Generate Synthetic Time Series Data for Anomaly Detection

Team Contribution

• Om Prakash Tripathi	- Data Collection and Documentation
• Saket	- EDA and Preprocessing
• Fareeha	- Data Generation and pipeline Maintenance
• Afrin	- Model Training and Evaluation
• Richa and Akash	- UI development (Streamlit)

1. Introduction

Project Overview

Our AIML internship final project titled “Generate Synthetic Time Series Data for Anomaly Detection.”

1 **Domain**

Artificial Intelligence and Machine Learning (AIML)

2 **Team Size**

6 members

3 **Goal**

To build a model that generates synthetic time series data for anomaly detection when real data is limited.

2. Problem Statement

Industrial systems like water pumps often fail due to rare operational anomalies.

Real anomaly data is scarce, making supervised learning models less effective.

The goal is to generate synthetic time-series data that mimics real anomalies to train robust detection models.

3. Objective

To leverage Generative Adversarial Networks (GANs) (specifically TimeGAN) to:

1. Synthesize realistic sensor time-series data.

2. Improve machine learning models for anomaly detection.

3. Enable predictive maintenance in IoT environments.

4. Project Architecture

Modules Overview

- Data Collection & Preprocessing
- Exploratory Data Analysis (EDA)
- TimeGAN Model Training
- Synthetic Data Generation
- Model Evaluation & Comparison

📊 Flow:
Raw Sensor Data → Data Preprocessing → TimeGAN Training → Synthetic Data → Classifier Training → Anomaly Detection

5. Data Preprocessing

Modules Overview

- Dataset: Water Pump Sensor Time-Series Data (Kaggle / IoT Source)
- Steps:
  - Handle missing/null values
  - Identify numerical sensor features
  - Normalize data using MinMaxScaler
  - Convert data into overlapping sequences for TimeGAN
- Example: Sequence length = 24 → Windows of 24 timestamps

6. Exploratory Data Analysis (EDA)

Visualized sensor behavior using:

- Line plots for time-based patterns
- Correlation heatmaps
- Distribution plots

Found differences between normal and faulty operational states.

7. Model Used (Timing and Time-Grain)

1

Combines advantages of GANs + RNNs for sequential data.

2

Architecture Components:

- Embedding Network
- Generator
- Discriminator
- Supervisor Network

3

Loss Functions:

- Supervised + Unsupervised adversarial loss

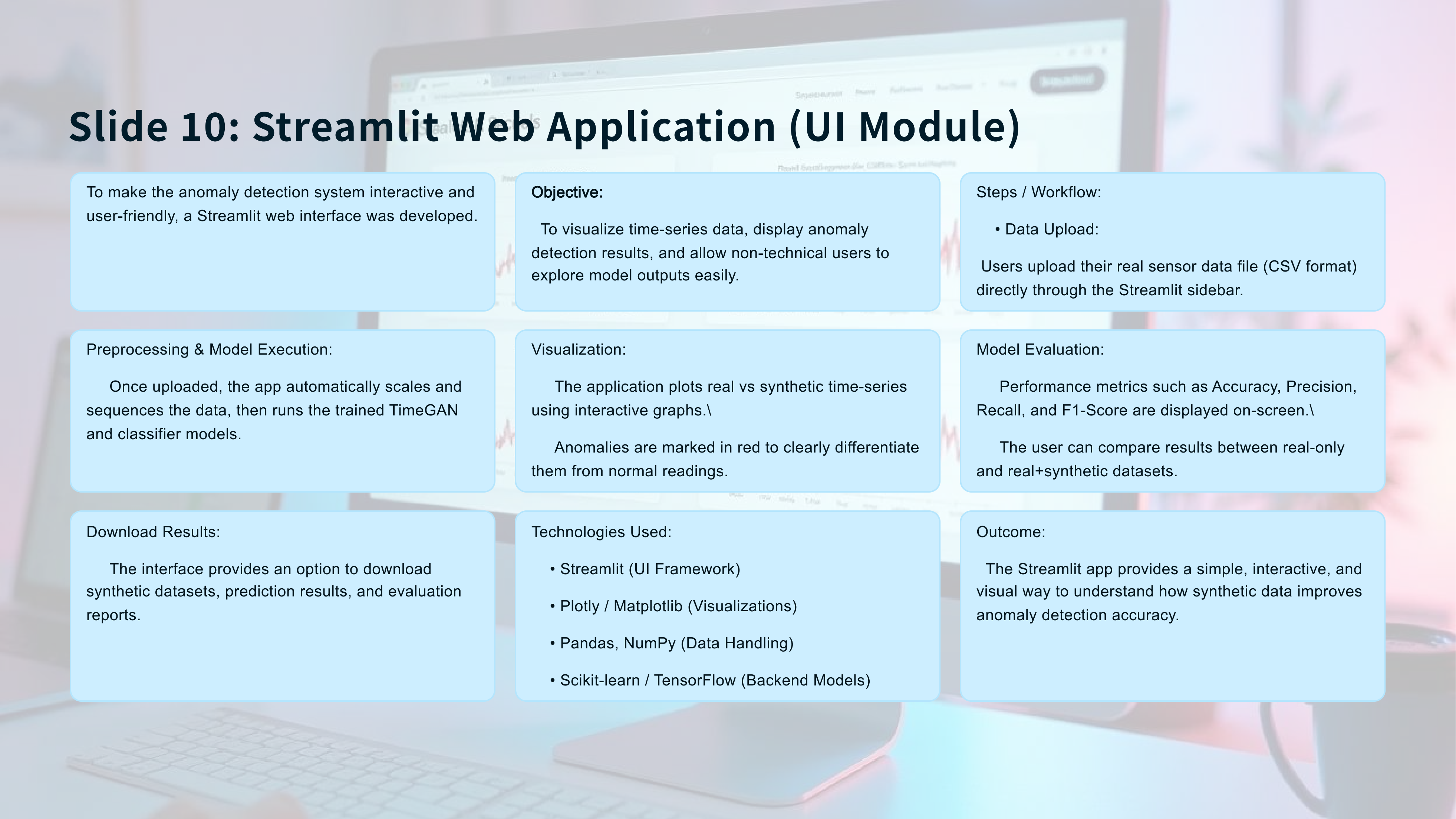
8. Synthetic Data Generation

Generated artificial sequences mimicking normal and anomaly states.

Compared statistical similarity with real data using:

- PCA visualization
- Dynamic Time Warping (DTW) distance

Result: Synthetic data closely matched real sensor patterns..



Slide 10: Streamlit Web Application (UI Module)

To make the anomaly detection system interactive and user-friendly, a Streamlit web interface was developed.

Objective:

To visualize time-series data, display anomaly detection results, and allow non-technical users to explore model outputs easily.

Steps / Workflow:

- Data Upload:

Users upload their real sensor data file (CSV format) directly through the Streamlit sidebar.

Preprocessing & Model Execution:

Once uploaded, the app automatically scales and sequences the data, then runs the trained TimeGAN and classifier models.

Visualization:

The application plots real vs synthetic time-series using interactive graphs.\

Anomalies are marked in red to clearly differentiate them from normal readings.

Model Evaluation:

Performance metrics such as Accuracy, Precision, Recall, and F1-Score are displayed on-screen.\

The user can compare results between real-only and real+synthetic datasets.

Download Results:

The interface provides an option to download synthetic datasets, prediction results, and evaluation reports.

Technologies Used:

- Streamlit (UI Framework)
- Plotly / Matplotlib (Visualizations)
- Pandas, NumPy (Data Handling)
- Scikit-learn / TensorFlow (Backend Models)

Outcome:

The Streamlit app provides a simple, interactive, and visual way to understand how synthetic data improves anomaly detection accuracy.

Slide 11: Application Screens & Examples

Below are key sections of our Streamlit-based web interface demonstrating project functionality and visual outputs:

Slide 11: Application Screens & Examples

🖥️ Figure 1: Dashboard / Home Page

• Displays the project title, brief description, and a sidebar for uploading sensor datasets.

• The sidebar also contains input options such as sequence length, latent dimension, and number of epochs.

• After data upload, users can start model execution with a single click.

📊 Figure 2: Real vs Synthetic Time-Series Visualization

• Shows interactive line plots comparing real and synthetic sensor readings.

• Anomalies are automatically highlighted in red for better visibility.

• This visualization helps verify that synthetic sequences resemble realistic data patterns.

📈 Figure 3: Model Evaluation & Metrics Panel

• Displays Accuracy, Precision, Recall, and F1-score for both Real and Real + Synthetic datasets.

• Includes confusion matrix and ROC curve plots to visually assess model performance.

• Clearly shows how synthetic data improves detection of rare anomalies.

💾 Figure 4: Download Section

• Provides buttons to export results, including generated synthetic data, performance metrics, and plots.

• Ensures users can reuse or share model outputs for further analysis or reporting.

Summary

The Streamlit application provides a complete visual workflow — from uploading raw sensor data to generating synthetic samples, visualizing anomalies, evaluating models, and exporting results.
This interface transforms complex backend ML operations into an intuitive, single-click experience for end users.

12. Results

Metric	Real Data	Real + Synthetic
Accuracy	85%	91%
Recall (Anomaly)	72%	88%
F1-Score	78%	89%

Visualization

Confusion Matrix & ROC Curve

Conclusion

Synthetic data significantly improved anomaly detection accuracy.

13. Challenges Faced

1

Tuning GAN hyperparameters (sequence length, latent size).

2

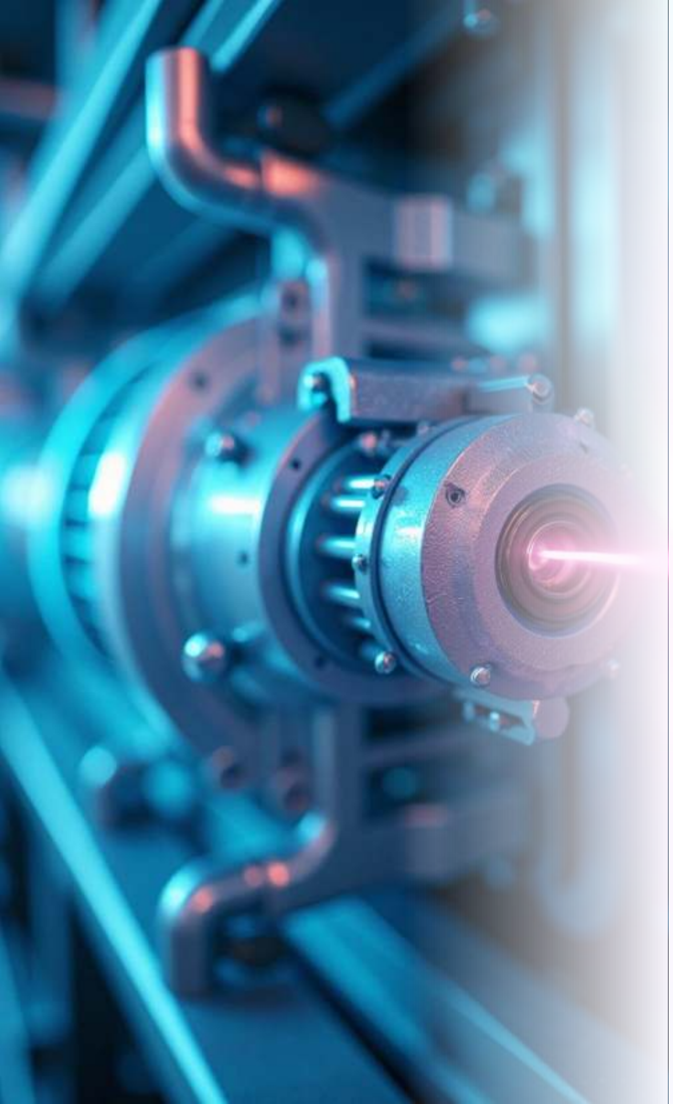
Preventing mode collapse in GAN training.

3

Balancing between realistic and diverse synthetic samples.

4

Computational constraints during model training.



14. Applications

- Predictive maintenance in industries
- IoT anomaly detection
- Financial fraud detection
- Healthcare time-series monitoring

15. Future Enhancements

1. Integrate Variational Autoencoders (VAEs) for hybrid generation.
2. Use transformer-based models (TimeGPT).
3. Deploy model as a web dashboard for real-time anomaly alerts.

16. Conclusion Successfully generated synthetic time-series data using TimeGAN.

Improved anomaly detection models for rare events.

Demonstrated the power of synthetic data augmentation in AI-driven maintenance.

Thank You