Budapest University of Technology and Economics
Faculty of Transportation Engineering and Vehicle Engineering
Department of Control for Transportation and Vehicle Systems

Traffic stability prediction using federated learning (supervised ML)

Traffic Modelling Simulation and Control
Homework

Authors:
Szabó Anna - J8GU0D
Kürti Ádám - HZUCDC

Consultant:
Wágner Tamás

2024

Contents

Project summary

20. Traffic stability prediction using federated learning (supervised ML):

• Collect and monitor traffic at multiple junctions using SUMO (travel time, traffic density, traffic flow, vehicle mean speed etc.)

• Create a stability classification model using federated learning (supervised machine learning)

Knowing if the traffic is stable on a road section can be highly beneficial for several reasons. It helps in optimizing fuel consumption and reducing vehicle wear and tear by avoiding frequent stop-and-go driving. For city planners and traffic management authorities, understanding traffic stability aids in making informed decisions regarding infrastructure improvements, traffic signal timings, and congestion management strategies. Additionally, businesses relying on timely deliveries can enhance their logistics and distribution efficiency by choosing routes with predictable traffic patterns.

Based on the stability measurement also traffic signal programs have to be changed in order to prevent or correct instability.

Problem statement

One of the prerequisites of determining road stability is knowing the mean vehicle speed on the road section. However, in real life vehicle speed measurement is troublesome: it requires expensive hardware e.g. radar or laser sensory based equipment or multiple inductive loop detectors built into the road. These solutions can provide more accurate results than cheaper video-based solutions.

Measuring the number of bypassing vehicles in a road section is easier. This can be done with high precision using video cameras providing a cheaper method of vehicle surveillance on a given section.

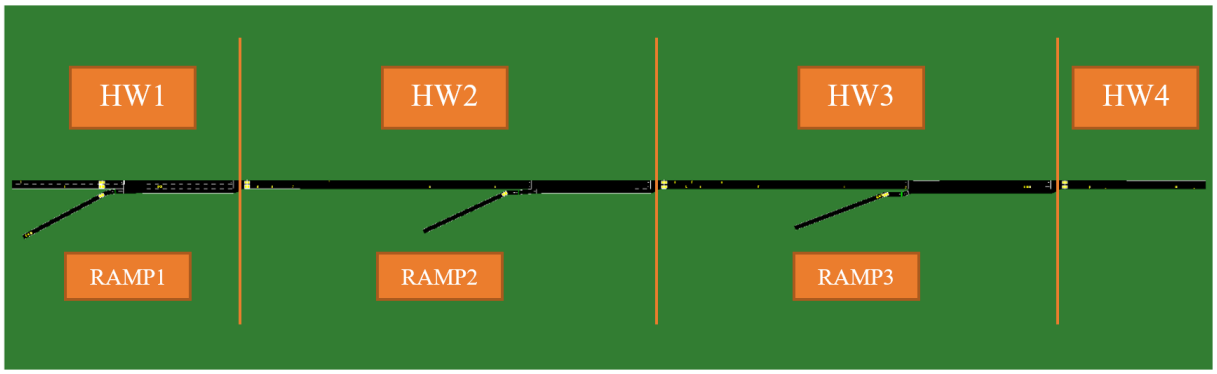
Proposed solution

We approach stability classification with a neural network based solution. We first generate labeled data for its training for which we use the SUMO simulation environment. With the acquired data we train the network in a supervised manner using federated learning. Lastly we evaluate the network based on a portion of the dataset, we draw conclusions from it and present further improvement ideas.

Simulation setup

For data generation we used the Sumo simulation environment. The created highway road section can be seen in the picture below. There are 4 highway segments with 3 ramps, where every ramp ends with a traffic light. In each highway segment, there is a loop detector in every lane (2 loop detectors for every highway segment) and there is a loop detector also at

the end of the ramps before the traffic lights. The elements of the highway are defined in the *highway.net.xml* file and the loop detectors are in the *additionals.add.xml* file .



Every traffic light has 4 programs:

1. Green phase duration: 60 sec,
   Yellow phase duration: 5 sec,
   Red phase duration: 25 sec
2. Green phase duration: 30 sec,
   Yellow phase duration: 5 sec,
   Red phase duration: 25 sec
3. Green phase duration: 45 sec,
   Yellow phase duration: 5 sec,
   Red phase duration: 10 sec
4. Green phase duration: 45 sec,
   Yellow phase duration: 5 sec,
   Red phase duration: 40 sec

Note that the yellow light is usually not used in highway ramps, the traffic lights contain only the red and the green lamps, but in the simulation it was easier to implement a simple traffic light with three colors. Since the yellow time is not used in the training process, the three colored traffic light does not have any impact on the training data. The traffic light programs are defined in the *additionals.add.xml* file and the fourth one is in the *highway.net.xml* file.

The type of the cars on the highway and the routes are defined in *routes.rou.xml* file. There are four possible routes, one from the beginning of the highway to the end of it (HW1 - > HW2 -> HW3 -> HW4). Moreover, one route starts at every ramp and goes along the highway (e.g., RAMP1 -> HW2 -> HW3 -> HW4).

Data collection

The output of the simulation is the data from the loop detectors and the actual traffic light programs. The simulation results are collected in a CSV file with the following columns:

- The green, (yellow) and red phase duration of the traffic lights (for each traffic light 3 different columns).
- The flow intensity values at the beginning of the highway and at each ramp.

- The mean speed and the number of the vehicles in the 1st to 4th highway segment in every lane (every highway segment has 4 data columns) and in the ramps.

The above mentioned first two values are the inputs of the simulation while the last one is the measured output.

The main idea of the data generation was to simulate and measure the traffic intensity in the highway sections with different initial vehicle flows (both from the highway section 1 and from the ramps) and with different traffic light programs. To change the vehicle flows and the traffic light programs, and to get the measured data from the loop detectors, we used the TraCI Sumo interface.

The initial inputs of the simulations are the above-mentioned traffic light programs, and the initial traffic flow at the beginning of the highway and on the ramps. The average of the measured data (mean speed and number of vehicles) is saved every 10 minutes to the CSV file.

At first the initial highway traffic flow was set to 200 vehicle/h and in every 10 minutes, the traffic light programs are changed randomly (can be a different program in every traffic light). In every 30 minutes, the ramp flow intensities are changed and in every 60 minutes, the highway flow intensity is changed randomly. The random changes are controlled in a way that the ramp flow intensities are randomly chosen from 10 equal intervals from 40 to 400 vehicles/h while the highway flow is chosen from 10 equal intervals from 200 to 2000 vehicles/h. Furthermore, the traffic light programs can be chosen from the above described 4 traffic programs.

At the second simulation, the initial highway flow was 400 vehicles/h and the 10 equal intervals are created from 400 to 4000 vehicles/h. This second simulation execution was necessary because in the first one there was a small amount of unstable traffic situations and the training process has led to overfitting.

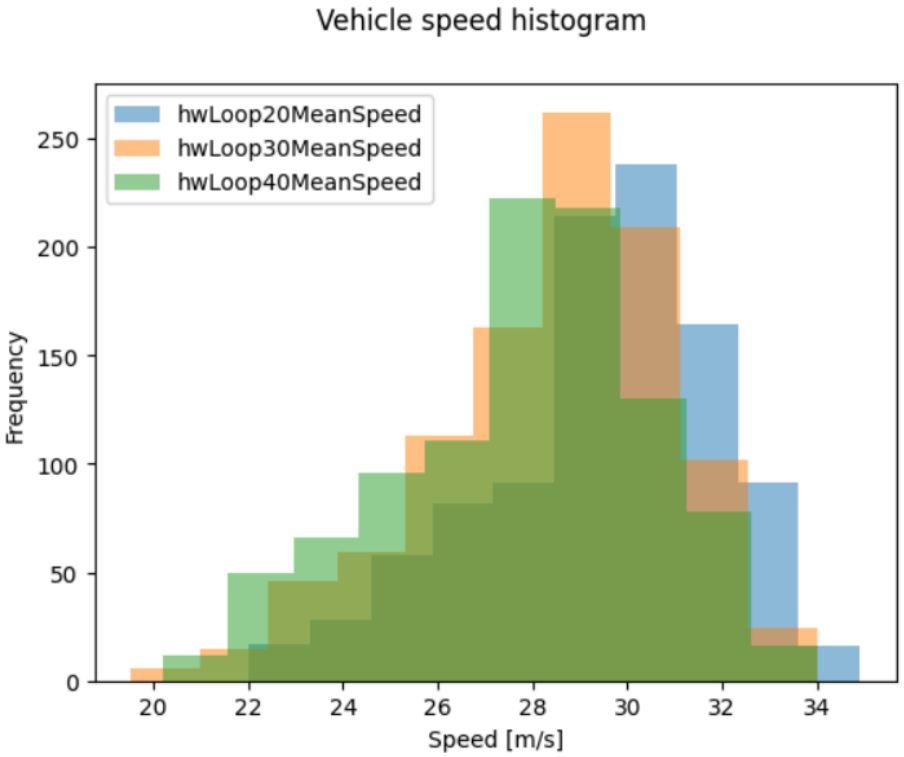
The output file was used to create both the training data and the labels to the federated learning. While the traffic light's green, yellow and red phase time per ramps and the traffic flows per ramps (plus the previous highway section) are the inputs of the neural network, the mean speed and the number of the vehicles in the 2nd, 3rd, and 4th highway sections are provide the input data to create labels for the federated learning.

Data preprocessing and analysis

Before training we analyze and preprocess the acquired data. For this mainly pandas, numpy and matplotlib is used.

As the first step of preprocessing, we plotted the acquired vehicle speed values to analyze its distribution, which is shown on the figure below. It can be seen that the distribution mean speed values gathered are overlapping with each other, but as the index grows i.e. we go further into the highway section, the distribution shifts to the left slightly, indicating that because of the ramps inflow of traffic that highway gets more saturated. The peak of the curves are at approximately 29 m/s. We set the threshold value for stability to this point: samples under this value are considered unstable, above it stable. With this configuration we got the following stableness ratios:

- Dataset1: 37% stable, 63% unstable
- Dataset2: 49% stable, 51% unstable
- Dataset3: 66% stable, 34% unstable



We diveded these 3*999 samples into 3 Dataset, specifically 3-3 train and val dataset for the clients with a validation set ratio of 0.2 resulting in a ratio of 48% stable in the validation set and 51% stable samples in the training set.

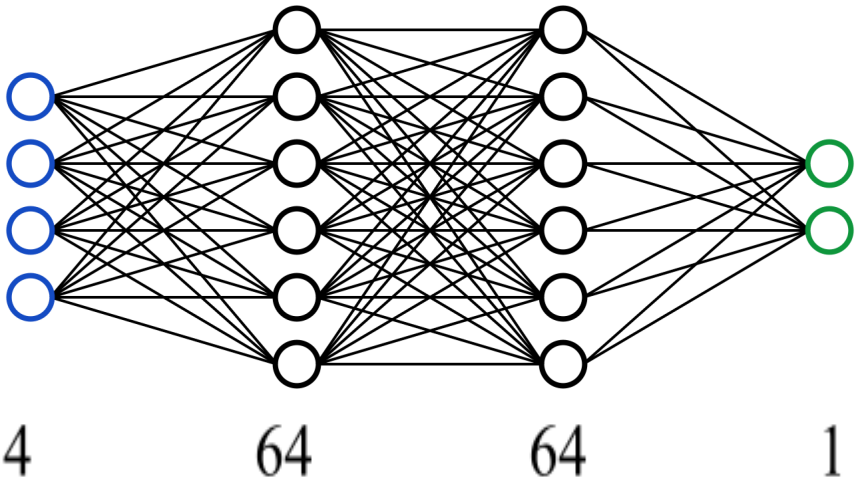
We then normalized the data to an interval of [0,1] for each column. This helps the training by preventing relatively big numbers to affect the parameter optimization disproportionately.

Neural network architecture

For the neural network we use a general fully connected network. It consists of 4 layers:

- Input layer with shape [4,1]
- First hidden fully connected layer with 64 neurons
- Second hidden fully connected layer with 64 neurons
- Output layer with 1 neuron

We use ReLU nonlinear activation function between layers to facilitate learnability. For regularization purposes we also use a dropout layer per stage. Sigmoid activation function is used at the output layer for determining stability over a 0.5 threshold value.



Layer (type)	Input Shape	Param #	Tr. Param #
Linear-1	[1, 1, 5]	320	320
ReLU-2	[1, 1, 64]	0	0
Linear-3	[1, 1, 64]	4,160	4,160
ReLU-4	[1, 1, 64]	0	0
Linear-5	[1, 1, 64]	4,160	4,160
ReLU-6	[1, 1, 64]	0	0
Linear-7	[1, 1, 64]	65	65

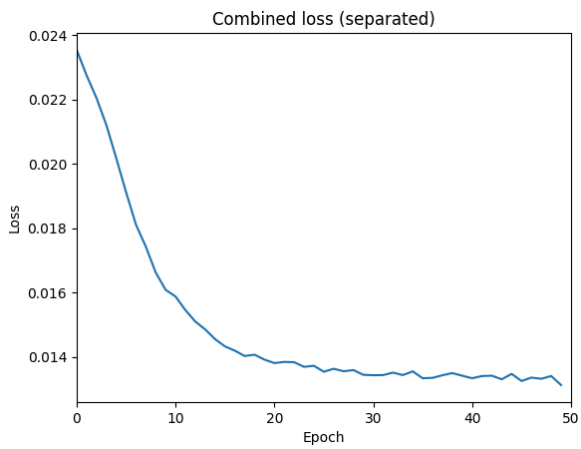
Total params: 8,705
Trainable params: 8,705
Non-trainable params: 0

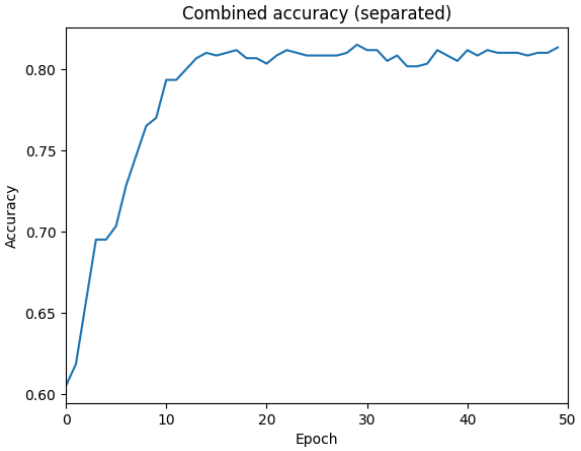
Figures above show the architecture of the network used. It has 8705 trainable parameters resulting in a small network with short inference times.

For simulating a real life scenario where each onboard computer built into the traffic signal controllers trains on acquired data in real time, we used federated learning. Unlike traditional centralized machine learning, which requires aggregating data from multiple sources into a single location, federated learning allows models to be trained across decentralized devices or servers where the data resides. We only used one computer for this, but with a simulated distributed training with distributed data using the Flower federated framework. For the federated training strategy we use FedMedian strategy which proved to be the best among the strategies available in Flower.

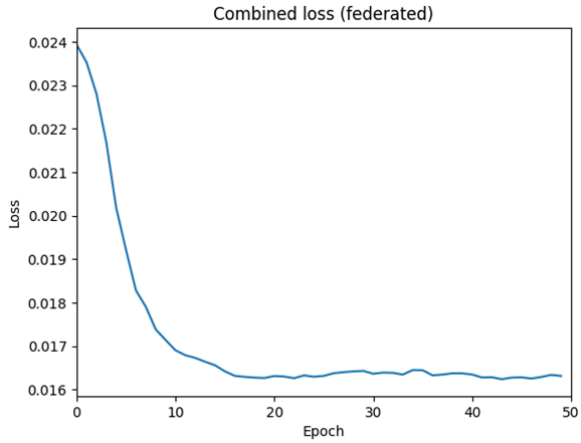
Training and evaluation

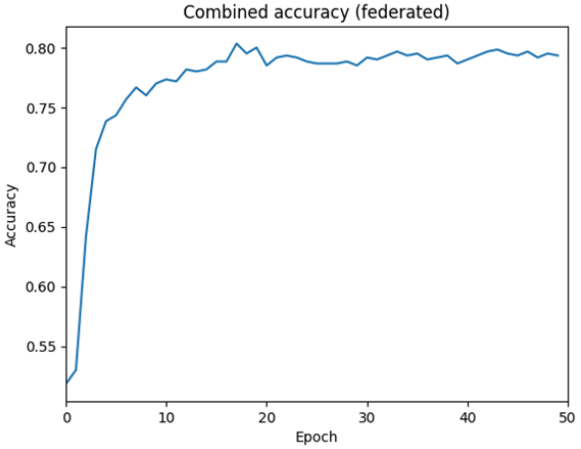
The implementation of the neural network pipeline was done in Python, with PyTorch. We trained the models over 50 epochs and with 3 clients. The models were trained using binary cross entropy loss function and Adam optimizer.





First, as a test we train 3 separate models with 1 dataset each. Above can be seen the loss and validation accuracy results plotted during the training. It is calculated as the mean of the 3 agent's results for every epoch.





Using federated training we got the following results. It is calculated as the weighted average of the accuracy and loss between the 3 agents.

One can observe that the results with federated learning are slightly worse. This is balanced by the fact that the training with a federated setup can be much faster and can be done on cheaper hardware rather than having to centralize all data for a more complex and thus expensive training machine.

Further improvement ideas

Further development ideas include:

- Gathering more and more diverse data
- Equipping the neural network with recurrent layers such as GRU or LSTM