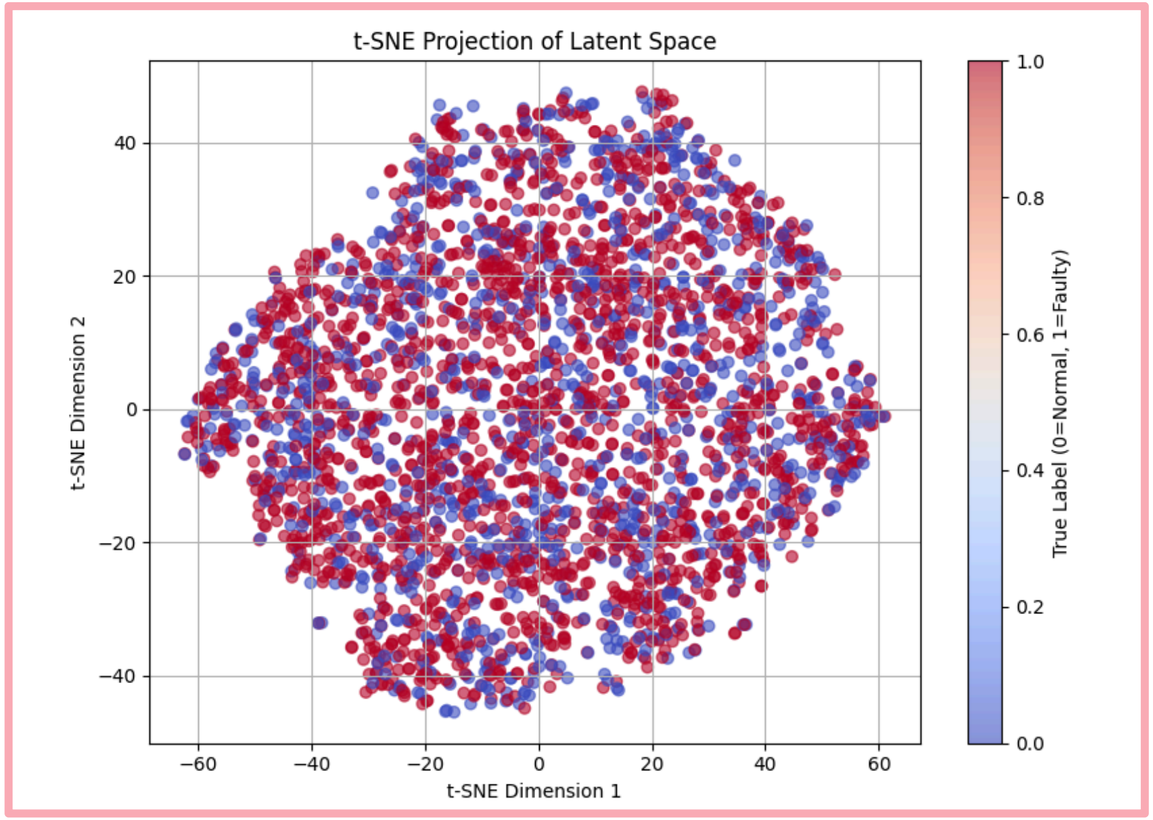
VISUALIZATION TECHNIQUES



Confusion Matrix Heatmap – Clearly shows model misclassifications

- Evaluates model's binary classification performance
- True Positives (bottom-right): 267,149 faulty samples correctly identified
- True Negatives (top-left): 105,446 normal samples correctly classified
- False Positives: 156,634 normal samples misclassified as faulty
- False Negatives: 177,076 faulty samples missed by the model
- Indicates room for improvement in recall and precision

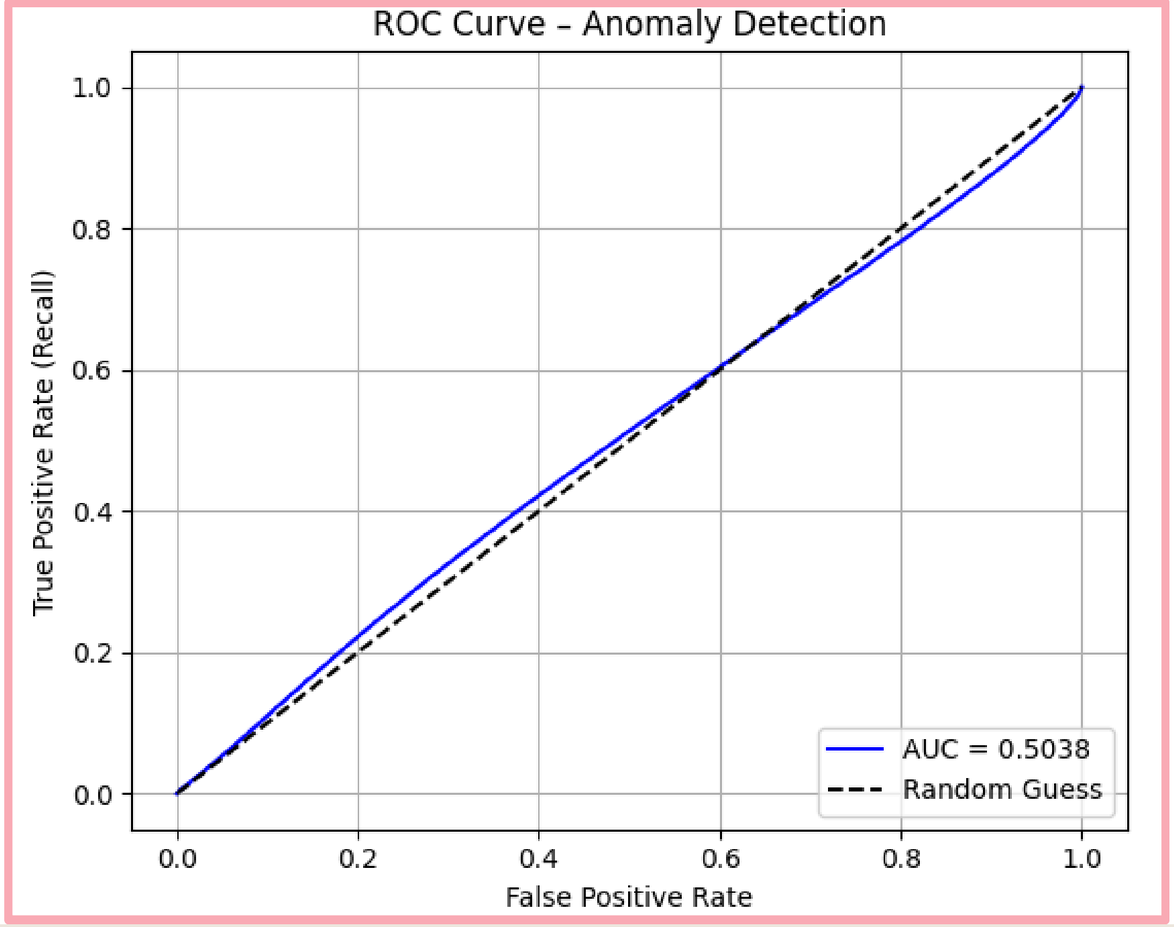
VISUALIZATION TECHNIQUES



Latent Space (t-SNE) – Clusters formed in compressed space, showing some separation between normal/faulty behavior

- Each dot = one input window, colored by true label
- Blue = normal, Red = faulty
- Shows some clustering but also overlap
- Helps understand how well the model separates anomalies

VISUALIZATION TECHNIQUES



ROC Curve

- ROC curve shows trade-off between true positive rate (recall) and false positive rate
- Blue line = model's performance
- AUC = 0.5038, close to random guessing (0.5)
- Indicates model is not effectively separating normal vs faulty samples
- May need threshold tuning or model retraining

MULTIMODAL INTEGRATION (LLMS)

- To enhance explanation quality, I passed the generated plots into BLIP-2 (a vision-language model) from Hugging Face.
- Prompt example:
- "This is a time-series anomaly detection plot showing reconstruction errors over time. Red dots indicate detected anomalies. Summarize what this plot reveals."
- Response:
- "The system remained stable until mid-sequence. Frequent spikes in reconstruction error suggest abnormal behavior toward the end, likely indicating system failure."
-

FUTURE WORK – MLOPS & DEPLOYMENT

To move this closer to a real-world application:

- Streamlit UI: A dashboard to upload CSV files, visualize results, and export reports
- Docker: Containerization for consistent deployment; can push to DockerHub
- REST API with FastAPI: Accept raw signal → return anomaly flags
- MLflow or TensorBoard: Track training, thresholds, and metrics over time
- Cloud Options: EKS, ECS, or even Sagemaker (optional, complex)
- Auto-Retraining: If new normal data drifts, retrain autoencoder automatically

CHALLENGES FACED

- Data Imbalance: The original dataset had a huge number of normal samples and very few faulty ones, which made evaluation tricky. I had to create a synthetic balance for fair testing.
- Latency: Using tools like t-SNE and BLIP-2 on Google Colab caused delays due to high memory and compute demands.
- Session Crashes: Some notebook sessions crashed, especially when training models or running heavy visualizations.
- Interpretability: Understanding and trusting the results wasn't easy—so I had to rely on multiple visualizations like reconstruction errors, ROC curves, and confusion matrices.
- Thresholding: Setting the anomaly detection threshold wasn't straightforward; I had to try different percentiles and observe the impact.

CONCLUSION

- This project showed how autoencoders can help detect anomalies without needing labeled data.
- I used the NASA bearing dataset and focused on modeling normal machine behavior.
- Evaluation on a balanced set gave mixed results and good at identifying normal data, but recall for faults needs improvement.
- This setup is a realistic foundation for predictive maintenance in industry.
- With better thresholding and more features, it could be improved and deployed for real-time monitoring.

Thank

You