Predicting sensor measurement using time-series Deep Learning

Sharaj Panwar

1 Problem Statement

The goal of this research project is to create a time-series Forecasting model for predicting future sensor measurements from past measurements. Various candidate machine learning models are trained and their performances are compared with the baseline. The model with the lowest root mean square (RMSE) is selected while the train set, and test sets are created across sensors.

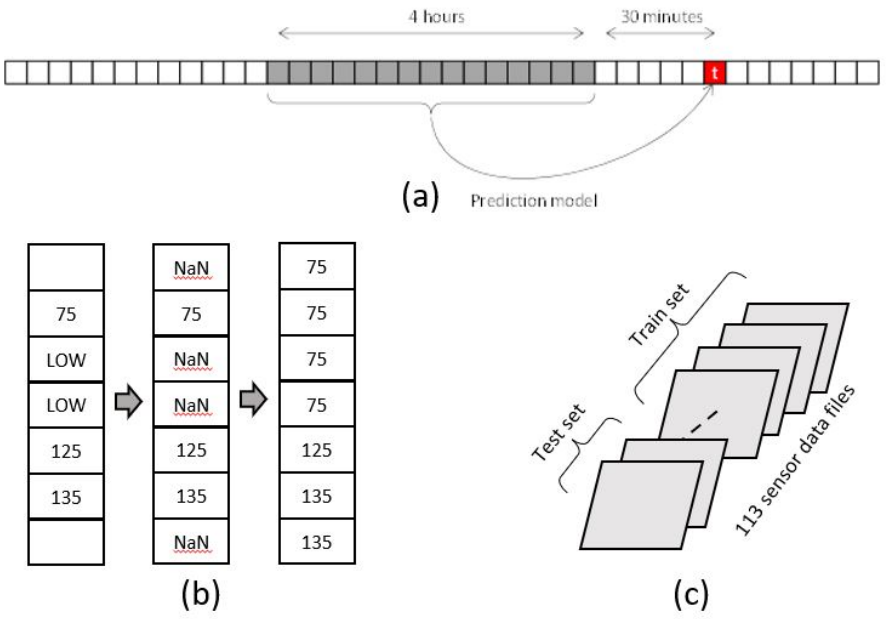

Figure 1: (a) Sensor Measurement Forecast from Past Measurements, (b) Data cleaning is performed by replacing character entries and null entries by previous/next available entries, (c) Out of 113 sensors' data, 90 randomly selected sensor' data is used for creating train set and the remaining 23 sensor's data is kept for test set.

2 Experiment and Results

2.1 Data set and Prepossessing

A sensor data containing 113 csv files is given. Each file has measurements from a separate sensor. The uni-variate measurements are recorded at a sampling rate of 5

minutes for about a month. Since our goal is to predict the sensor measurement at a future time step from the past measurements, we will prepossess the data to form a supervised regression problem. A time sequence of 4 hours measurements (48 time steps) will be used to predict the measurement at the 30 minute (6^{th} time step) in future as shown in Figure 1a.
The first step in this project is to create a data processing pipeline. In our pipeline, we can pass arguments for assigned variables to modify the 4 hours time sequence of predictor and/or 30 mins time step lead of forecast. This is useful for reproducibility as we may want to perform a comparative study on what should be the optimal time sequence length and how far away in the future we can get reasonable predictions. To achieve this, we first create a function which takes a sensor file as input and create predictor and label pair. We observe that some entries in the sensor files are missing while some are character entries such as an entry being 'LOW'. These entries will adversely affect the training and need to be handled. Within this function itself, we first convert all the character entries to NaN (Not a Numeric). Thereafter we replace these NaN values by the first previously available entry and whenever a NaN value is the first entry in the sensor file we replace it by the next available entry as shown in Figure 1b.
In our next step, we create another function which randomly picks 90 sensor files (approx. 80%) for creating train set and remaining 23 files for creating test set as shown in Figure 1c. The cross validation across sensors prevents intra-sensor information leak and is crucial for modelling inter-sensor variability. We shall repeat the train and test split to report the average performance of our predictive model. Therefore, the function takes an argument to assign the random state (seed) for random selection of sensor files for train and test split. The train and test sets are saved with the corresponding seed value appended to their file name for identification and recreation. Finally, we have our last function in the pipeline which takes an argument for number of training and test sets we want to create, and generate that many random states. By iteratively assigning these random states we can create multiple training and test sets dynamically. With the creation of this pipeline, we have high flexibility in curating our data by the assignment of few parsing arguments and without having to change the codes.

2.2 Deep Models: LSTM

The time-series forecasting problem concerns with modelling the temporal information for future time step prediction. Some of the candidate models are LSTM, Attention Network, and Transformer model. To being with, we are training LSTM model as our baseline model. The size of our processed training data is (1573979, 48) which needs to be handled with a considerably deep model and longer training epoch for convergence. Several LSTM networks are explored for varying model depth, width and training epochs. The RMSE is reported in Table 1. It can be observed from the results of Table 1 that a larger LSTM with 6 layers and 50 neurons each, trained for 50 epochs has the best performance so far. This Model achieves the minimum RMSE of 37.96. However, it does takes longer time to train as compared to other architecture.

We also visually analyze and compare the performance of our best two models from S.No 4 and S.No 5 in Figure 2 and Figure 3. We show the sensor measurement forecast from our trained models compared to the ground truth values of corresponding

Table 1: RMSE Based LSTM Model Architecture Selection

S.No.	Nos. of LSTM layers	Nos. of Neurons	Training Epochs	RMSE
1	3	30	25	190
2	3	50	25	60.066
3	5	50	25	48.59
4	4	50	50	44.65
5	6	50	50	37.96

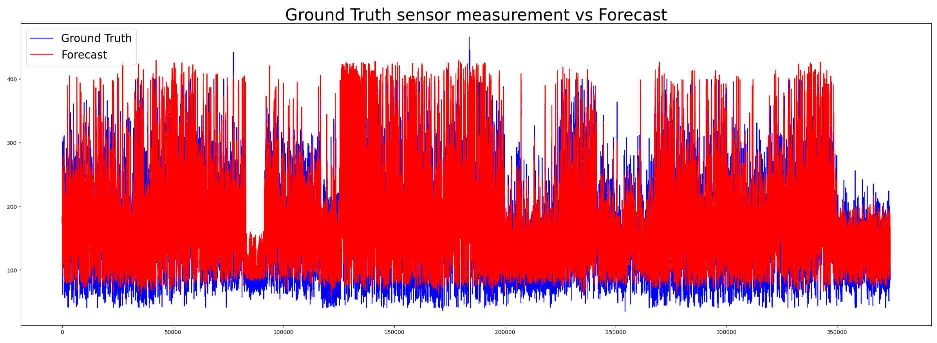

Figure 2: Sensor measurement forecast of model 4 vs ground truth measurements

measurements in the test set. We can see that the predictions from model 5 are closer to the ground truth values as compared to the model 4 predictions. Which is consistent with the quantitative results of Table 1. The current modelling challenge concerns with the considerably long training time taken for hyper-parameter tuning. With the finalized architecture and data prepossessing pipeline for creating train and test sets across sensors, we will be able to create several models to report average performance. The future work is to explore transformer models and other candidate machine learning models to choose the best model.

2.3 Deep Models: Transformer

-to be written-

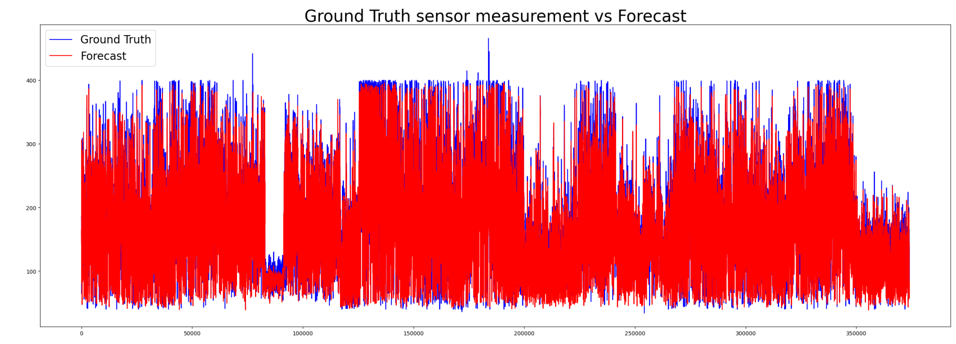

Figure 3: Sensor measurement forecast of model 5 vs ground truth measurements