**Preliminary Report: Anomaly Detection in IoT Sensor Data using Autoencoders**

**1. Introduction**

**1.1 Problem Statement**

The rapid adoption of IoT devices in industries, healthcare, and smart cities has led to the generation of vast amounts of sensor data. However, detecting anomalies in real-time is challenging due to data complexity, noise, and volume. Traditional rule-based detection methods are inefficient in identifying unseen patterns. This project aims to leverage **Autoencoders**, a deep learning model, to detect anomalies in IoT sensor data by learning normal behavior and identifying deviations.

**1.2 Objectives**

* Develop an **Autoencoder-based anomaly detection system**.
* Improve the accuracy of detecting **faults, cyber-attacks, and sensor failures**.
* Implement a **real-time detection pipeline** with IoT sensor logs.

**2. Dataset Details**

**2.1 Data Source**

* Dataset used: **AnoML-IoT Dataset** from Kaggle.
* Contains time-series sensor data with labeled anomalies.

**2.2 Data Characteristics**

* **Format:** CSV.
* **Sensor types:** Temperature, humidity, voltage, and other industrial sensor readings.
* **Anomalies:** Sudden spikes, missing values, drifts, outliers.

**2.3 Preprocessing**

* **Handling missing values:** Forward fill, interpolation.
* **Feature Scaling:** Min-max normalization.
* **Data Augmentation:** Synthetic anomaly generation.
* **Splitting:** 80% training, 10% validation, 10% testing.

**3. Model Development**

**3.1 Model Selection**

* **Autoencoder:** An unsupervised neural network that learns to reconstruct normal sensor data.
* **Reconstruction Error:** Used to classify normal vs. anomalous patterns.

**3.2 Network Architecture**

* **Input Layer:** Multivariate sensor time-series.
* **Hidden Layers:** Dense/Conv1D for feature extraction.
* **Bottleneck Layer:** Compressed latent representation.
* **Decoder:** Reconstructs input data.
* **Loss Function:** Mean Squared Error (MSE) for anomaly detection.

**4. Implementation**

**4.1 Tools & Technologies**

* **Python**: TensorFlow/Keras or PyTorch.
* **Data Processing**: Pandas, NumPy, Matplotlib.
* **Jupyter Notebook/Google Colab** for experimentation.

**4.2 Training Configuration**

* **Optimizer:** Adam
* **Loss Function:** MSE
* **Epochs:** 50-100
* **Batch Size:** 32

**5. Model Evaluation**

**5.1 Performance Metrics**

* **Reconstruction Error Threshold:** To detect anomalies.
* **Precision & Recall:** Measures detection performance.
* **AUC-ROC Curve:** Evaluates anomaly classification.

**5.2 Expected Results**

* Threshold tuning for optimal anomaly detection.
* **90-95% accuracy** in identifying anomalies.

**6. Deployment Strategy**

**6.1 Deployment Pipeline**

* **Model Export:** Convert to TensorFlow Lite or ONNX.
* **Edge Deployment:** Raspberry Pi, NVIDIA Jetson Nano.
* **API Development:** Flask/FastAPI for real-time monitoring.

**6.2 Real-Time Application**

* **Integration with IoT Dashboards** (Grafana, AWS IoT).
* **Alert Mechanisms:** Email, SMS, MQTT notifications.

**7. Challenges & Future Work**

**7.1 Challenges**

* **Imbalanced data** (few anomalies in large datasets).
* **Concept drift** (changing sensor behaviour over time).

**7.2 Future Enhancements**

* **Self-learning AI** to adapt to new anomalies.
* **GANs** for synthetic anomaly generation.

**8. Conclusion**

This preliminary report outlines a structured approach to developing an anomaly detection system using Autoencoders. The project aims to enhance IoT security and efficiency by accurately identifying abnormal sensor readings. Future work will focus on improving robustness and real-time efficiency.