DL Based Sensor Monitoring Model for detecting Sensors Data Injection Attacks

Raichuri Vishnu Sai Rayulu (rr4302)

Introduction

In the era of Industry 4.0, predictive maintenance (PdM) has emerged as a game-changing solution, harnessing the power of cutting-edge machine learning algorithms, particularly deep learning, and Internet of Things (IoT) sensors to predict component or system faults. This transformative technology offers tremendous potential for improving the reliability and efficiency of industrial processes. This project aims to explore employ a Deep Learning approach to detect various data injection attacks in sensor data. Predictive maintenance models play a crucial role in operational efficiency and safety across various industries. These models often rely on sensor data from machines to predict their Remaining Useful Life (RUL). However, the integrity of these models can be compromised through Faulty Data Injection Attacks (FDIA), where sensor data is tampered with, leading to inaccurate predictions. This report delves into the development of a deep learning model for predictive maintenance and examines the impact of injection attacks on its performance.

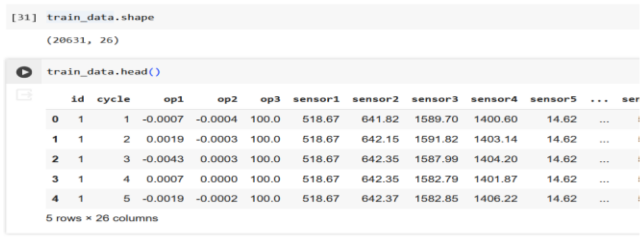
Literature Survey

Recent advancements in Intrusion Detection Systems (IDS) have increasingly leveraged deep learning models. [1] demonstrated an IDS using a stacked auto-encoder (SAE) to classify network traffic, including impersonation, flooding, injection, and normal traffic. Notably, a simpler two-hidden-layer model achieved a remarkable 98.67% accuracy, surpassing the three-layer variant. Complementarily, [2] also proposed a deep learning-based IDS, categorizing wireless network traffic into similar classes. These studies underscore the efficacy of deep learning in network security, particularly in identifying and classifying various cyber threats.

Data Set and Data Pre-processing

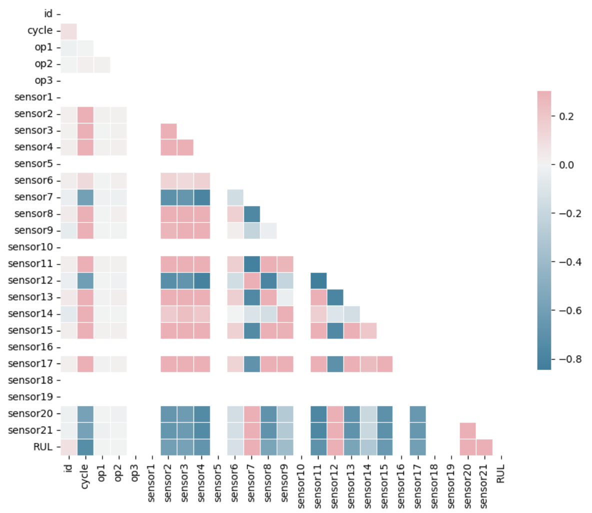
DataSet: https://www.kaggle.com/datasets/behrad3d/nasa-cmaps
For this project I utilized the widely recognized NASA turbofan engine degradation simulation dataset known as C-MAPSS (Commercial Modular Aero-Propulsion System Simulation). This dataset consists of 21 sensor data streams captured under various operating conditions and fault scenarios. Within C-MAPSS, there are four distinct sub-datasets denoted as FD001-04, each of which contains both training and test data. The test data within these sub-datasets contains run-to-failure data collected from multiple engines of the same type.

As dataset comprises sensor data from various engines. The primary objective is to predict the Remaining Useful Life (RUL) of these engines based on sensor readings. Analyzing the data patterns and selecting relevant features is critical for the model's accuracy. Certain sensors show high correlation with RUL, while others exhibit constant readings, leading to their exclusion.

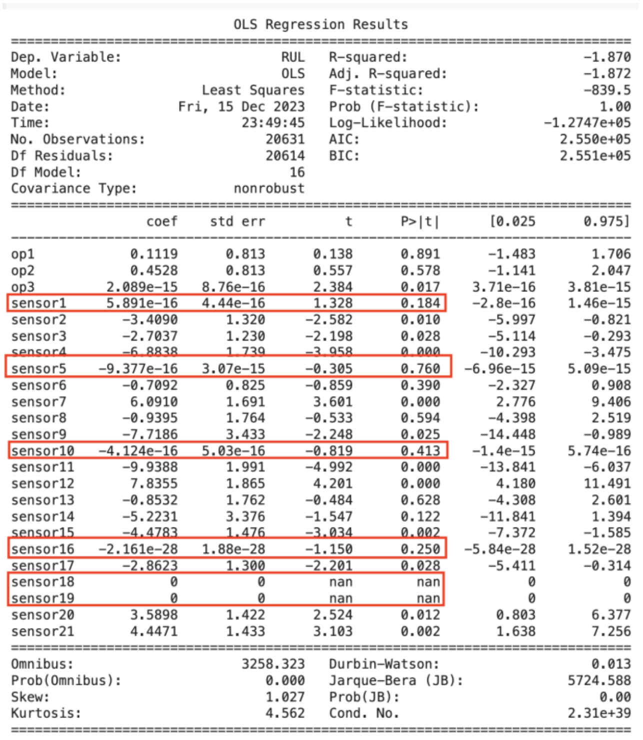


```
[31] train_data.shape

     (20631, 26)

     train_data.head()

        id  cycle      op1      op2    op3  sensor1  sensor2  sensor3  sensor4  sensor5  ...  ser
     0   1      1  -0.0007  -0.0004  100.0   518.67   641.82  1589.70  1400.60    14.62  ...
     1   1      2   0.0019  -0.0003  100.0   518.67   642.15  1591.82  1403.14    14.62  ...
     2   1      3  -0.0043   0.0003  100.0   518.67   642.35  1587.99  1404.20    14.62  ...
     3   1      4   0.0007   0.0000  100.0   518.67   642.35  1582.79  1401.87    14.62  ...
     4   1      5  -0.0019  -0.0002  100.0   518.67   642.37  1582.85  1406.22    14.62  ...
     5 rows × 26 columns
```

For data visualization, specifically to understand the relationships (correlations) between different variables in the dataset a heat map is generated.

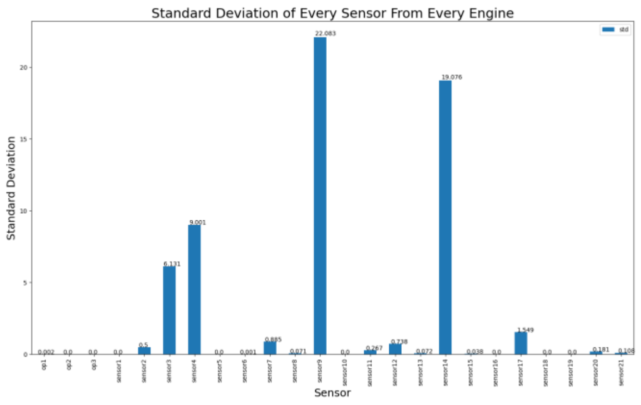


Most of the sensors are pairwise correlated with RUL, except for sensors 1, 5, 10,16, 17, 18. The code is written to perform linear regression to predict the RUL using features like sensor readings. It scales the features for normalization, fits an Ordinary Least Squares (OLS) model, and prints a summary with coefficients and statistical measures. This analysis helps in understanding how different parameters affect the RUL.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    RUL   R-squared:                      -1.870
Model:                            OLS   Adj. R-squared:                 -1.872
Method:                 Least Squares   F-statistic:                    -839.5
Date:                Fri, 15 Dec 2023   Prob (F-statistic):               1.00
Time:                        23:49:45   Log-Likelihood:            -1.2747e+05
No. Observations:               20631   AIC:                         2.550e+05
Df Residuals:                   20614   BIC:                         2.551e+05
Df Model:                          16
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
op1            0.1119      0.813      0.138      0.891      -1.483       1.706
op2            0.4528      0.813      0.557      0.578      -1.141       2.047
op3         2.089e-15   8.76e-16      2.384      0.017    3.71e-16    3.81e-15
sensor1     5.891e-16   4.44e-16      1.328      0.184    -2.8e-16    1.46e-15
sensor2       -3.4090      1.320     -2.582      0.010      -5.997      -0.821
sensor3       -2.7037      1.230     -2.198      0.028      -5.114      -0.293
sensor4       -6.8838      1.739     -3.958      0.000     -10.293      -3.475
sensor5    -9.377e-16   3.07e-15     -0.305      0.760   -6.96e-15    5.09e-15
sensor6       -0.7092      0.825     -0.859      0.390      -2.327       0.908
sensor7        6.0910      1.691      3.601      0.000       2.776       9.406
sensor8       -0.9395      1.764     -0.533      0.594      -4.398       2.519
sensor9       -7.7186      3.433     -2.248      0.025     -14.448      -0.989
sensor10   -4.124e-16   5.03e-16     -0.819      0.413    -1.4e-15    5.74e-16
sensor11      -9.9388      1.991     -4.992      0.000     -13.841      -6.037
sensor12       7.8355      1.865      4.201      0.000       4.180      11.491
sensor13      -0.8532      1.762     -0.484      0.628      -4.308       2.601
sensor14      -5.2231      3.376     -1.547      0.122     -11.841       1.394
sensor15      -4.4783      1.476     -3.034      0.002      -7.372      -1.585
sensor16   -2.161e-28   1.88e-28     -1.150      0.250   -5.84e-28    1.52e-28
sensor17      -2.8623      1.300     -2.201      0.028      -5.411      -0.314
sensor18            0          0        nan        nan           0           0
sensor19            0          0        nan        nan           0           0
sensor20       3.5898      1.422      2.524      0.012       0.803       6.377
sensor21       4.4471      1.433      3.103      0.002       1.638       7.256
==============================================================================
Omnibus:                     3258.323   Durbin-Watson:                   0.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5724.588
Skew:                           1.027   Prob(JB):                         0.00
Kurtosis:                       4.562   Cond. No.                     2.31e+39
==============================================================================
```

Some sensors have no significant coefficients in a simple OLS (performed with normalized data). Sensors with near 0 variability dropped. Standard deviation of remaining sensors was plotted. Measurements from some sensors are constant across engines and cycles(5, 15,10, 16, 18, 19). Based on observation these features were dropped.

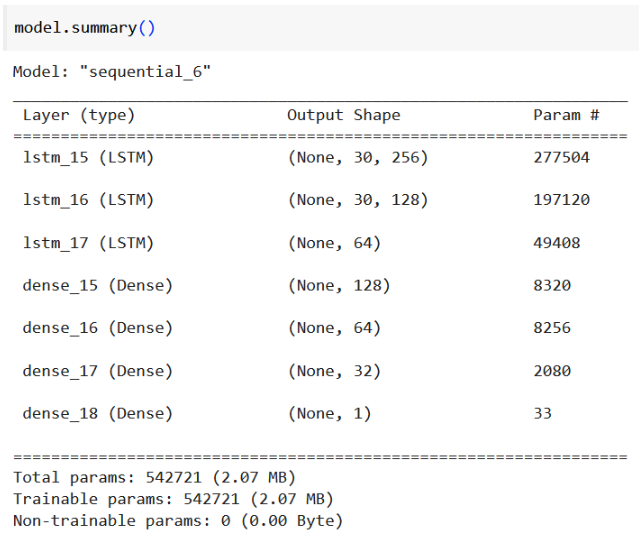


Technical Details of the Deep Learning Model

The core of the predictive maintenance system is a Recurrent Neural Network (RNN). This model uses sensor data to predict the RUL of engines. Key model parameters include the RMSE loss function, Adam optimizer, and TanH and Relu activation functions. There is no RUL data provided for the training set. So, I counted RUL based on the cycles in the data from Engine-1 fails after 192 cycles. So, at the first cycle of Engine-1, its RUL is 191, after cycle 2, its RUL is 190 and so on. As it fails after 192nd cycle, its RUL is 0. We also have a vector of true RUL values for the test data.

Input Variables – Data from the 21 sensors
Target Variable – RUL (Remaining Useful Life)

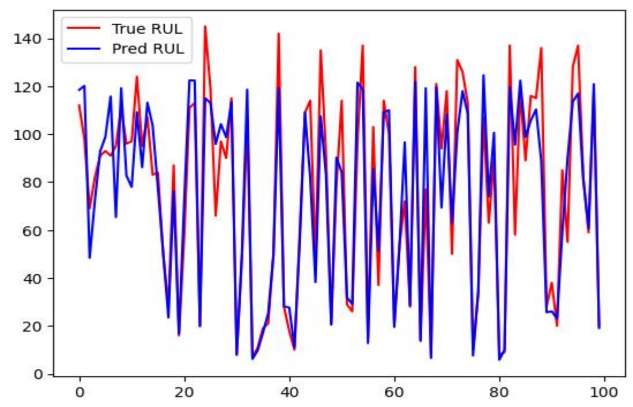
To fine tune training process, the learning rate is set to 0.001 for the first 5 epochs and from the 6th epoch, the learning rate is updated with each epoch according to the formula, learingrate*tf.math.exp(-0.1).



```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_15 (LSTM)              (None, 30, 256)           277504

 lstm_16 (LSTM)              (None, 30, 128)           197120

 lstm_17 (LSTM)              (None, 64)                49408

 dense_15 (Dense)            (None, 128)               8320

 dense_16 (Dense)            (None, 64)                8256

 dense_17 (Dense)            (None, 32)                2080

 dense_18 (Dense)            (None, 1)                 33

=================================================================
Total params: 542721 (2.07 MB)
Trainable params: 542721 (2.07 MB)
Non-trainable params: 0 (0.00 Byte)
```

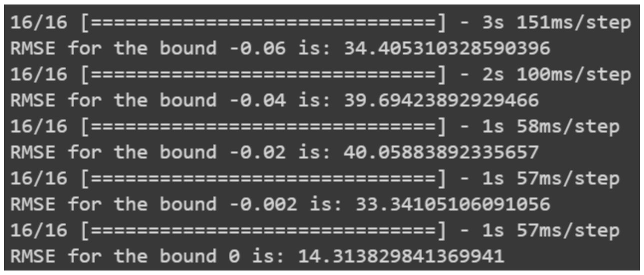
A plot between true RUL and predicted RUL is compared and found dl model performs fairly well with RMSE: 14.31.



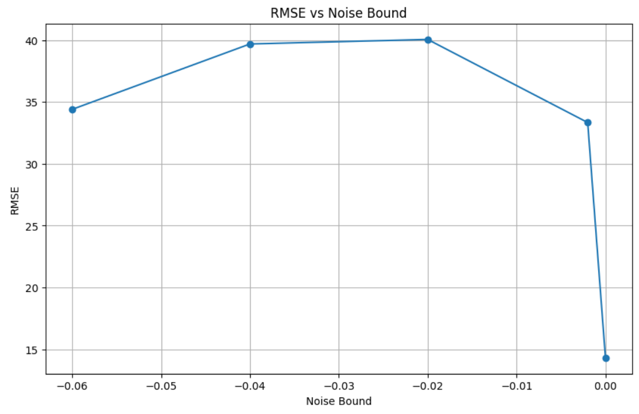
Injection Attacks

Injection attacks pose a significant threat to predictive maintenance models. In the case of FDIA, small alterations are made to the sensor data, simulating scenarios of data corruption. These attacks can subtly affect the model's predictions, highlighting the need for robust data validation and security measures.

To model the FDIA on sensors, a malicious vector is added to the original vector. This alteration modifies the sensor output by a very small margin—between 0.01% and 0.05% for random FDIA, and precisely 0.02% for biased FDIA. In this context, 'random FDIA' refers to the noise added to the sensor output, which can vary within the specified range. This process simulates real-world data inconsistencies by injecting noise into a dataset. Specifically, to establish a range of noise levels (bounds) that will be applied to certain sensor readings in the test dataset and RMSE is calculated.



```
16/16 [==============================] - 3s 151ms/step
RMSE for the bound -0.06 is: 34.405310328590396
16/16 [==============================] - 2s 100ms/step
RMSE for the bound -0.04 is: 39.69423892929466
16/16 [==============================] - 1s 58ms/step
RMSE for the bound -0.02 is: 40.05883892335657
16/16 [==============================] - 1s 57ms/step
RMSE for the bound -0.002 is: 33.34105106091056
16/16 [==============================] - 1s 57ms/step
RMSE for the bound 0 is: 14.313829841369941
```



Conclusion

The implementation of the LSTM RNN model and the simulation of injection attacks are demonstrated by Random False Data injection attacks. It includes data preprocessing, model training with optimized training process of a neural network model by dynamically adjusting the learning rate. This approach is indicative of a focus on fine-tuning the model for better performance. The code also showcases the process of iteratively adding noise to the sensor data and evaluating the model's performance under these altered conditions. Overall, the deep learning model in this case performs very poorly even by smaller addition noise by noticing the change in corresponding RMSE values.

GIT HUB Link: https://github.com/Vishnusairayulu/DL_Project.git

References

[1] V. L. L. Thing, "IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach,," *IEEE,* 2017.

[2] B. L. M. Y. Z. Y. Shaoqian Wang, "Intrusion Detection for WiFi Network: A Deep Learning Approach," *Springer,* vol. 264, 2019.