# EEC4400 Assignment (AY2023/24)
# Anomaly Detection using Deep Learning

## 1 Anomaly Detection

As the world steps into Industry 4.0, the Internet of Things (IoT) enables real-time visibility into various processes in industry through data from sensor measurements. Data has become the lifeblood of the 'digital' factory. Data analysis provides insights into various processes, and the results provide guidance on appropriate actions to improve operations.

Anomaly Detection is one of the essential components of data analysis in the modern industrial world. It involves developing models that identify rare events, or observations which deviate significantly from most of the data and do not conform to a well-defined notion of normal behaviour. In most scenarios, the data collected reflects the activities in the system, and the anomalies are due to unusual operation behaviours. Thus, the definition of an anomaly varies according to the context of specific problems.

In this assignment, an anomaly refers to the abnormal conditions which indicate that a machine in a factory will fail in the near future.

## 2 Goal

For this assignment, students will train several models to detect anomalies which indicate that the machine of interest will fail in the near future. This is useful so that actions can be taken and/or further analysis can be done to prevent such failures. Through this assignment, students will become familiar with practical implementations of deep learning models such as CNN, LSTM and Hybrid CNN-LSTM and their usefulness for anomaly detection.

## 3 Data Description

The dataset provided consists of four sensor measurements of a machine, i.e. voltage, rotation, vibration and pressure, recorded hourly over a period of one year.

Two files are given, the file ‘sensor_data_ampl.csv’ contains the timestamps and measurements, while the file ‘fail_log.csv’ consist only of the timestamps of known failure events.

In the context of this assignment, an anomaly is defined as the abnormal conditions before failure events.

**Specifically, we will perform anomaly detection by classifying, at each point in time, whether current and previous sensor measurements are abnormal.**

Data labelling has been done in the sensor_data_ampl.csv file, where abnormal sensor measurements occurring 24 hours before each failure are labelled with ‘1’ while the normal instances are labelled with ‘0’.

# 4 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 Jupyter notebook file with skeleton code is provided which must be used by students to complete the assignment by following the prompts in the file. Some useful aspects of the assignment are already coded, and their functionalities must be expanded by writing additional code in sections indicated with [WriteCode] in order to complete the assignment.

# 5 Overview

For this assignment, students will design deep learning models such as CNN, LSTM and Hybrid CNN-LSTM to tackle anomaly detection tasks:

This assignment highlights that real-world anomaly detection tasks are complex and requires careful data processing and model architecture design in order to obtain good prediction results.

The tasks in this assignment are as follows:

1. Data ingestion, exploration, and pre-processing.
2. Setting up TensorBoard.
3. Specifying, Training and Evaluating CNN, LSTM and Hybrid CNN-LSTM models for anomaly detection.
4. Comparison of CNN, LSTM and Hybrid CNN-LSTM models for anomaly detection.
5. Visualization 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]".

# 6 Assignment

## 6.1 Data ingestion, exploration, and pre-processing

This section involves loading the given dataset and performing data pre-processing. In addition, some exploration of the underlying characteristics of the data is also done.
Students must perform the following steps:

- Import the dataset.
- Explore the data by plotting the time sequence diagram with failure events. (Done)
- Apply sliding window averaging to remove the effects of high frequency noise from the sensor readings. (Done; but do further investigation)

- Pre-process the training and test data by adding new column “label” for anomaly detection. The “label” column is populated in such a way that the instances, e.g. 24 hours, before the logged failure events are assigned ‘1’ while others are assigned ‘0’ pertaining to the task. (Done)
- Data normalization with min-max scaling.
- 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 as time series data have temporal relationship (*note*: this is already provided in the skeleton file; students can observe the code to understand the process).

## 6.2 Setting Up TensorBoard

This section involves creating the log directory ("ee4400_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 visualization of results.

(*note*: this is already provided in the skeleton file, students can observe the code to understand the process).

## 6.3 Specifying, Training and Evaluating CNN, LSTM and Hybrid CNN-LSTM models for Binary Classification

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

1. Construct a CNN model (Conv1D or Conv2D) 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 failures in the entire test period and compute the confusion matrix and relevant metrics (accuracy, precision, recall, F-score).
5. Construct an LSTM model for binary classification.
6. Repeat steps 1-4 for the LSTM model.
7. Construct a Hybrid CNN-LSTM model for binary classification.
8. Repeat steps 1-4 for the Hybrid CNN-LSTM model.

*Note 1*: The Hybrid CNN-LSTM model has not been covered in lectures. It is supposed to combine the advantages of CNN and LSTM. Do your own research on how to implement a Hybrid CNN-LSTM network using TensorFlow.

*Note 2*: The training data consists of data from 00:00 on 1 January 2015 till 23:00 on 31 May 2015. Strictly speaking, the test data should be different from training data, but due to limited failure instances and the complexity of the problem, we will use the entire period from 00:00 on 1 January 2015 till 06:00 on 1 January 2016 as test data.

For binary classification, the decision threshold affects the positive (‘anomaly’) and negative (‘normal’) decisions. Determine the appropriate threshold values for each of the 3 models that lead to good precision, recall and F-score.

*Note 3*: The **same threshold value** should be used for a particular model for determining the anomaly detection performance on **both** the training data and the test data.

### 6.4 Comparison of the three models

In this section, students are expected use the various performance metrics computed in the previous sections to compare the different models and their performance. Discuss the relative performance of the models and provide explanations where possible.

Explain the number of trainable parameters reported by model.summary().

### 6.5 Visualization 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 the runs in the log directory created and can now be visualized in TensorBoard.

(*note*: how to record data for TensorBoard is already done in the skeleton file, students can observe the code to understand the process)

## 7 Results and Report Writing

The experiment above is performed for two scenarios:

- Sequence length (seq_len) = 30
- Sequence length (seq_len) = 40

For sequence length (seq_len) = 30, investigate the effects of changing the sliding window averaging parameters sliding_window_width and sliding_step_size for all the cases.

Collect all the required plots (including TensorBoard visualizations) and performance metrics.

### 7.1 Evaluation

In this section, students must **use the experimental results obtained** to answer several questions.

1. Compare the plotted data before and after applying the sliding window averaging and make some observations. Why would removing noise be a significant step in data pre-processing? Can you spot some potential failures that are not logged from the original sensor measurements plot? What about from the sliding window averaged data plot?
2. After training the models, you may notice that the training accuracy and validation accuracy can be quite high. What could be the reason? Does high training accuracy necessarily mean a good model? In the context of this assignment, which metric(s) is(are) more important? Why?
3. Compare the performance of the CNN model for seq_len = 30 and 40. Comment on the performance of the CNN model when predicting after observing a longer past sequence pattern (i.e. going from 30 to 40 time points in the past).
4. Compare the performance of the LSTM model for seq_len = 30 and 40. Comment on the performance of the LSTM model when predicting after observing a longer past sequence pattern (i.e. going from 30 to 40 time points in the past).

5. Compare the performance of the Hybrid CNN-LSTM model for seq_len = 30 and 40. Comment on the performance of the Hybrid CNN-LSTM model when predicting after observing a longer past sequence pattern (i.e. going from 30 to 40 time points in the past).
6. Using the same seq_len and sliding window averaging parameters, compare the performance of the three models. What is the effect of using recurrent networks like LSTM compared to CNN, as well as the Hybrid CNN-LSTM model? (just state the facts; explanations and/or hypotheses are to be provided in a separate section – see Section 7.2)
7. Investigate the effects of changing the sliding window averaging parameters sliding_window_width and sliding_step_size. Explain the different results observed.
8. Comment on the model weight distribution plots in TensorBoard. Can some parallels be drawn about the distribution/values/magnitude/behaviour of the weights to the differences in performance in the three models?

## 7.2 Report Writing

Write a short report of no more than 10 pages highlighting your work in this assignment. The report should contain the following information:

- The reasoning behind the design of the network architectures.
- Answers to the questions in Section 7.1.
- Explanations and/or hypotheses on the relative performance of the different models.

Any other efforts or explorations can be highlighted in the report, but you should adhere to the page limit.

*Note*: the code and results in the Jupyter notebook carry half the marks, and the report carries the other half of the marks, for this assignment.

## 7.3 Division of Responsibilities

The tasks to be completed by individuals or the group are summarized here:

| | **Task** | **Student** | | |
|---|---|---|---|---|
| | | 1 | 2 | 3 & 4 |
| **Jupyter notebook** | Data normalization with min-max scaling. | all | | |
| | Construct CNN model | ✓ | | |
| | Construct LSTM model | | ✓ | |
| | Construct Hybrid CNN-LSTM model | | | ✓ |
| | Performance metrics; Comparison of the three models | all | | |
| | Visualization with TensorBoard | ✓ | ✓ | ✓ |
| **Report** | Reasoning behind the design of the network architectures | ✓ | ✓ | ✓ |
| | 7.1.1, 7.1.2, 7.1.6, 7.1.7, 7.1.8 | all | | |
| | 7.1.3 | ✓ | | |
| | 7.1.4 | | ✓ | |
| | 7.1.5 | | | ✓ |
| | Explanations and/or hypotheses on the relative performance of the different models | all | | |

all: collaborative item; ✓: individual item

Student 3 and student 4 are treated as an entity, all individual tasks assigned to student 3 and 4 should be done through collaboration of the pair. Apart from the tasks stated in the table above, you are welcome to write extra codes to collect experiment results, tune hyperparameters etc. for the purpose of validating your analysis done in the report. If such code is included, please organize the experiment codes by person, and indicate each person's work clearly.

# 8 Submission and Grading

## 8.1 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. 01 to 10) **by 11.59pm on Friday 29 December 2023** following the submission instructions (e.g. e-mail to Mr Chai Yingtao: yingtao.chai@nusricq.cn). 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 following the submission instructions. There should only be 1 submission from each group.

## 8.2 Grading

- Students within the same group may obtain different scores.
- Individual tasks will be graded independently, i.e. model constructed by student 1 will not affect the grading of students 2, 3 and 4.
- Model construction is an elaborate exercise. There are many aspects to be considered for solving a specific real-life problem. You are expected to make use of the theoretical concepts learnt in class, and do some research (if necessary) to support your choices made.
- Performance of the models will not be the main factor considered in grading; however, models should do reasonably well in the performance metrics. Students are expected to explore, conduct experiments, collect results and analyse them.
- For the report, students should not spend too much effort summarizing/explaining the code, but instead, focus on analysis of why and how. Some examples are given below:
  - e.g. Why was a 1D CNN is chosen over 2D CNN (or the other way round)? What are the pros and cons of the 2 approaches?
  - e.g. How was value *th* determined as the threshold value?
  - e.g. Do the collected results agree with theory?
- The hybrid CNN-LSTM model has not been covered in class. Students 3 and 4 are expected to do research on the topic. Justify the mechanism behind distributed CNN models through time.
- Cite your reference sources. The reference list will not count towards the 10-page limit.