LaDyBrain:
Predicting Kalman Denoising Variables for Car Lateral Dynamics using Long Short Term Memory Recurrent Neural Networks

Cristina Todoran

Abstract

- LaDyBrain is a Python program using TensorFlow library for implementation of recurrent neural networks based on Long Short Term Memory (LSTM) cells for prediction of Kalman denoising variables used for filtering the signals and computations of the lateral dynamics of a car
- After the training and the validation, LaDyBrain is able to predict for new tests, the target values

Motivation

- LaDyBrain is designed to replace day by day work of users, which consists in finding the best Kalman pair which models the system noise, using data from cars, especially different tests or crashes, e.g., double lane change, fishhook, slalom
- The purpose of the software is to replace human actions by computer predictions in order to achieve more accurate results in a more efficient and powerful way

Problem Definition and Goals

- Lateral Dynamics (LaDy) is part of the Integrated Vehicle Dynamics algorithm which calculates the lateral velocity of a car, based on physical parameters and on a mathematical model
- Noise reduction can be defined as the process of removing noise or perturbations from a signal
- Kalman filter algorithm runs in real time on the embedded system of the car and it is implemented in the LaDy algorithm for the calculation of the lateral velocity. The system noise will be modelled by a constant matrix Q
- LaDyBrain learns to predict using Long Short Term Memory Recurrent Neural Networks the two Kalman denoising variables used in the LaDy algorithm, the system noise elements Q00, Q11 are calibration parameters and have the following influences:
  - Q00 – system noise which has influence on Lateral Velocity
  - Q11 – system noise which has influence on friction coefficient

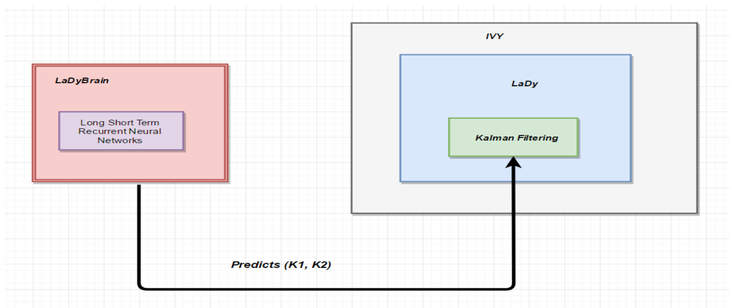

Figure 1: LaDyBrain interaction with LaDy

$$Q = \begin{bmatrix} Q_{00} & 0 \\ 0 & Q_{11} \end{bmatrix}$$

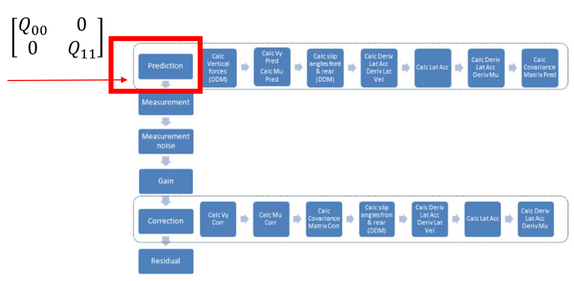

Figure 2: Kalman predicted variables for denoising

Data

- The input to the program is represented by different type of tests, recorded from cars: slalom driving, fishhook driving, sliding on ice, normal driving
- Data was structured as a matrix with 9 columns (features): Steering Angle, Wheel Speed Front Left, Wheel Speed Front Right, Wheel Speed Rear Left, Wheel Speed Rear Right, Lateral, Acceleration (XLow), Longitudinal Acceleration (YLow), YawRate Reference Vy

Type	Train (K1,K2)	Validation (K1,K2)	Test (K1,K2)
All Crashes	847 (0.1, 0.1)	36 (0.1, 0.1)	unknown
Bl_real	63 (0, 7)	10 (0, 7)	unknown
Cm_real	7 (0, 3.7)	1 (0, 3.7)	unknown
Dlc	129 (0, 1.4)	13 (0, 1.4)	unknown
Fishhook	165 (0, 5.8)	20 (0, 5.8)	unknown
Fishhook_real	18 (0, 5.8)	3 (0, 5.8)	unknown
Fluchtwende	69 (0, 0.3)	13 (0, 0.3)	unknown
J_turn	129 (6.4, 0.1)	13 (6.4, 0.1)	unknown
Lane_real	9 (0, 0.2)	2 (0, 0.2)	unknown
Slalom	165 (0, 8.5)	25 (0, 0.2)	unknown
Slalom_real	13 (6.6, 1.7)	2 (6.6, 1.7)	unknown
Weschselboeschung	129 (0, 6.8)	20 (0, 6.8)	unknown
Mixed_10_of_a_type	660 [(0, 0.1),(0.5, 8.9),(2.5, 9.3),(0, 2.8), (0, 8.8),(0, 0.3),(0, 8.8),(0, 1.6),(0, 7.8),(0, 3.8)]	66 (0,2.8)	unknown
Mixed_50_of_a_type	300 (0, 7.8)	50 (0.1, 0.1)	unknown
Real_Crashes	140 (0, 7)	6 (0, 7)	unknown
Unknown (new tests)	-	-	24 (unknown)

Table 1: Training Data, Validation Data and Testing Data

Model

- Recurrent neural networks are especially useful for sequential data, because each unit or neuron can use its memory to remember previous input
- These signals (data) recorded from cars are time series, at each n step, depending on the values at step n-1
- An innovative idea to recurrent neural networks was the Long Short-Term Memory (LSTM). This architecture was developed for a specific purpose, to address the “vanishing and exploding sensitivity”

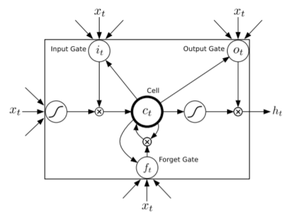

Figure 3: LSTM Unit

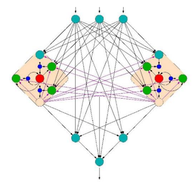

Figure 4: Mix LSTM cells and others

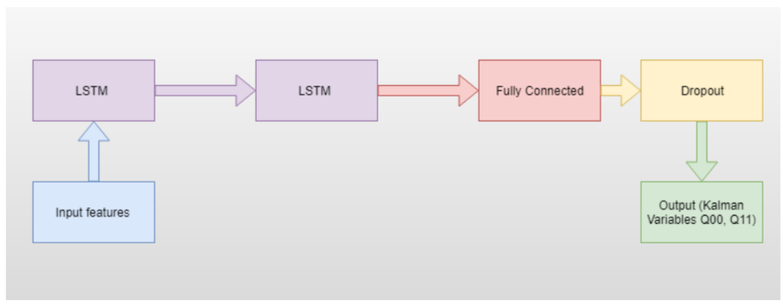

Figure 5: Two layers LSTM model

Results

- Backward pass through the network, determines which weights contributed most to the loss and find ways to adjust them so that the loss decreases
- For updating the weights, the learning algorithm used was Adam Optimizer with learning rate 0.001
- Training: 350 epochs
- Validation: at each 50 epochs
- Test : predicted for new tests with an accuracy of over 85%

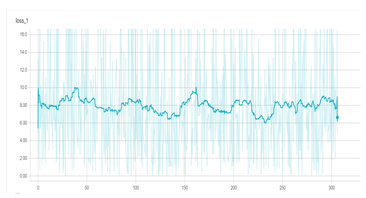

Figure 6a: Training Loss

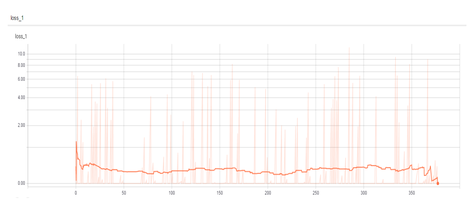

Figure 6b: Validation Loss

Conclusions

- Because the signals are very long (even 400 000 time steps) in the implementation LSTM cells were used in order to avoid vanishing gradient or overfitting
- It may be possible that the neural network to gain more knowledge from the data and later the users could find the parameters by using the patterns found by the neural networks. An outcome is that the neural networks will find patterns in the signals that the calibrators could not observe
- As a result we obtained a model able to predict the target values for new data which will be recorded

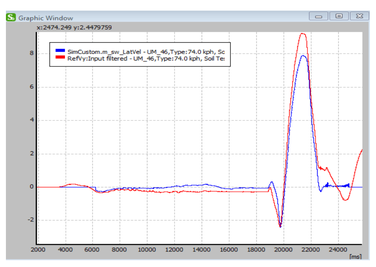

Figure 7a: Calibration using predicted Kalman Variables

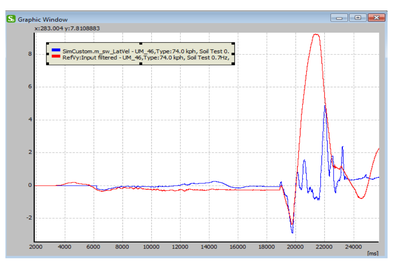

Figure 7b: Calibration without Kalman Variables