Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

Università di Pisa

Attacks detection in Cyber-Physical Systems with Neural Networks: a case study

Cinzia Bernardeschi, Gianluca Dini, Maurizio Palmieri, ***Alessio Vivani***

Department of Information Engineering
Unversity of Pisa Pisa, Italy

FORESEEN Project
PRIN PNRR 2022

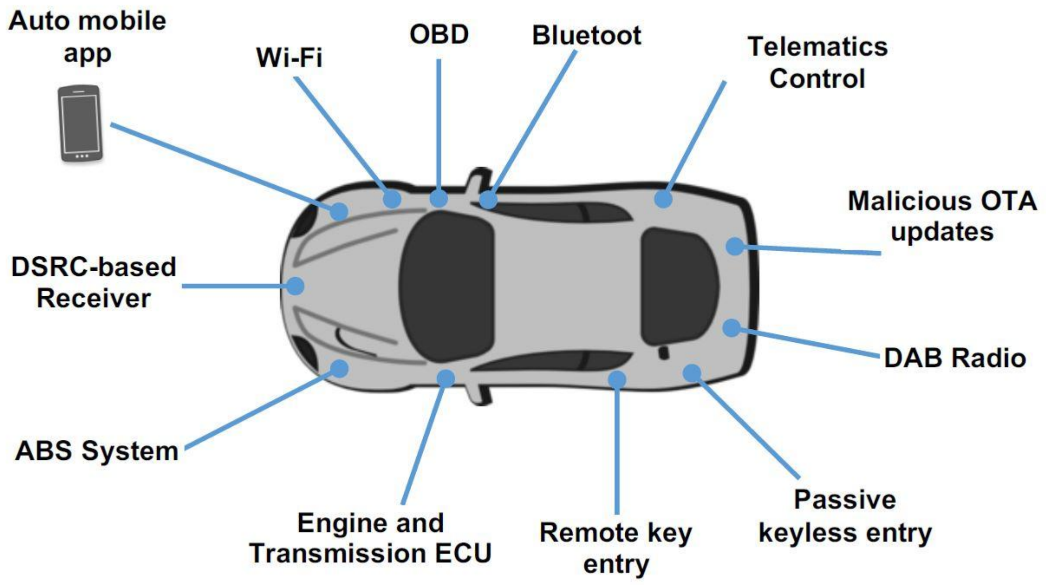
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
UNIVERSITÀ DI PISA

Summary

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
UNIVERSITÀ DI PISA

1. Introduction – Cybersecurity in CPSs

- Cyber Physical Systems are complex systems, genrally composed of a plant and controller, that interacts with the external environment.
- Modern cars are highly computerized systems, often connected to a network, thus making them vulnerable to cyber attacks.
- State of the art clearly highlight a large set of threats, for example an attacker could gain access leveraging the OBD port, Bluetooth connection or even the CD player.
- For those reasons, addressing *Cybersecurity* measures is necessary to ensure *safety*.



1. Introduction – Research Objective

Main objective of this work:

define a methodology and develop a framework for the detection of attacks in a CPS with a machine learning algorithm.

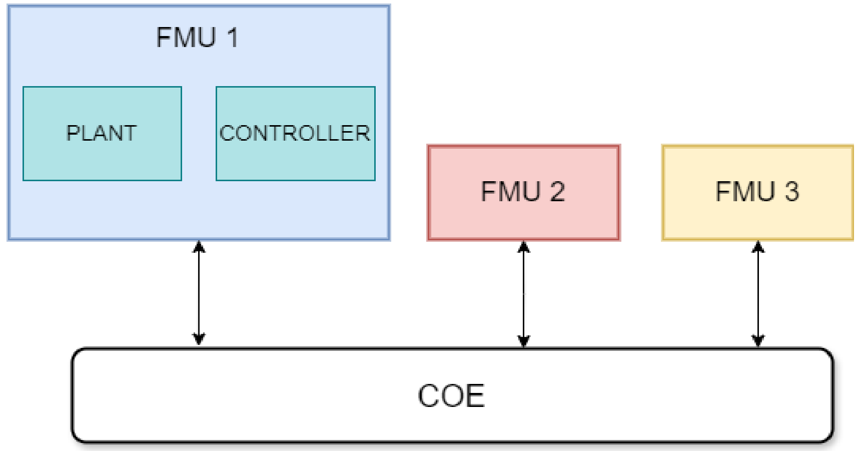
In particular,

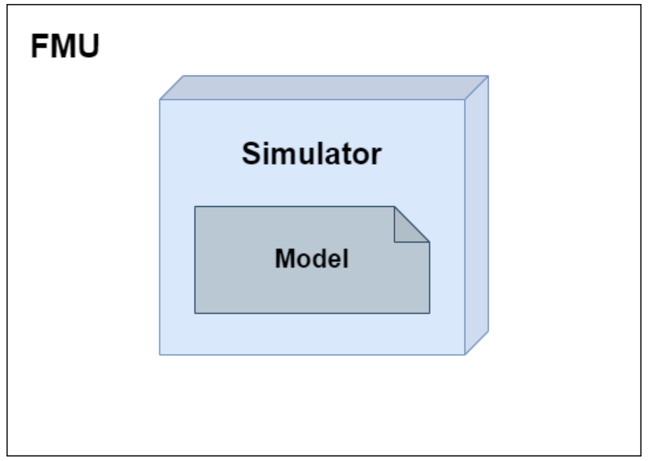
- the methodology exploits a software replica in co-simulation to gather data used for training and testing attack detection models based on machine learning algorithms.

- results obtained on a case study shows high accuracy in detecting attacks on the cyber-physical system.

2. Background – Co-Simulation

- In model-based design, a commonly used approach to simulate cyber-physical systems is the use of co-simulation.
- Co-simulation is a flexible and modular solution that allows to globally simulate a system composed of many smaller components that can be developed in different modelling environments and languages.
- Functional Mockup Interface is a standard for Co-Simulation, where the key components are the Functional Mockup Units (FMUs)
- Each FMU models the behavior of a component of the physical system.
- FMUs are coordinated by a Cosimulation Orchestration Engine.



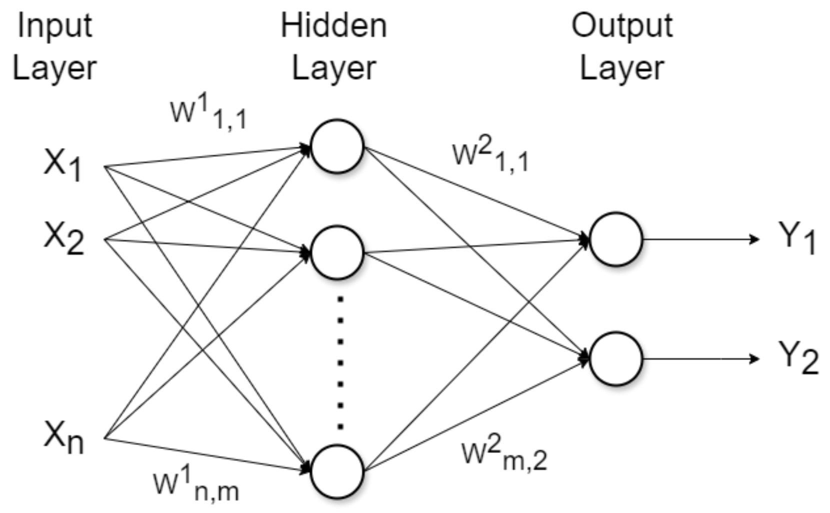


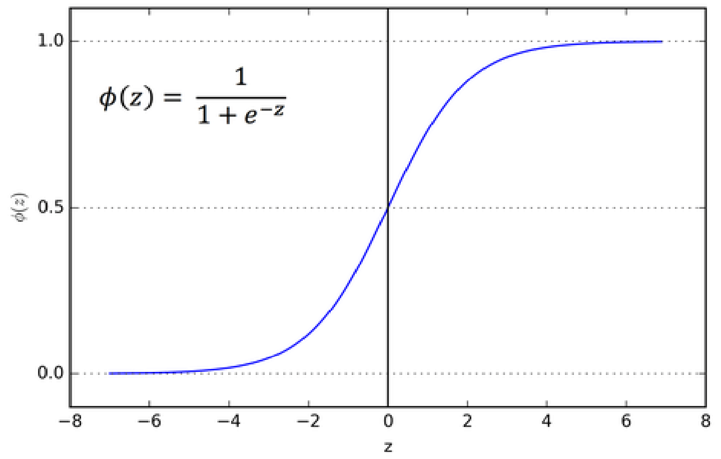
2. Background – Neural Networks

- Deep Learning is a type of Machine Learning process which comprehends Neural Networks. An example of neural networks are Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs).

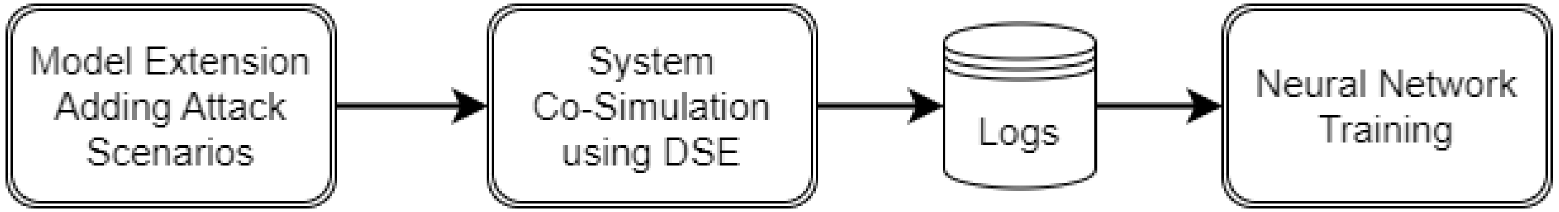
- Each neuron has an activation function that takes the weighted sum of the inputs to generate the output, a commonly used function for Classifiers is the Sigmoid.

- Both MLPs and CNNs can be used to solve many problem, such as for example classification.





3. Methodology - Overview


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

Main steps of the methodology:

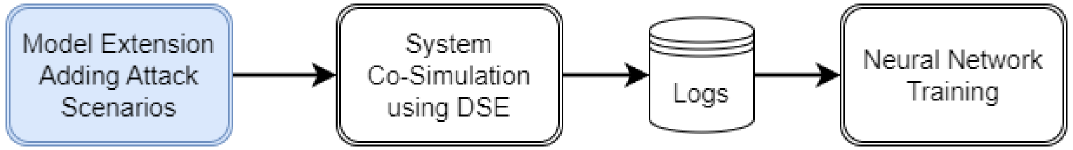
- Model extension adding attack scenarios
- System Co-simulation using DSE & data gathering
- Neural network training & testing

The chosen tools for the above steps are:

- **INTO-CPS :** An toolbox for co-simulation. It was selected since it provides a *Design Space Exploration* (DSE) tool.
- **MATLAB :** More precisely, *Simulink* for the modeling and extension phase, *Matlab* for the training phase.

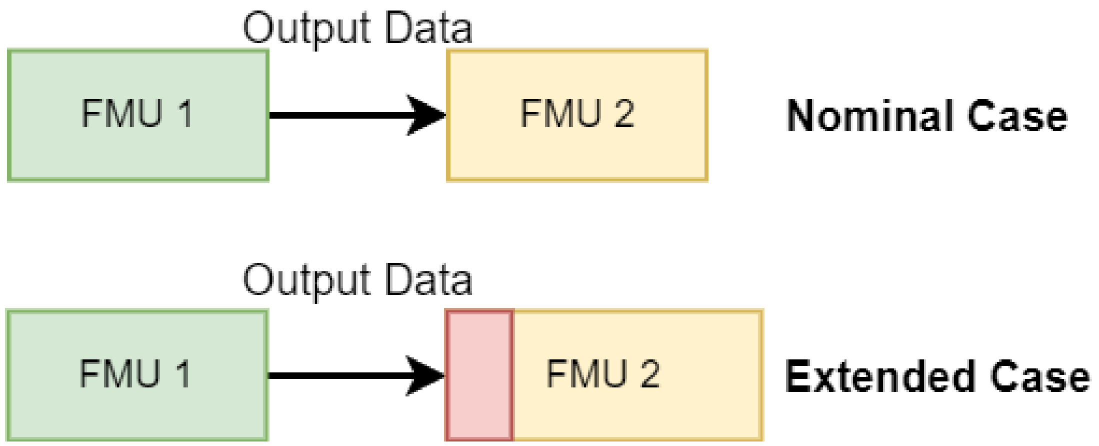
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università di Pisa

3. Methodology – Model Extension

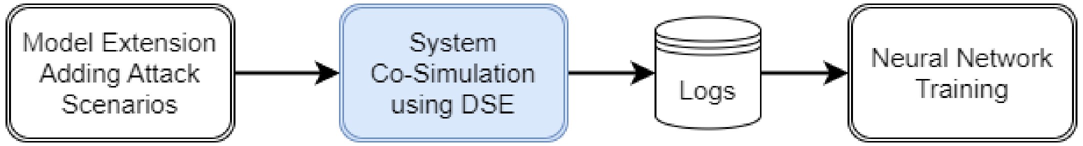

Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

Data-alteration attacks

- The first step is the extension of model, to include the effects of attack scenarios. In our case we take into account data-alteration attacks.
- Extend one or more pre-existing FMUs, adding a method to model the effects of an attack.

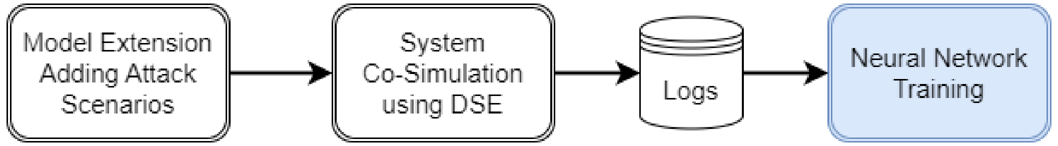


3. Methodology – Co-Simulation and Data Gathering


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

- INTO-CPS provides the Design Space Exploration tool (DSE) that allows to automatically run a large number of simulations.
- The DSE tool will create each simulation configuration taking data from a file that contains the list of values for each parameter of the system (initial state related parameters etc).
- The configuration file is customizable.
- At the end of each run, the DSE tool will save the timeseries of the state evolution in a csv file

3. Methodology – Training


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

- To be able to generalize and allow a good learning process for the neural network, a *large* dataset composed of data derived from *uncorrelated* scenarios is preferred.
- The configuration file is filled with randomly generated values for parameters of interest, such as the attack's intensity or the initial state of the system.
- The set of values for each parameter needs to stay between an acceptability range according to the physical limitations of the system.

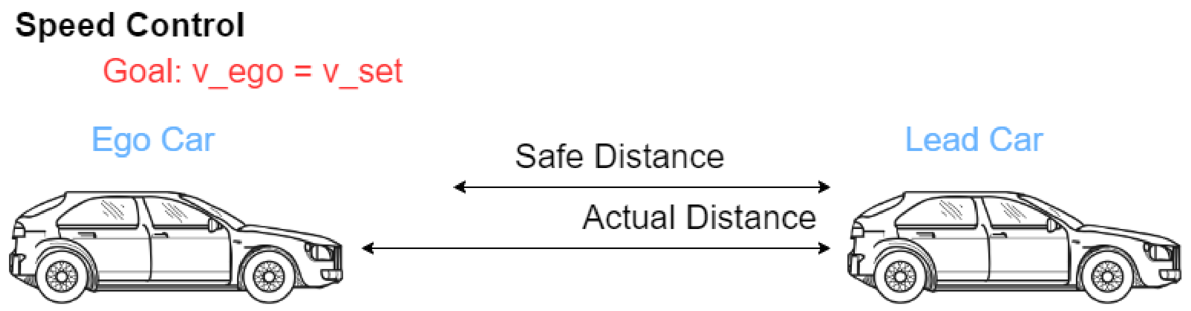
4. Case Study – Adaptive Cruise Control System using Model Predictive Control

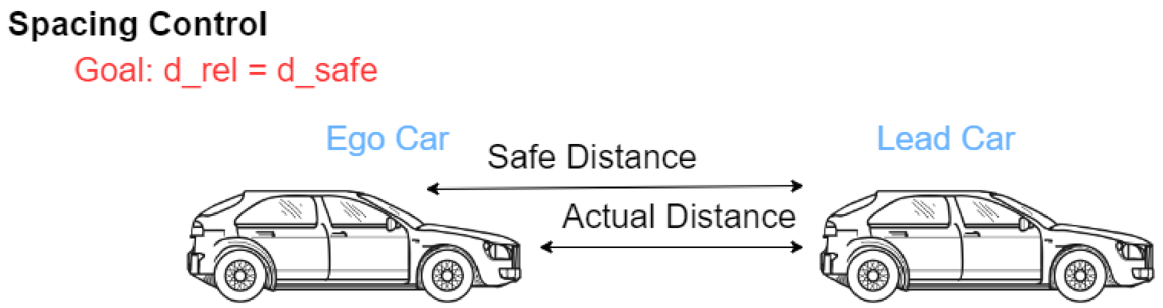
The Ego Car has a set of sensors to measure distance and velocity from the Lead Car.
The Ego Car uses the Adaptive Cruise Control (ACC) with the measurements to follow the Lead Car.

Two operational modes:
- Speed Control (Driver-set velocity **v_set**)
- Spacing Control (Safety distance **d_safe**)

Leader:
- Acceleration function modeled with a sine function
- Amplitude, Frequency, Phase and Offset changed during simulation

Ego Car:
- Safe distance changes depending on the current speed
- d_safe = d_default + t_gap * v_ego
- d_default is the standstill default spacing
- t_gap is the time gap between the vehicles



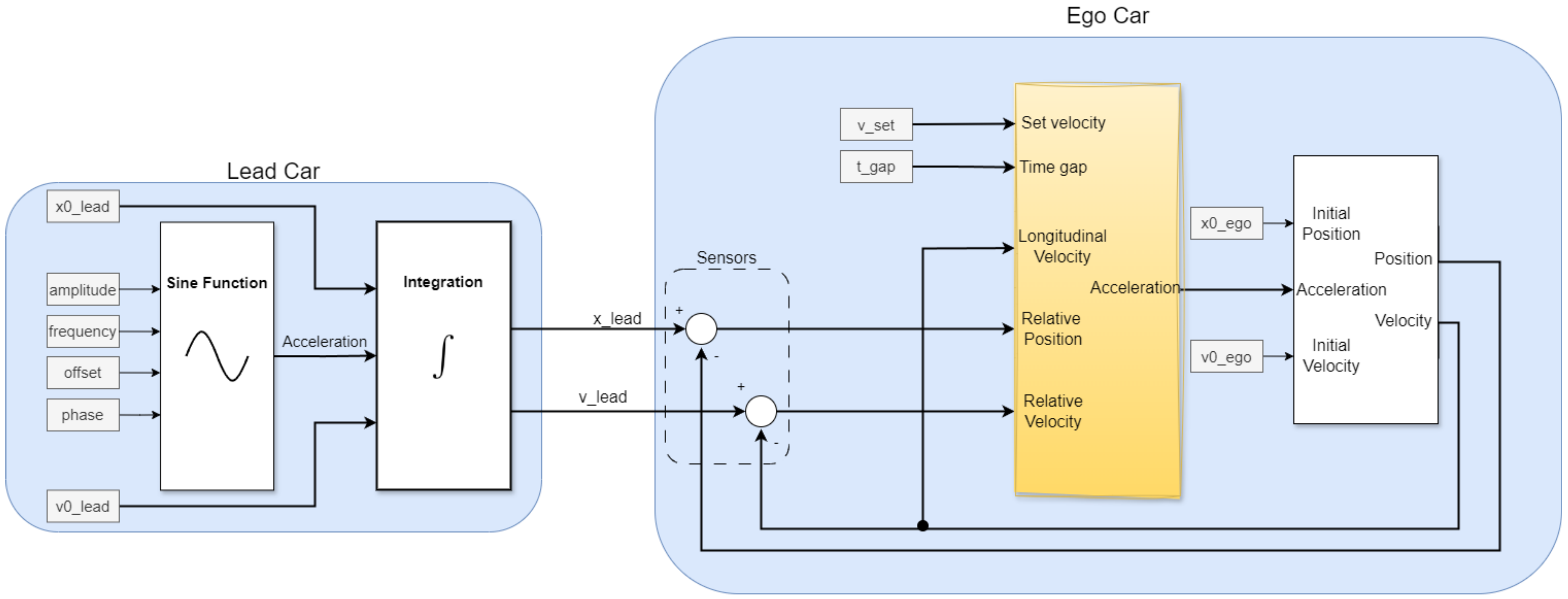


mathworks project, URL: https://www.mathworks.com/help/mpc/ug/adaptive-cruise-control-using-model-predictive-controller.html

Finanziato dall'Unione europea
NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università di Pisa

4. Case Study – Baseline Model

Simulink Model of the Project



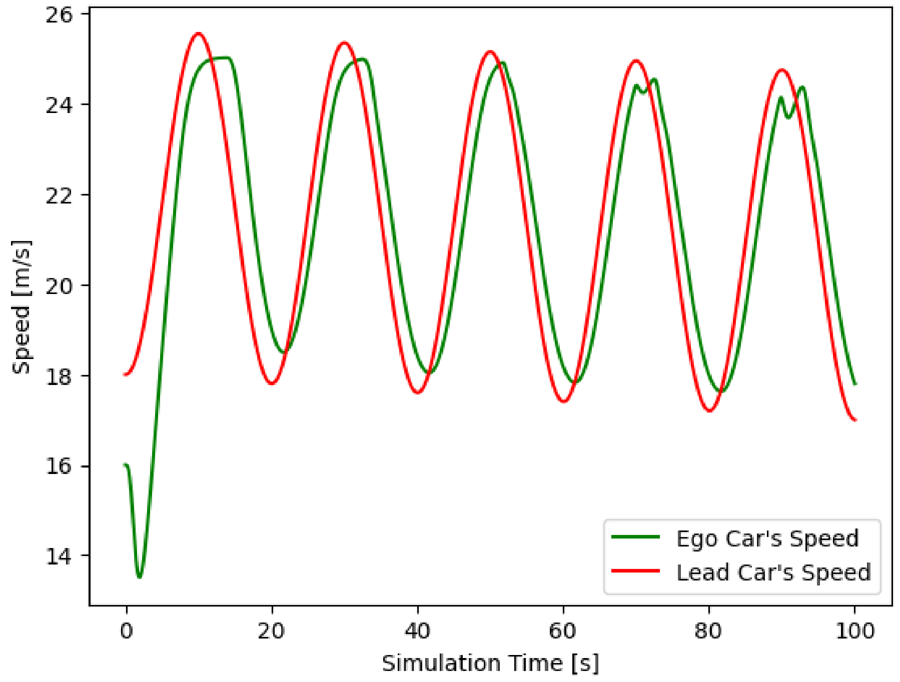
4. Case Study – Nominal Case Simulation

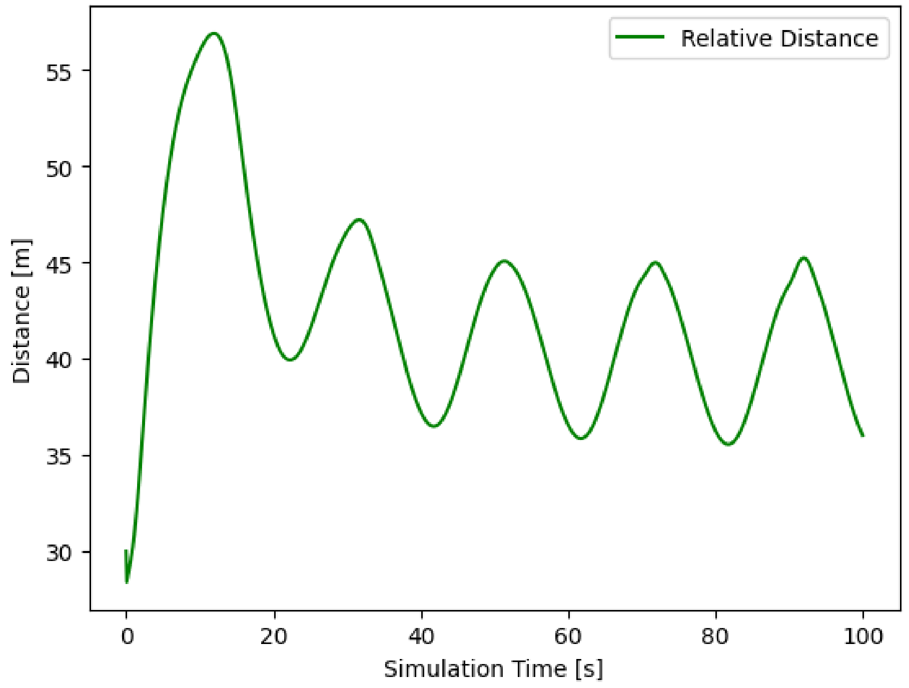
Ego Car:

- t_gap = 1.4 s
- v_set = 25 m/s
- v0_ego = 16 m/s
- x0_ego = 0 m

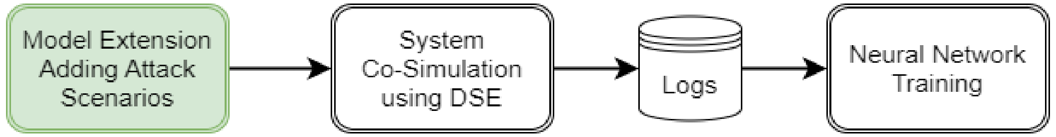
Lead Car:

- v0_lead = 18 m/s
- x0_lead = 30 m
- frequency = 0.05 Hz
- offset = - 0.01
- amplitude = 1.2
- phase = 0.0





4. Case Study – Ego Car Extension, Attacks


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

- Attacks were modeled by introducing an «*Attack*» method within the pre-existing Ego Car model.

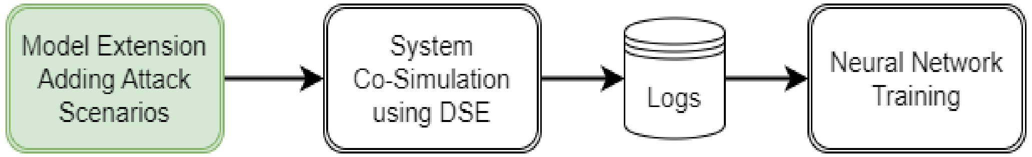
- Attacks are represented by a simple and generic equation: ***modified_value = real_measurement ± offset.***
  Each attack have a *start time* and and *intensity* parameter, set at configuration time.

- The offset is computed from three different attack functions:

  - Step Function: Starts at *start_time* and has a height equal to *intensity*.
  - Ramp Function: Starts at *start_time*, the slope is equal to *intensity* and at *start_time* the value is 0.
  - Sine Function: Starts at *start_time*, the amplitude is the *intensity* and the argument is the simulation time.

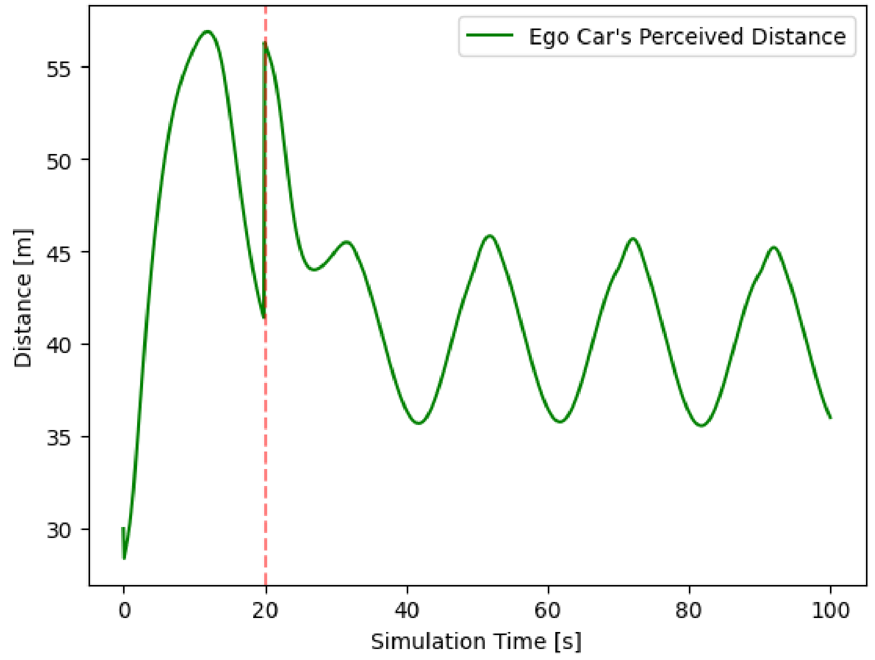
- Each attack can be «***constant***» or «***periodic***».
  If «***constant***» it starts at *start_time* and continues until the end of the simulation.
  If «***periodic***» starts at *start_time*, stops after a *period* (settable parameter), starts again after *period* and so on.

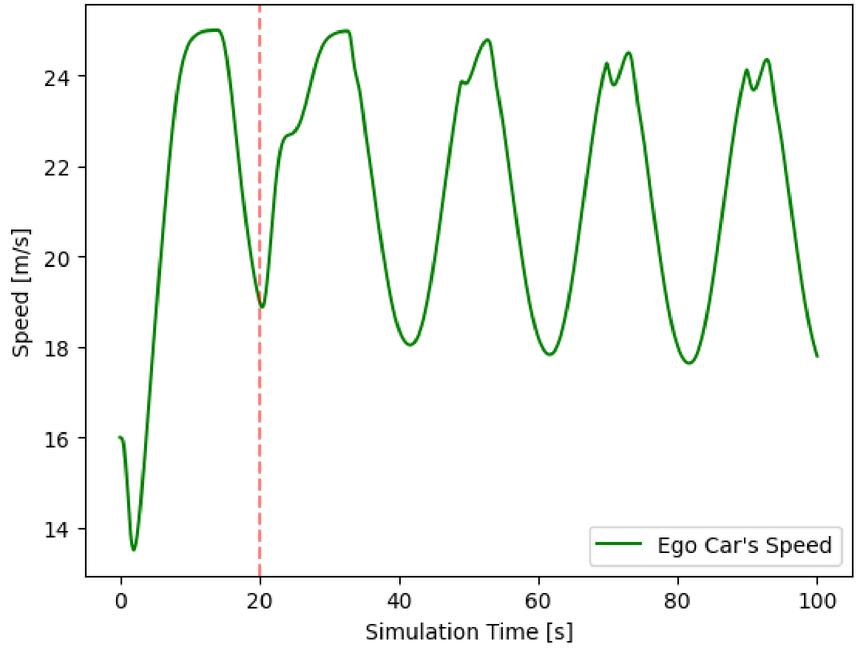
4. Case Study - Attack Example Simulation


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

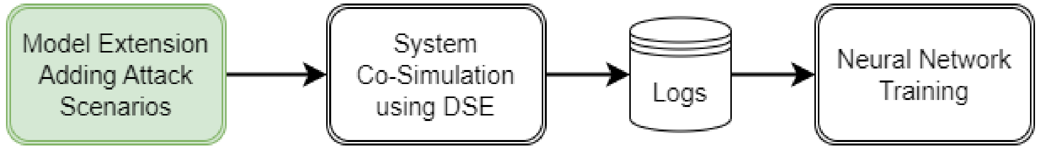
Attack:

Constant step function attack with height of 15 [m] on the measured distance, starting after 20 seconds.
The figure on the right shows the side-effect of the attack on Ego Car's speed.

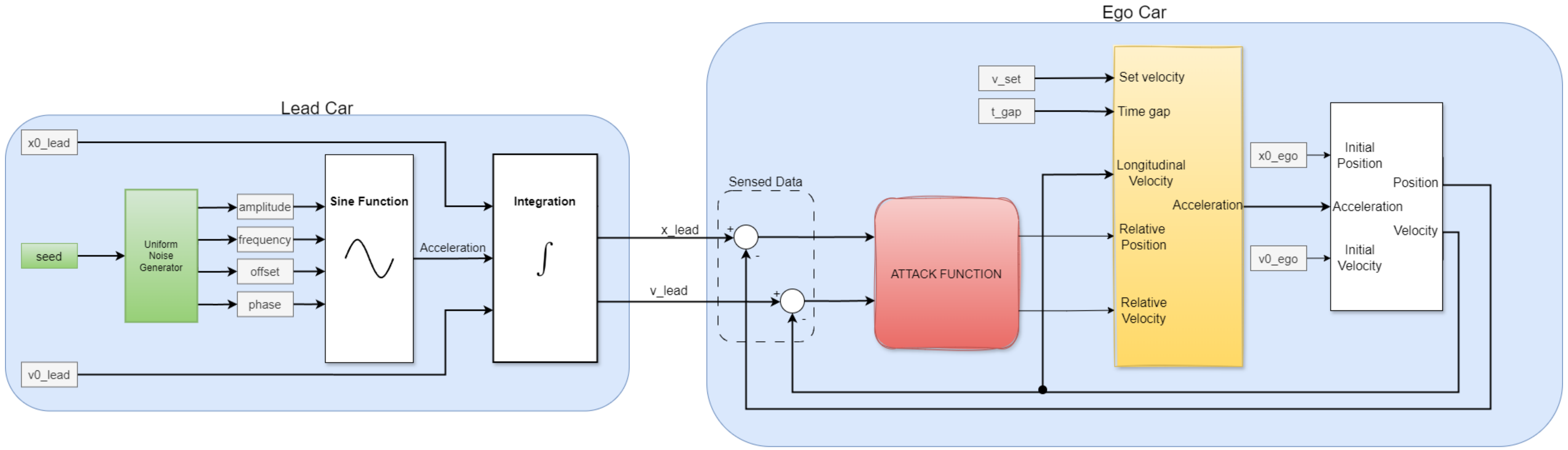




4. Case Study – Final Model

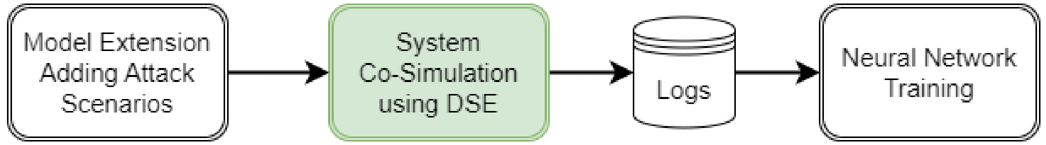

Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

Frequency, Phase, Offset and Amplitude are substituted by a Seed.



Finanziato dall'Unione europea NextGenerationEU | Ministero dell'Università e della Ricerca | Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA | UNIVERSITÀ DI PISA

4. Case Study – Co-Simulation and Data Gathering


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

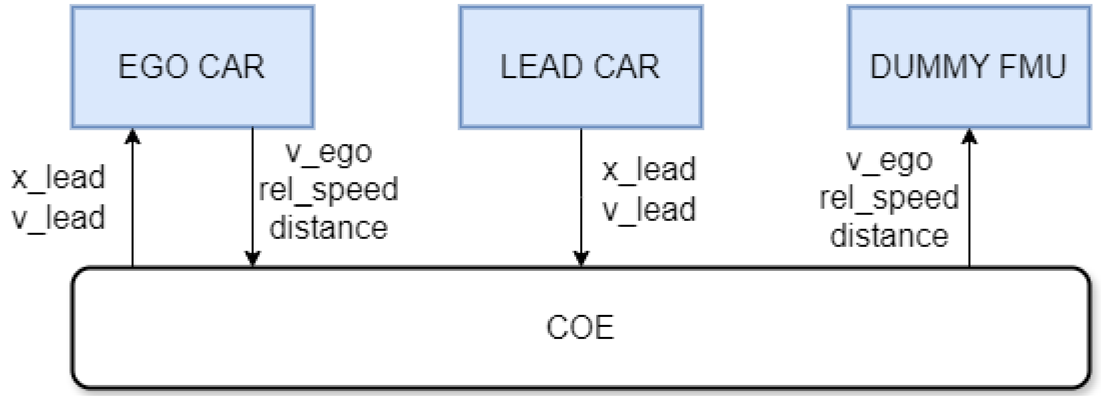
The data gathering process was divided in four smaller batches, one for each attack scenario and one for the nominal case.

For each attack scenario around 3000 simulations were ran.

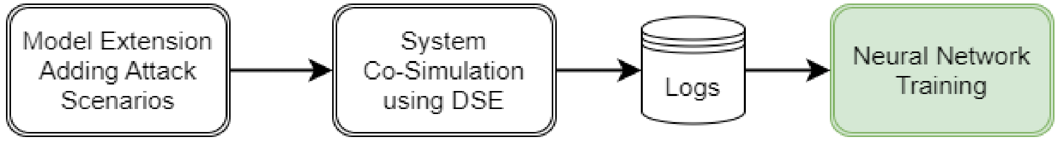
The parameters were set in order to have a balanced dataset between data with an attack and data without one.

The Dummy FMU allows to save internal state variables.

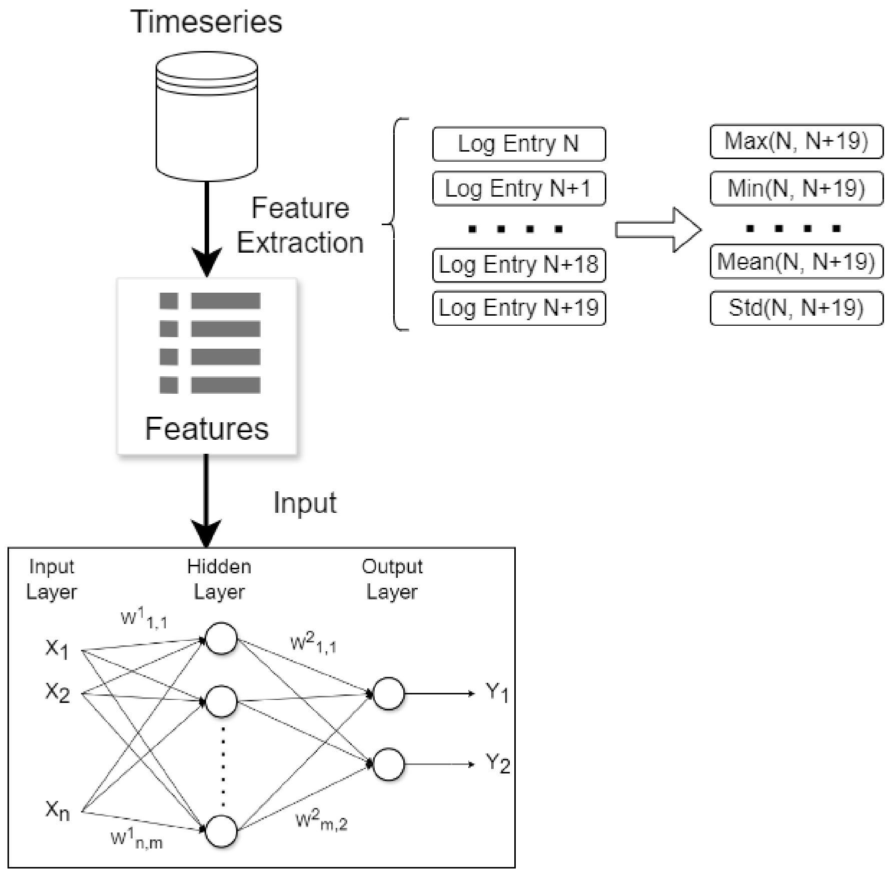
Some simulations may reach an inconsistent state after a while (in case of crashing vehicles, co-simulation doesn't stop), a script was developed to cycle through the csv files and remove these wrong samples (check the vehicles position).



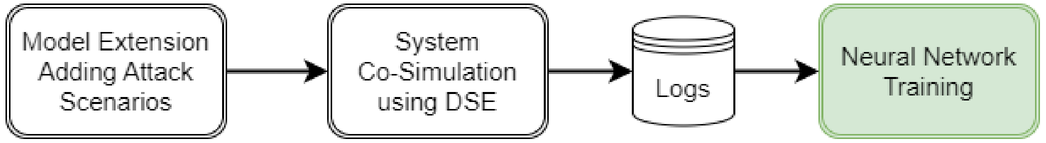
4. Case Study – Classifiers


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

- A compromise between safety and overhead has been found by setting the classification period at 2 seconds.
- The time step of the cosimulation was 0.1 seconds.
- The classifier requires a set of 6 features computed from a window of 20 entries (2 s / 0.1 s) related to 3 different measurements:
  ego_speed, relative_distance, relative_velocity
- The features computed are:
  max, min, mean, std, harmonic mean and skewness



4. Case Study – MLP Results


Model Extension Adding Attack Scenarios
System Co-Simulation using DSE
Logs
Neural Network Training

Best Training Algorithm:
  Bayesian regularization backpropagation (trainbr)

Best Network Structure:
  18 input neurons
  1 hidden layer of 25 hidden neurons

Training Confusion Matrix

269.902 51%	30.020 5,6%	Predicted Output: No Attack
1.978 0,4%	227.565 43%	Predicted Output: Under Attack
Label: No Attack	Label: Under Attack	Overall Accuracy: 94%

Testing Confusion Matrix

47.316 50,6%	5.303 5,7%	Predicted Output: No Attack
366 0,4%	40.450 43,3%	Predicted Output: Under Attack
Label: No Attack	Label: Under Attack	Overall Accuracy: 93,9%

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
UNIVERSITÀ DI PISA

5. Conclusions and Further Work

- To further assess the validity of the results, another testing phase was made using data taken from new scenarios, never seen before by the network, obtaining the same accuracy
- The value for the false negatives is rather high and generally this is not ideal. In our case, most of the false negatives correspond to system states reached after a previously detected attack scenario. Also, the MPC of the Ego Car mitigates the effects of attacks.
- The classifier could be exported as an FMU, allowing the detection of attacks in co-simulation, thus making a step forward for the creation of a security related digital twin.
- The dataset could be enhanced using real data and the simulations could be tuned accordingly, allowing a more significative and trusty training dataset.
- The application of this methodology to detect attacks in a platooning application in an ongoing work (PRIN PNRR 2022 FORESEEN project www.forseen.dii.unipi.it).