EEC4400 Assignment on Predictive Maintenance

Predictive maintenance involves developing models that accurately determine the condition of in-service equipment (typically industrial machinery), thereby determining when the equipment might encounter failures and/or when maintenance must be performed on them. This approach achieves significant cost savings as maintenance tasks are performed only when warranted as opposed to planned periodic maintenance.

1 Goal

For this assignment, you will work in a group of four students to develop a predictive maintenance system. You will use this system to: (1) predict whether or not an industrial machine is going to fail, as well as (2) predict the point of time at which the machine will fail by predicting its "Remaining Useful Lifetime" (RUL). Through this assignment, you will become familiar with practical implementations of Deep Learning models like CNN and LSTM.

2 Data Description

The data set is derived from running turbofan engines. The data set is divided into a training set and a test set. The dataset contains 21 sensor readings which form the multi-variate time series data and 100 different machines' data are recorded.

Each machine initially has some wear, and there are manufacturing variations, but this is considered normal. There are three operational settings that affect engine performance. Data is also contaminated by sensor noise.

At the beginning, the engine starts at normal state, and a fault occurs at some later point. In the training set, the fault grows until a system failure occurs. Hence, each observation in the training set is an operation cycle and each machine's total run to failure data is captured. In the test set, the time series ends before a system failure occurs. Thus, the problem statement is to predict the number of remaining operational cycles before a system failure occurs (also called the "Remaining Useful Lifetime" (RUL)) at the end of each sequence in the test set.

3 Platform

Students may use any Python platform with recent versions of the numpy, pandas, scikit-learn, matplotlib, seaborn and tensorflow packages installed. For example, the Anaconda Python distribution has most of the packages except tensorflow, which can be added easily (this also installs tensorboard). Note that the final submission must be done in a Jupyter notebook containing all code, output and results (see page 7).

A skeleton code file is provided which must be used by students to complete the assignment by following the instructions in the file. Some useful aspects of the assignment are already coded, and their functionality must be built upon to complete the assignment.

4 Overview

For this assignment, students will design deep learning models like CNN and LSTM to tackle two types of prediction tasks:

1. Binary Classification: Predicting if a machine will fail in the next window of “w1” operating cycles.
2. Regression: Predicting the exact time a machine will fail by determining the number of operation cycles left till failure (RUL).

This assignment highlights that real-world prediction tasks are complex and requires different model architectures which produces different prediction results.

The tasks in this assignment are as follows:

1. Data ingestion, exploration, and pre-processing.
2. Setting up TensorBoard.
3. Training and Evaluating 1D CNN and LSTM models for Binary Classification.
4. Training and Evaluating 1D CNN and LSTM model for RUL Regression.
5. Comparison of CNN and LSTM models for each prediction task.
6. Performing Hyperparameter Optimisation / Tuning.
7. Visualisation with TensorBoard.

A skeleton Jupyter notebook file is provided. Copy the Jupyter notebook to your computer.

Note: While each student can work on his/her part using separate Jupyter notebooks, all parts need to be combined into a single Jupyter notebook for submission at the end of the assignment.

Write the required Python code in the cells in the Jupyter notebook at places indicated by “[WriteCode]”.

The following table shows the responsibilities of each student (shown in a particular colour) and the models and parameter settings for the different cases:

Failure Classification				**RUL Regression**			
Model	**Network architecture**	**Other hyper-parameters**	**No. of epochs & Batch size**	**Model**	**Network architecture**	**Other hyper-parameters**	**No. of epochs & Batch size**
Student 1				**Student 3**			
CNN-Bin	CNN1	CNN-Base-P	Settings cell	**CNN-RUL**	CNN1	CNN-Base-P	Settings cell
CNN-Bin-Alt	CNN1	CNN-Alt-P1		**CNN-RUL-Alt**	CNN1	CNN-Alt-P2	
Student 2				**Student 4**			
LSTM-Bin	LSTM1	LSTM-Base-P	Settings cell	**LSTM - RUL**	LSTM1	LSTM-Base-P	Settings cell
LSTM-Bin-Alt	LSTM1	LSTM-Alt-P1		**LSTM - RUL-Alt**	LSTM1	LSTM-Alt-P2	

Note: Entities with the same label, e.g. “CNN1”, must have the same characteristics (*note*: final output layer can be different for classification vs regression). Likewise, entities with different labels, e.g. “CNN-Alt-P1” and “CNN-Alt-P2”, can have different characteristics.

5 Failure and Remaining Useful Lifetime Prediction with Deep Learning

5.1 Data ingestion, exploration, and pre-processing

This section involves loading the given dataset and performing data pre-processing. In addition, there is some exploration of the underlying characteristics of the data.

Students must perform the following steps:

- *Import the dataset. (Done)*
- *Explore the data by plotting the cross-correlation heat map. (Done)*
- *Identify the machines with the shortest and longest operating times before failure and plot the corresponding data. (Done)*
- *Pre-process the training and test data by adding new columns "label1" for binary classification and "RUL" for regression. The "RUL" column for each machine ID would contain numbers in decreasing trend which denote the number of operating cycles remaining till failure. After populating the "RUL" column, "label1" column is populated in such a way that observations which are "w1" operating cycles or closer to failure are assigned '1' while others are assigned '0' pertaining to the classification task. (note: please refer to the skeleton code file for detailed information) (Done)*
- Data normalization with min-max scaling.
- Plotting the last 50 operating cycles of one particular machine.
- *Convert training and test data into sequences. Conversion to sequence is necessary in order to pass time series data to sequential deep learning models like CNN and LSTM since time series data have temporal relationships (note: this is already provided in the skeleton file; students can observe the code to understand the process). (Done)*

5.2 Setting Up TensorBoard

This section involves creating the log directory ("eec4400_logs") to store each model's training run to visualize in TensorBoard. The following steps must be performed:

- Create a log directory for storing each training run of the models.
- Set up TensorBoard to use the logs in this directory for visualisation of results.

Note: This is already provided in the skeleton file, students can observe the code to understand the process.

5.3 Training and Evaluating 1D CNN and LSTM models for Binary Classification

In this section, students are expected to do the following:

1. Construct a 1D CNN model for binary classification.
2. Train and evaluate the CNN model.
3. Compute the accuracy, confusion matrix, precision, recall, and F1 score for training data.
4. Use the trained model to predict test data and compute relevant metrics.
5. Construct an LSTM model for binary classification.
6. Repeat the above steps for the LSTM model.

5.4 Training and Evaluating CNN 1D and LSTM models for RUL Regression

In this section, students are expected to do the following:

1. Construct a 1D CNN model for regression using the same architecture as classification.
2. Train and evaluate the CNN model.
3. Compute the Mean Absolute Error (MAE) metric for training data prediction.
4. Use the trained model to predict test data and compute relevant metrics.
5. Construct an LSTM model for regression using the same architecture as classification.
6. Repeat the above steps for the LSTM model.

5.5 Comparison of CNN and LSTM models for each prediction task

In this section, students are expected to use the various performance metrics computed in the previous sections to compare the different models and their performance with respect to the two different prediction tasks. By keeping the model architecture the same for both tasks, the difference in the performance can be observed and correlated to the defining characteristics of each model. Through this, students should obtain reasonable clarity on the following:

1. The edge recurrent neural network models like LSTM have over feedforward models like CNN in time series data prediction.
2. Not all prediction tasks are the same for a given application or available data, and the need for models to be customized for the task at hand.

5.6 Hyperparameter Optimisation

Hyperparameter optimisation (or tuning) tries to pick the best possible values for various aspects like network architecture (no. and types of layers, no. of nodes per layer, no. and type of convolution filters, activation functions), type of optimizers and values for learning rate, dropout rate etc. for each of the models. One way to do hyperparameter optimisation is to perform a grid search, but this is highly computationally expensive due to the large number of possible permutations of the values of the parameters. Other methods are Bayesian optimisation, random sampling etc.

Since grid search and other hyperparameter optimisation methods are highly computationally expensive, you will only design and implement an improved CNN or LSTM model (referred to as the "alternative" or "alt" model) with the **same** network architecture (no. and types of layers, no. of nodes per layer, no. and type of convolution filters, activation functions) as the baseline CNN or LSTM model, but you are free to select different values for the other parameters (you may vary the value(s) of one or more parameters) that improves the performance of the original model.

1. Train and evaluate the alternative models. Use the trained alternative models to predict the test data.
2. Compute the relevant performance metrics for both classification and regression.

5.7 Visualisation with TensorBoard

Once the various models are trained and evaluated, their results can be examined using TensorBoard. The callback function (refer to the skeleton file for more information) used when training the models stores data from the runs in the log directory created and can be visualized in TensorBoard.
Note: This is already provided in the skeleton file, students can observe the code to understand the process.

Explore the different TensorBoard panels: scalars, graphs, distributions, histograms, time series, and the various settings. Refer to online resources for information on TensorBoard features.

6 Results and Report Writing

Two scenarios are considered:

- Sequence_length = 25
- Sequence_length = 50

Write a report of about 10 pages in total for the group. Indicate the names and student numbers of Student 1, Student 2, Student 3 and Student 4 (corresponding to the scope of work defined above and below) clearly at the top of the first page of your report.

The report should have the following sections:

Data Exploration (All Students)

Use the data exploration techniques implemented – cross-correlation heat map and time series and histogram data plots – to make some observations about the characteristics of the sensor data. What can be said about the correlation between different sensor readings? Is there any relationship between the underlying measurements and the lifetime of the machine before failure?

Failure Classification – CNN (Student 1)

Obtain the performance metrics from the results in the Jupyter notebook and TensorBoard scalars panel.

Compare the performance of the baseline CNN model for binary classification for sequence_length = 25 and 50. Comment on the performance of the CNN model when predicting after observing a longer past sequence pattern (i.e. going from 25 to 50 time points in the past).

For sequence_length = 50, what is the effect of performing hyperparameter optimisation, i.e. using an alternative set of parameter values for learning rate, optimizer, dropout rate, batch size, no. of epochs etc.? *note*: You may vary the value(s) of one or more parameters.
Comment on the performance of the alternative model with respect to the baseline model.
You are encouraged to use the results from TensorBoard for the baseline and alternative models, e.g. distribution/magnitude/behaviour of the values of the weights as training progresses, in your explanations.

Failure Classification – LSTM (Student 2)

Obtain the performance metrics from the results in the Jupyter notebook and TensorBoard scalars panel.

Compare the performance of the baseline LSTM model for binary classification for sequence_length = 25 and 50. Comment on the performance of the LSTM model when predicting after observing a longer past sequence pattern (i.e. going from 25 to 50 time points in the past).

For sequence_length = 50, what is the effect of performing hyperparameter optimisation, i.e. using an alternative set of parameter values for learning rate, optimizer, dropout rate, batch size, no. of epochs etc.? *note*: You may vary the value(s) of one or more parameters.
Comment on the performance of the alternative model with respect to the baseline model.
You are encouraged to use the results from TensorBoard for the baseline and alternative models, e.g. distribution/magnitude/behaviour of the values of the weights as training progresses, in your explanations.

RUL Regression – CNN (Student 3)

Obtain the performance metrics from the results in the Jupyter notebook and TensorBoard scalars panel.
Compare the performance of the baseline CNN model for RUL regression for sequence_length = 25 and 50. Comment on the performance of the CNN model when predicting after observing a longer past sequence pattern (i.e. going from 25 to 50 time points in the past).

For sequence_length = 50, what is the effect of performing hyperparameter optimisation, i.e. using an alternative set of parameter values for learning rate, optimizer, dropout rate, batch size, no. of epochs etc.? *note*: You may vary the value(s) of one or more parameters.
Comment on the performance of the alternative model with respect to the baseline model.
You are encouraged to use the results from TensorBoard for the baseline and alternative models, e.g. distribution/magnitude/behaviour of the values of the weights as training progresses, in your explanations.

RUL Regression – LSTM (Student 4)

Obtain the performance metrics from the results in the Jupyter notebook and TensorBoard scalars panel.

Compare the performance of the baseline LSTM model for RUL regression for sequence_length = 25 and 50. Comment on the performance of the LSTM model when predicting after observing a longer past sequence pattern (i.e. going from 25 to 50 time points in the past).

For sequence_length = 50, what is the effect of performing hyperparameter optimisation, i.e. using an alternative set of parameter values for learning rate, optimizer, dropout rate, batch size, no. of epochs etc.? *note*: You may vary the value(s) of one or more parameters.
Comment on the performance of the alternative model with respect to the baseline model.
You are encouraged to use the results from TensorBoard for the baseline and alternative models, e.g. distribution/magnitude/behaviour of the values of the weights as training progresses, in your explanations.

Joint Student 1 and Student 3

Verify the number of trainable parameters (weights) in the CNN model, i.e. do a calculation based on your understanding and compare with the reported no. of trainable parameters.

Explain the reasoning behind the design of the baseline and alternative CNN network architecture and the choice of the values of the parameters.

Joint Student 2 and Student 4 Verify the number of trainable parameters (weights) in the LSTM model, i.e. do a calculation based on your understanding and compare with the reported no. of trainable parameters. Explain the reasoning behind the design of the baseline and alternative LSTM network architecture and the choice of the values of the parameters.
Joint Student 1 and Student 2 Fix the sequence_length = 50 and compare the performance of CNN vs LSTM for failure classification. What are the strengths and weaknesses of CNN vs recurrent networks like LSTM for failure classification?
Joint Student 3 and Student 4 Fix the sequence_length = 50 and compare the performance of CNN vs LSTM for RUL regression. What are the strengths and weaknesses of CNN vs recurrent networks like LSTM for RUL regression?
Execution Time and No. of Parameters (All Students) For sequence_length = 50, compare the execution times and no. of trainable parameters (weights) with performance of all the implemented models. Are the execution times and no. of trainable parameters (weights) justified for each of the implemented models?

7 Submission

Submit your completed **Jupyter notebooks** with **all code, output and results** (which should be named as "EEC4400-Groupxx.ipynb", where xx is your group no.) **by 11.59pm on Saturday 23 November 2024**. There should only be 1 submission from each group.

Submit your **report in PDF format** with file name "EEC4400-Groupxx-Report.pdf" by the same deadline. There should only be 1 submission from each group.

Submission instructions will be provided later.