**Low-Level Design Document (LLD)**

**Document Version Control**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 2024-12-07 | Naman Bajpai | Initial document with implementation details. |

**Abstract**

The **Low-Level Design (LLD)** provides a detailed description of the architecture, components, and implementation for the **Turbo Engine RUL Prediction System**. This includes how the system utilizes **Streamlit** for user interaction, a **Random Forest** model (machine learning), a **Convolutional Neural Network (CNN)** for predictions, **Grid Search Hyperparameter Tuning** to optimize performance, and how data is processed, predicted, and presented to the user.

**1. System Overview**

The **Turbo Engine RUL Prediction System** is a web-based application that predicts the **Remaining Useful Life (RUL)** of a turbo engine. The system processes input data (sensor readings and operational settings), passes it through either a **Random Forest** or a **CNN** model, and provides an estimated RUL. The model's performance is optimized using **Grid Search Hyperparameter Tuning**.

**2. System Components**

**2.1. Frontend (Streamlit)**

* **Streamlit Interface**: Provides a user-friendly interface to input engine sensor data and operational settings. The interface allows the user to adjust parameters using sliders and buttons and view the predicted RUL.

**Key Features:**

* **Input Fields**: Sliders for input values such as engine speed, temperature, and pressure.
* **Output**: Displays the predicted **Remaining Useful Life (RUL)** and maintenance recommendation.

**2.2. Backend (Model Prediction)**

**Data Preprocessing**

* **Data Formatting**: Input data is collected and preprocessed to match the expected input shape of the machine learning models. This includes reshaping and scaling the data.
* **Data Handling**: Libraries such as **Pandas** are used to process the data before feeding it into the model.

**Model (Random Forest, CNN, and TensorFlow)**

* **Random Forest Model**:
  + Trained using **Scikit-learn**.
  + **Grid Search** is employed for **hyperparameter tuning**, adjusting parameters like the number of trees, maximum depth, and minimum samples per leaf.
  + The model is used to predict the **Remaining Useful Life (RUL)**.
* **Convolutional Neural Network (CNN)**:
  + A **CNN** model is used to predict the **RUL in the GUI**.
  + **Keras/TensorFlow** is used to build and train the CNN model for RUL prediction, utilizing the engine's operational data transformed into image-like representations (e.g., spectrograms or time-series converted to 2D matrices).

**Model Integration:**

* The models (Random Forest, CNN, and TensorFlow) are loaded into the backend.
* The models take the user-provided input, process it, and return the **RUL** prediction.

**3. Detailed Architecture and Flow**

**3.1. Data Flow**

1. **Input Collection**: The user enters the operational settings (speed, temperature, pressure, etc.) through the **Streamlit** interface.
2. **Data Preprocessing**: The data is formatted and cleaned using **Pandas**. It is then normalized or scaled using techniques such as **StandardScaler** or **MinMaxScaler**.
3. **Model Prediction**:
   * The **Random Forest** model is applied to the preprocessed data to predict the RUL.
   * **Grid Search Hyperparameter Tuning** is applied on the Random Forest model before prediction to optimize its accuracy.
   * The **CNN** model is used for predictions, trained to recognize patterns in time-series data transformed into an image-like form.
4. **Prediction Display**: The predicted RUL value is shown to the user on the **Streamlit** dashboard.

**3.2. Model Implementation**

* **Random Forest Model**:
  1. **Training**: The model is trained using the historical data of engine operational settings and RUL values.
  2. **Hyperparameter Tuning**: A **GridSearchCV** object is used to fine-tune the model's hyperparameters.
  3. **Model Saving**: After training, the model is saved using **joblib** or **pickle** for deployment.
* **CNN Model**:
  + **Keras/TensorFlow** is used to create a **CNN** model that predicts RUL from image-like time-series data.
  + The CNN is trained using historical engine sensor data, transformed into spectrograms or other 2D representations.

**4. System Functionality**

**4.1. Streamlit Interface**

* **Input Fields**:
  + Sliders for entering operational settings like engine speed, temperature, and pressure.
  + A button to trigger the model prediction.
* **Prediction Output**:
  + Displays the predicted **Remaining Useful Life (RUL)**.
  + Includes a maintenance recommendation based on the predicted RUL.

**5. Error Handling**

* **Invalid Input Handling**: If the user enters data outside of the expected ranges, the system will display an error message.
* **Model Error Handling**: In case of issues with model inference (e.g., invalid model format), an appropriate error message is shown.

**6. Deployment and Hosting**

* **Model Deployment**:
  + The model is locally deployed.
  + The **Streamlit** app is deployed to serve the frontend and handle user interactions

**7. Performance**

* **Prediction Time**: The system aims to provide predictions in under 1 second for most inputs.
* **Scalability**: The backend model can handle a moderate number of concurrent users.

**Conclusion**

This **Low-Level Design Document (LLD)** details the specific implementation and integration of the components for the **Turbo Engine RUL Prediction System**. It covers the **Streamlit frontend**, **Random Forest model**, **CNN model**, and **Grid Search hyperparameter tuning**, along with data preprocessing, model training, and deployment strategies. The LLD serves as a guide for developers to implement the system efficiently.