**High-Level Design Document (HLD)**

**Document Version Control**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 2024-12-07 | Naman Bajpai | Initial version of the document. |

**Abstract**

This document outlines the **High-Level Design (HLD)** for the **Turbo Engine RUL Prediction System**, a machine learning-based solution built using **Streamlit** for the user interface and **TensorFlow** for the predictive model. The system predicts the **Remaining Useful Life (RUL)** of a turbo engine based on operational settings and sensor data, providing actionable insights for maintenance scheduling.

**Introduction**

The **Turbo Engine RUL Prediction System** is designed to predict the remaining operational cycles of a turbo engine. By leveraging machine learning, the system can forecast when the engine is likely to fail, allowing for preemptive maintenance and improved operational efficiency.

**Why this High-Level Design Document?**

This document provides an overview of the system architecture, design, and key components involved in building and deploying the **Turbo Engine RUL Prediction System**. It serves as a reference for understanding the design approach, technical requirements, and solution architecture.

**Scope**

The scope of this document is limited to the description of the architecture and high-level design of the **Turbo Engine RUL Prediction System**. It does not cover implementation-level details or specific coding practices.

**Definitions**

* **RUL (Remaining Useful Life)**: The predicted number of cycles remaining before the engine reaches failure.
* **Sensor Data**: Data captured from various sensors embedded in the engine that provide information about operational conditions.
* **Operational Settings**: Configurations related to the engine’s operational parameters (e.g., speed, pressure).

**General Description**

**Product Perspective**

The **Turbo Engine RUL Prediction System** is a predictive maintenance solution aimed at improving the reliability and efficiency of turbo engines. It uses sensor data and operational settings to predict the engine’s remaining useful life.

**Problem Statement**

Turbo engines face a risk of sudden failure, leading to expensive repairs and unplanned downtime. The lack of predictive analytics increases the risk of such failures, which is addressed by this system.

**Proposed Solution**

The proposed solution involves building a system that:

1. Collects real-time operational settings and sensor data from the turbo engine.
2. Using machine learning models to predict the remaining useful life (RUL).
3. Displays the predicted RUL and suggests maintenance schedules to avoid failures.

**Further Improvements**

Future improvements may include integrating real-time data streams from sensors, enhancing the model with additional data features, and improving the model's accuracy with more advanced deep learning techniques and tuning.

**Technical Requirements**

* **Python**: Programming language used for building the system.
* **TensorFlow/Scikit-learn**: For the machine learning model.
* **Streamlit**: For building the web interface.
* **NumPy**: For data preprocessing and reshaping.
* **Pandas**: For handling large datasets and preprocessing.
* **Matplotlib**: For data visualization.

**Data Requirements**

* **Sensor Data**: At least 25 sensors with data over 13 time steps.
* **Operational Settings**: Data for settings like engine speed, pressure, temperature, etc.
* **Training Data**: Historical RUL data to train the machine learning model.

**Tools Used**

* **TensorFlow**: Deep learning framework for model training.
* **Scikit-learn**: Random Forest Machine learning technique.
* **Streamlit**: Web framework for building the UI.
* **NumPy**: Library for handling arrays and reshaping input data.
* **Pandas**: Data handling and preprocessing.

**Hardware Requirements**

* **Processor**: Minimum dual-core processor, recommended quad-core.
* **RAM**: Minimum 8GB (16GB recommended for large datasets).
* **Storage**: At least 1GB for storing model and dataset files.

**ROS (Robotic Operating System)**

This system does not currently utilize **ROS**. However, if the system is integrated into robotic platforms, ROS can be employed for sensor data collection and communication.

**Constraints**

* The model’s accuracy is dependent on the quality and quantity of training data.
* The system is designed for engines with a specific set of sensors and operational parameters.
* Input data must be reshaped to match the model’s expected format.

**Assumptions**

* The user has access to historical data on engine performance and maintenance history.
* Sensors provide accurate, consistent data that can be used for training the model.

**Design Details**

**Process Flow**

1. **User Input**: The user provides operational settings and sensor data via a **Streamlit** interface.
2. **Data Preprocessing**: The input data is reshaped and padded as necessary to match the model's input format.
3. **Prediction**: The preprocessed data is passed to the TensorFlow model for predicting the Remaining Useful Life (RUL).
4. **Display Output**: The RUL prediction is displayed along with a maintenance recommendation.

**Model Training and Evaluation**

* The model is trained on historical RUL data.
* Model performance is evaluated using metrics such as **Mean Absolute Error (MAE)** or **Root Mean Squared Error (RMSE)**.
* After training, the model is saved in the .h5 format for deployment.

**Deployment Process**

* The system is deployed as a **Streamlit web app**.
* Users can interact with the system through a browser, inputting data via sliders and receiving predictions on the same page.

**Event Log**

* **Info Log**: Logs when the model is loaded and the prediction is made.
* **Error Log**: Logs any errors related to the input data or prediction process.

**Error Handling**

* The system includes error handling for invalid input data (e.g., out-of-range sensor readings).
* If the model cannot process the data, the user is shown a helpful error message.

**Performance**

* The prediction time should be minimal, typically under 1 second for most inputs.
* The system can handle a moderate number of concurrent users.

**Reusability**

* The model can be reused for other engines with similar sensor configurations by retraining on their respective data.
* The Streamlit frontend can be adapted for other types of predictive maintenance systems.

**Application Compatibility**

* The system is compatible with modern web browsers (Chrome, Firefox, Edge).
* **TensorFlow** requires a Python environment with the appropriate dependencies installed.

**Resource Utilization**

* The system utilizes minimal resources, relying mostly on CPU for model inference. GPU resources can be used if model training is required.

**Deployment**

* The system is deployed locally on the user's machine.
* The **Streamlit** app serves the frontend, and the backend processes predictions.

**Dashboards**

* The **Streamlit dashboard** displays the operational settings, sensor values, and predicted RUL.
* There are interactive elements such as sliders to adjust the inputs and view results dynamically.

**KPIs (Key Performance Indicators)**

* **Prediction Accuracy**: How close the predicted RUL is to actual RUL values.
* **Prediction Time**: The time taken by the model to provide a result.
* **System Uptime**: The time the system remains available for use without failure.

**Conclusion**

This **High-Level Design Document (HLD)** provides a comprehensive overview of the **Turbo Engine RUL Prediction System**. The system leverages **Streamlit** for the frontend and **TensorFlow** for the machine learning model, offering a solution that predicts the remaining useful life of turbo engines. This document covers the system architecture, design details, technical requirements, and deployment strategies.