CubeSat Fault Prediction System
Complete Documentation

Development Team

October 2024

Abstract

This project implements an AI-powered fault prediction system for CubeSat satellites using machine learning. The system can predict 5 different operational states (normal + 4 fault types) by analyzing real-time telemetry data from 8 critical sensors. The final model is optimized for embedded deployment at only 3.3KB, making it suitable for space-grade hardware with limited computational resources.

Contents

1 Executive Summary

The CubeSat Fault Prediction System addresses the critical need for proactive maintenance in small satellite missions. By leveraging machine learning and real-time telemetry analysis, this system provides early warning capabilities for potential system failures, significantly improving mission reliability and success rates.

1.1 Key Features

- **Real-time Monitoring**: Continuous analysis of 8 sensor parameters
- **Early Warning System**: Predicts faults before they become critical
- **Embedded Deployment**: Ultra-lightweight model (3.3KB) for space-grade hardware
- **High Accuracy**: ¿95% validation accuracy on synthetic data

2 Project Overview

2.1 Problem Statement

CubeSats are small, cost-effective satellites that are prone to unexpected failures in the harsh space environment. Traditional reactive maintenance approaches result in mission loss when critical systems fail. This project provides predictive maintenance capabilities to identify potential failures before they occur.

2.2 Solution Approach

- **Machine Learning Model**: Neural network trained on telemetry patterns
- **Real-time Monitoring**: Continuous analysis of 8 sensor parameters
- **Early Warning System**: Predicts faults before they become critical
- **Embedded Deployment**: Ultra-lightweight model (3.3KB) for space-grade hardware

3 Technical Architecture

3.1 System Components

3.1.1 On-Board Computer (OBC) Monitoring

- **Bus Voltage**: Power system stability (4.9-5.1V normal range)
- **Current Draw**: Power consumption patterns (0.4-0.6A normal)
- **Power Consumption**: Overall electrical load (2.0-3.0W normal)
- **MCU Core Temperature**: Processor thermal state (20-50°C normal)
- **Heartbeat Signal**: System alive indicator (1=alive, 0=dead)

3.1.2 Telemetry, Tracking & Command (TTC) Monitoring

- **UART Packets Received**: Communication throughput (95-99 packets normal)
- **CRC Error Count**: Data integrity failures (0-1 errors normal)
- **UART Timeout**: Communication system health (0=healthy, 1=timeout)

3.2 Fault Classification System

Class	Fault Type	Characteristics
0	Normal Operation	All parameters within specification ranges
1	Minor Fault	Intermittent issues, occasional communication glitches
2	Power System Fault	High power consumption, battery degradation, solar panel issues
3	Communication Fault	TTC system problems, antenna pointing, radio failures
4	Data Integrity Fault	High error rates, radiation-induced bit flips

Table 1: Fault Classification System

4 Machine Learning Implementation

4.1 Model Architecture

```
Input Layer (8 features) → Hidden Layer 1 (64 neurons, ReLU)
→ Hidden Layer 2 (32 neurons, ReLU) → Output Layer (5 classes, Softmax)
```

4.2 Training Specifications

- **Algorithm**: Sequential Neural Network (TensorFlow/Keras)
- **Loss Function**: Sparse Categorical Crossentropy
- **Optimizer**: Adam (learning_rate=0.001)
- **Training Data**: 10,000 synthetic telemetry samples
- **Validation Split**: 20%
- **Epochs**: 50 with early stopping

4.3 Performance Metrics

- **Accuracy**: ¿95% on validation data
- **Model Size**: 3.3KB (TensorFlow Lite format)
- **Inference Time**: ¡1ms on embedded hardware
- **Memory Usage**: ¡50KB RAM during inference

5 Data Analysis & Insights

5.1 Sensor Distribution Analysis

5.1.1 Normal Operation Ranges

- **Bus Voltage**: 4.95-5.05V (tight tolerance)
- **Current Draw**: 0.4-0.6A (varies with activity)

- **Temperature**: 20-50°C (thermal cycling)
- **Packet Rate**: 95-99/period (high reliability)

5.1.2 Fault Signatures

- **Power Faults**: Current ¿0.7A, Power ¿3.5W
- **Communication Faults**: Packets ¡50, Timeouts present
- **Integrity Faults**: CRC errors ¿5, data corruption

5.2 Correlation Patterns

- **Strong Correlations**: Current Draw Power Consumption (r=0.98)
- **Moderate Correlations**: Temperature Power Consumption (r=0.65)

6 Visualization Capabilities

6.1 Dynamic Analysis Tools

1. **Sensor Distribution Analysis**

   ```
   python visualization/visualize_data.py --distributions
   ```

2. **Fault Pattern Analysis**

   ```
   python visualization/visualize_data.py --faults
   ```

3. **Correlation Matrix**

   ```
   python visualization/visualize_data.py --correlation
   ```

4. **Time Series Simulation**

   ```
   python visualization/visualize_data.py --timeseries
   ```

7 Deployment & Integration

7.1 Hardware Requirements

- **Minimum**: STM32F4 series microcontroller
- **RAM**: 64KB minimum (50KB for model + buffers)
- **Flash**: 32KB for model storage
- **Processing**: ARM Cortex-M4 @ 168MHz

7.2 Software Dependencies

```
tensorflow-lite-micro  # Inference engine
numpy>=1.21.0         # Data processing
pandas>=1.3.0         # Data manipulation
matplotlib>=3.3.0     # Visualization
seaborn>=0.11.0       # Statistical plots
scikit-learn>=1.0.0   # ML utilities
```

7.3 Integration Steps

1. **Data Collection**: Implement telemetry data pipeline
2. **Model Loading**: Load fault_model.tflite into embedded system
3. **Preprocessing**: Normalize sensor inputs to training ranges
4. **Inference**: Run prediction every 1-10 seconds
5. **Action**: Trigger alerts/actions based on fault predictions

8 Business Impact & Applications

8.1 Mission-Critical Benefits

- **Reduced Mission Loss**: Prevent catastrophic failures through early detection
- **Extended Mission Life**: Proactive maintenance and system optimization
- **Cost Savings**: Avoid expensive satellite replacements
- **Improved Reliability**: Higher mission success rates

8.2 Use Cases

1. **Real-time Health Monitoring**
2. **Predictive Maintenance**
3. **Fleet Management**
4. **Research & Development**

9 Project Structure & Codebase

9.1 Directory Structure

```
cubesat-fault-predictor/
          data/
                cubesat_data.csv          # Training dataset
          models/
                fault_model.tflite        # Deployable ML model (3.3KB)
          src/                          # Core ML pipeline
                generate_data.py          # Synthetic data generation
                train_model.py            # Model training & conversion
                predict.py                # Inference demonstration
          visualization/                # Analysis tools
                visualize_data.py         # Interactive plotting suite
                README.md                     # Visualization documentation
          demo.py                           # Quick project demonstration
          requirements.txt                  # Python dependencies
          README.md                         # Project documentation
```

9.2 Key Files Description

9.2.1 src/generate_data.py

```
# Creates realistic CubeSat telemetry data
# Simulates 5 operational states with appropriate noise
# Generates 10,000 balanced training samples
```

9.2.2 src/train_model.py

```
# Implements neural network architecture
# Trains model with validation monitoring
# Converts to TensorFlow Lite format for deployment
```

10 Getting Started Guide

10.1 Quick Setup (5 minutes)

```
# 1. Clone/navigate to project
cd cubesat-fault-predictor

# 2. Install dependencies
pip install -r requirements.txt

# 3. Run complete demo
python demo.py

# 4. Generate visualizations
python visualization/visualize_data.py
```

10.2 Full Development Workflow

```
# 1. Generate fresh training data
python src/generate_data.py

# 2. Train new model
python src/train_model.py

# 3. Test inference
python src/predict.py

# 4. Analyze results
python visualization/visualize_data.py --faults
```

11 Future Enhancements

11.1 Technical Improvements

- **Multi-satellite Learning**: Train on data from multiple CubeSat missions

- **Online Learning**: Update model based on real flight data
- **Sensor Fusion**: Incorporate additional telemetry sources
- **Uncertainty Quantification**: Provide confidence intervals with predictions

11.2 Operational Features

- **Web Dashboard**: Real-time monitoring interface
- **Alert System**: Automated notifications for predicted faults
- **Historical Analysis**: Long-term trend analysis and reporting
- **Integration APIs**: Connect with existing ground control systems

12 Contact & Support

12.1 Development Team

- **Project Lead**: [Your Name]
- **ML Engineer**: [Team Member]
- **Systems Integration**: [Team Member]

12.2 Technical Specifications

- **Model Version**: 1.0
- **Last Updated**: October 2024
- **Compatibility**: TensorFlow Lite 2.8+
- **Target Hardware**: STM32F4/F7 series