Automatic calibration of traffic simulations with artificial neural networks

Candidate: Rodrigo França Daguano

Supervisor: Prof. Dr. Leopoldo Rideki Yoshioka, PhD

Polytechnic School of the University of São Paulo

April 2019

Contents

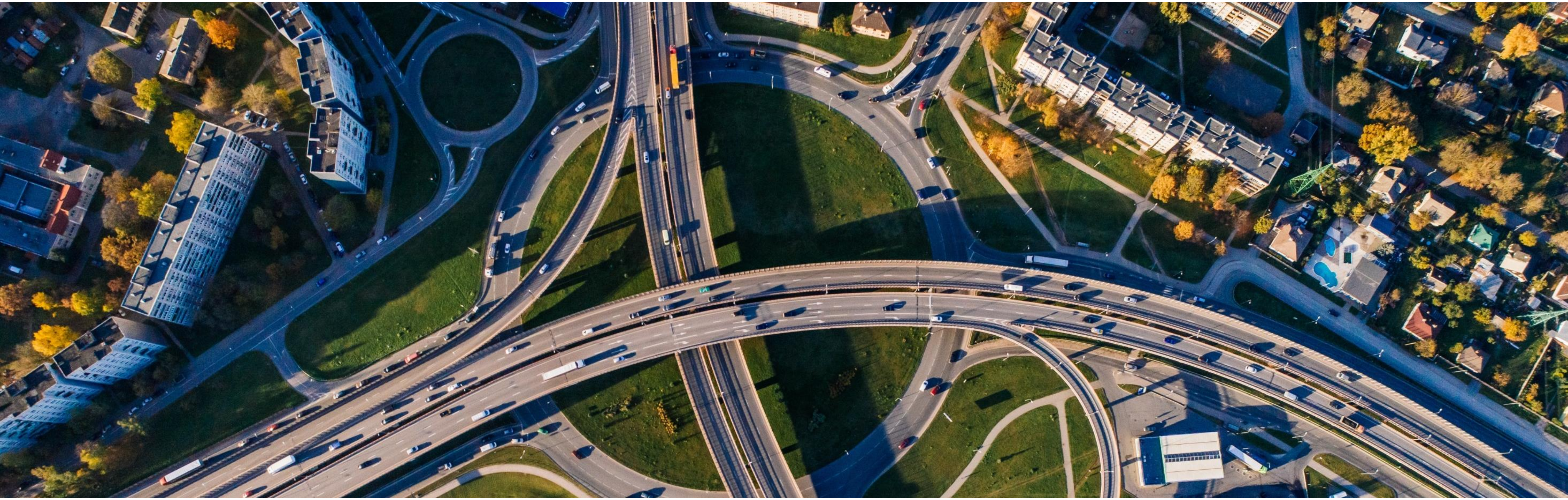
- Introduction and motivation
- Research proposal
- Methodology and experiments
- Partial results and insights

Introduction

Introduction

Smart Cities (MOHANTY; CHOPPALI; KOUGIANOS, 2016)

Introduction

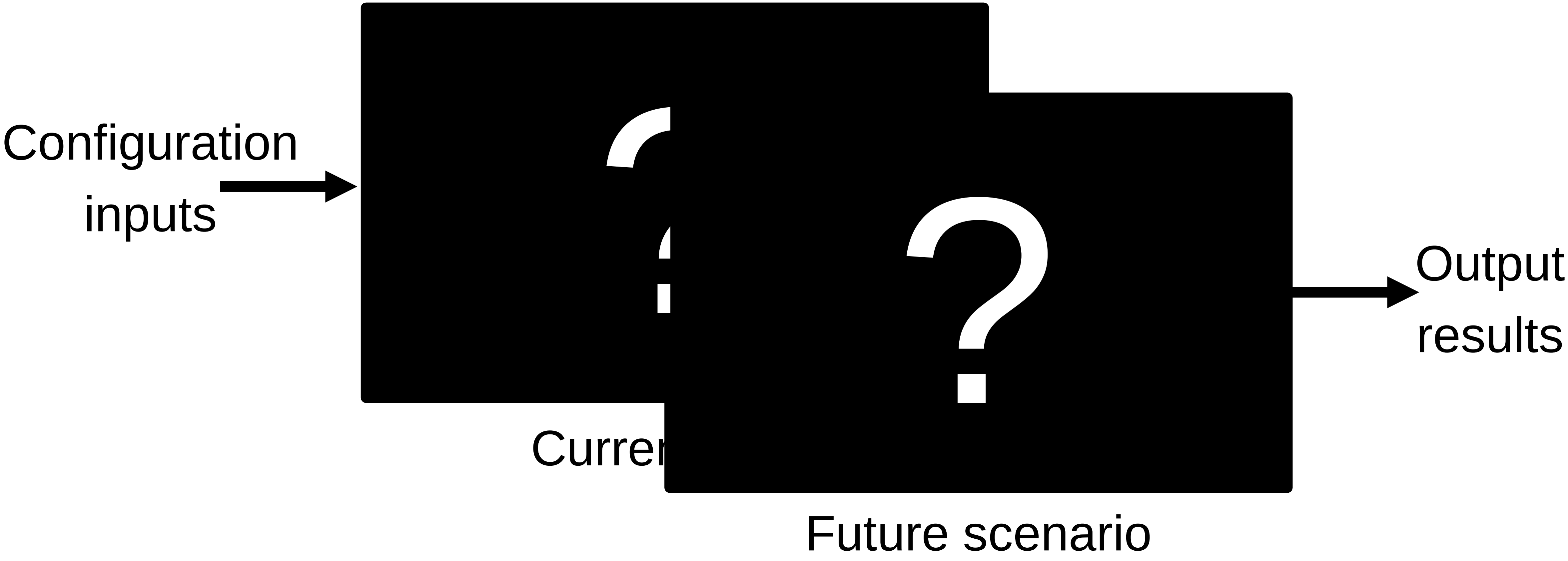


Smart Transportation – Planning and operation (XIONG et al., 2012)

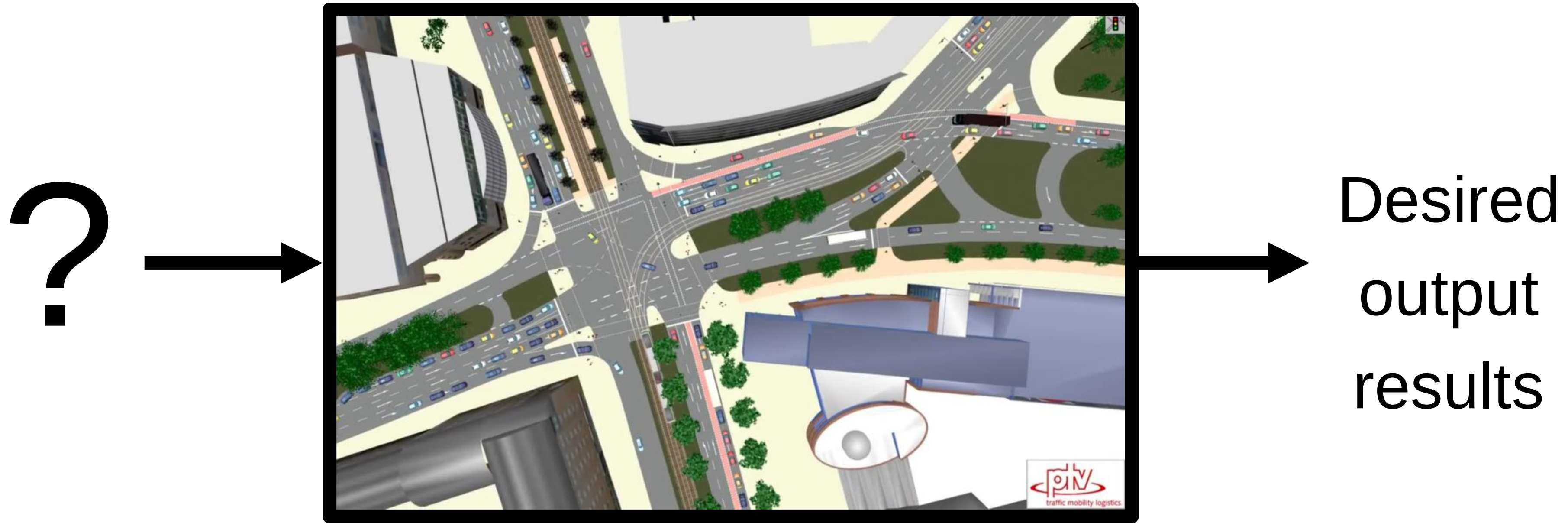
Introduction

Traffic simulations (CHU et al., 2003; HOLLANDER; LIU, 2008)

New infrastructure proposal

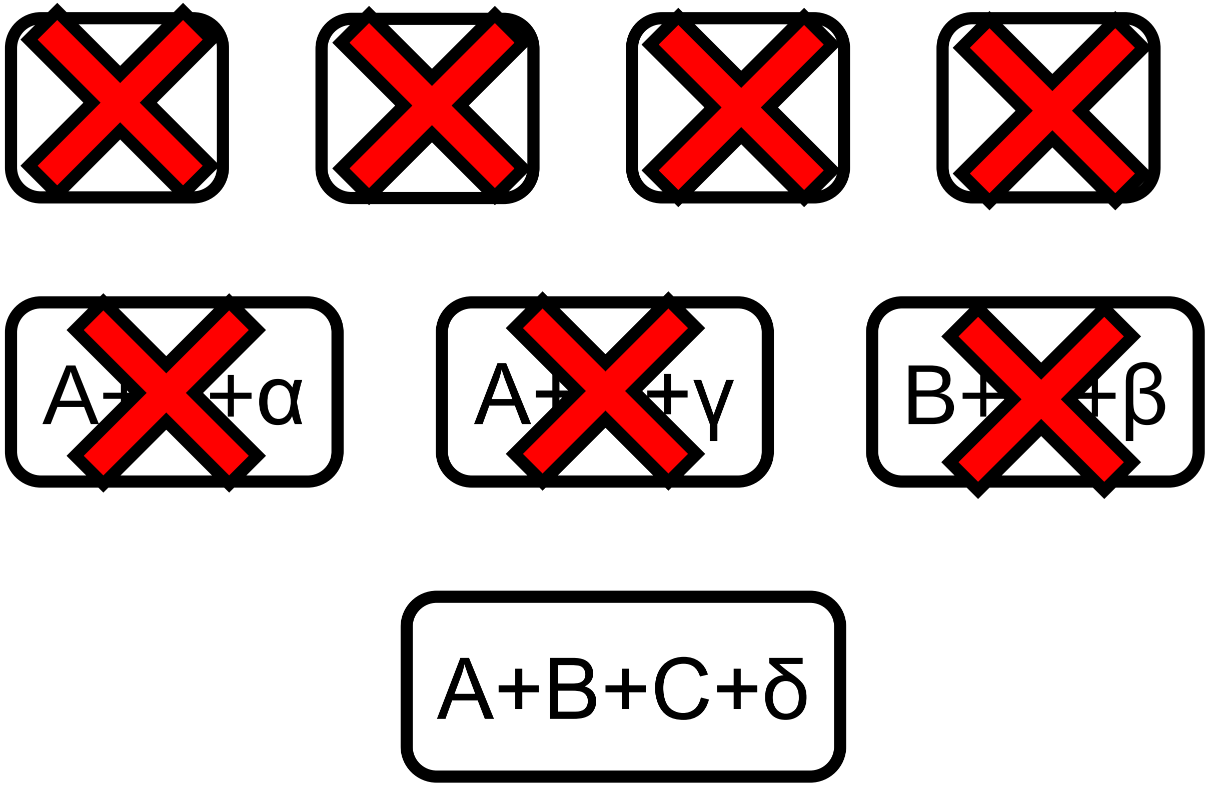


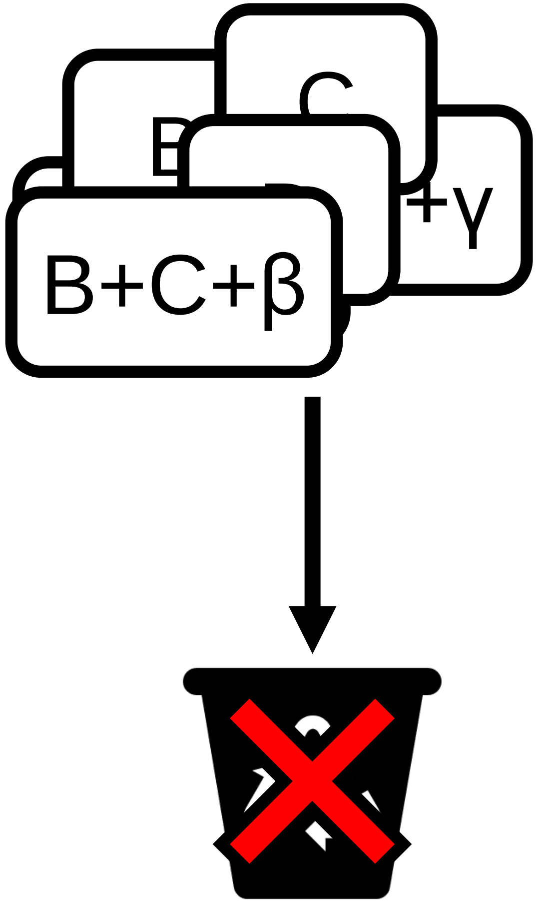
Calibration



- Adequate inputs to minimize the error
- Trend in optimization is the Genetic Algorithm (RRECAJ; BOMBOL, 2015)

Genetic algorithm example





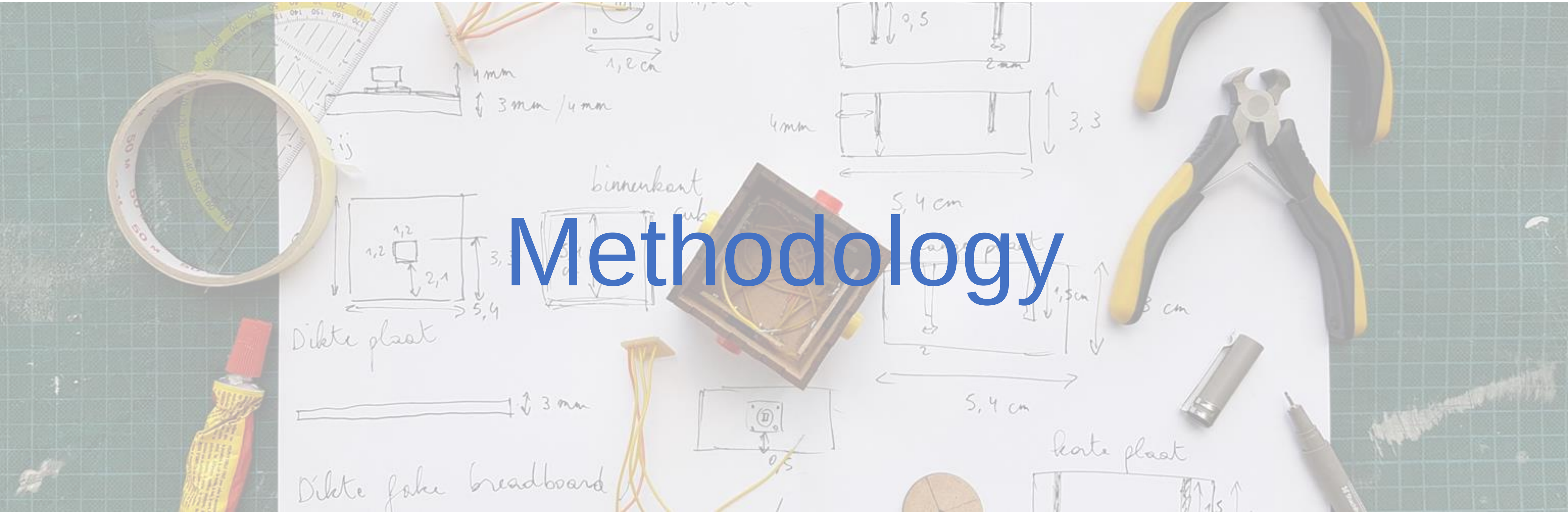
- New simulation, new cycle

Research question

- How to create automatic calibration models that are reusable across similar traffic simulations?

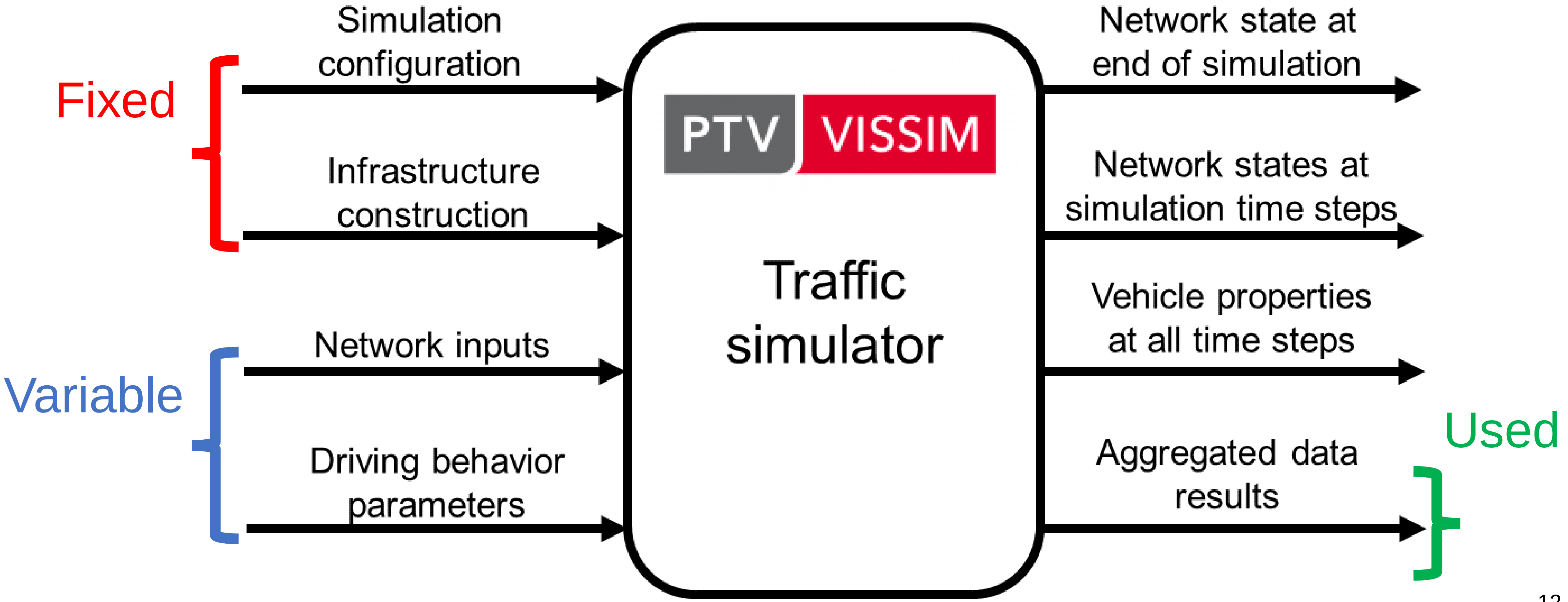
Solution proposal:

- Methodology to create machine learning tools that can automatically calibrate traffic microsimulations.

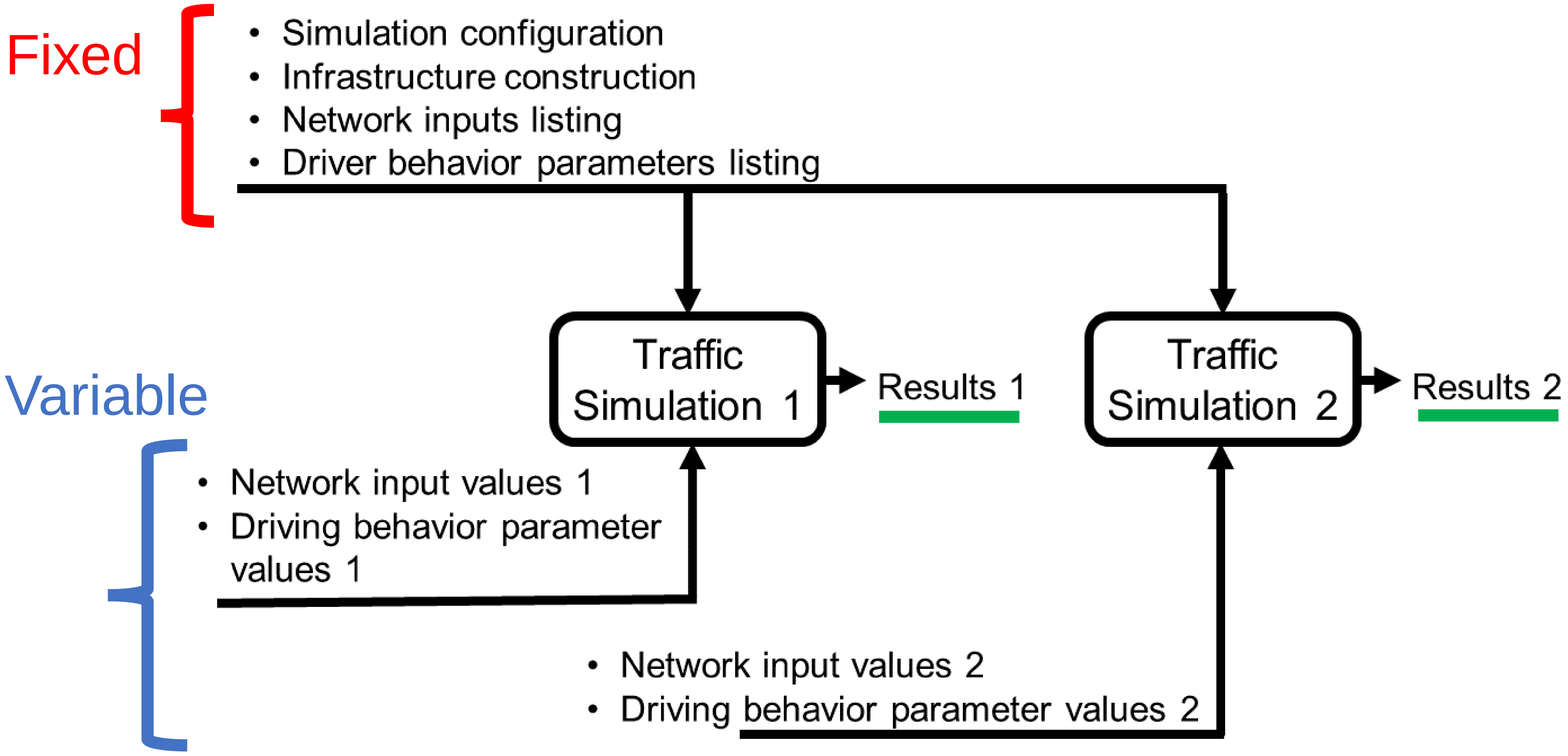


Methodology

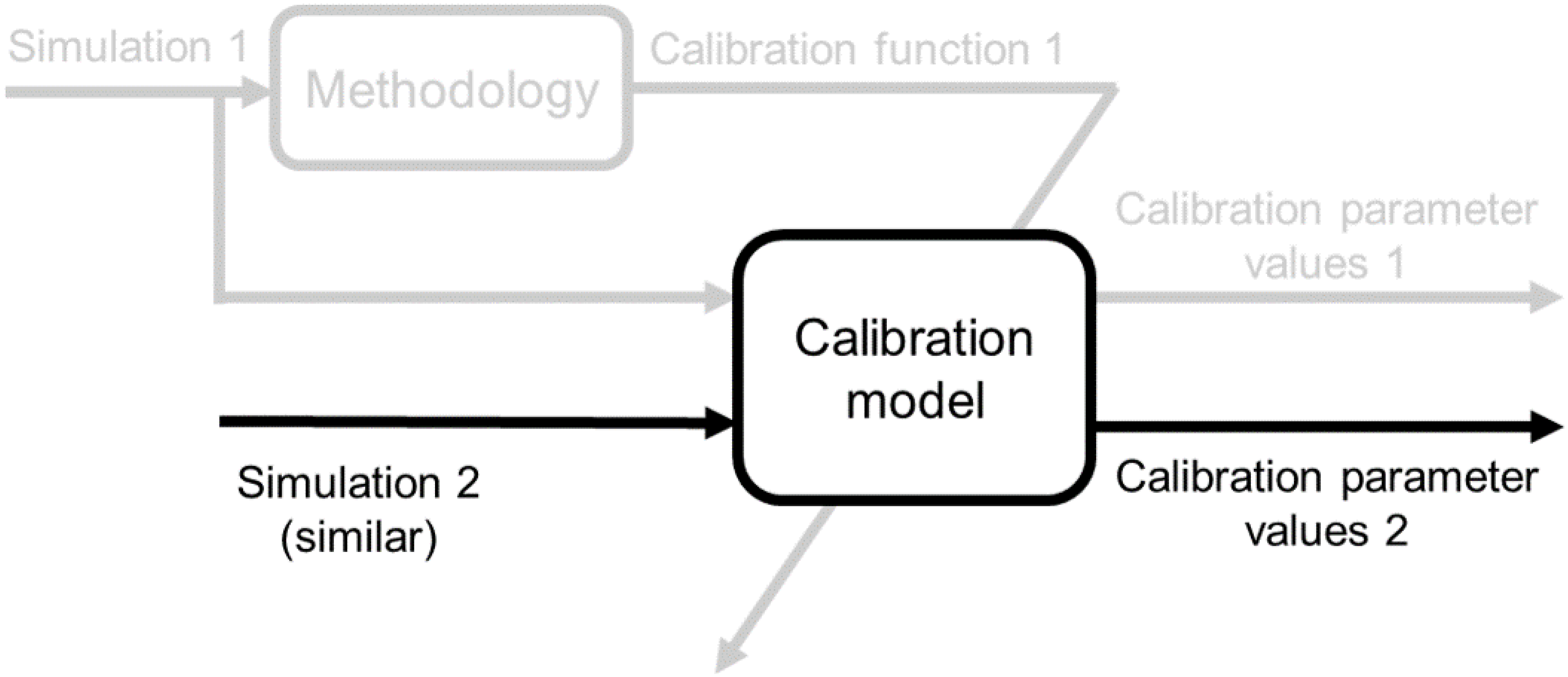
PTV Vissim as a function



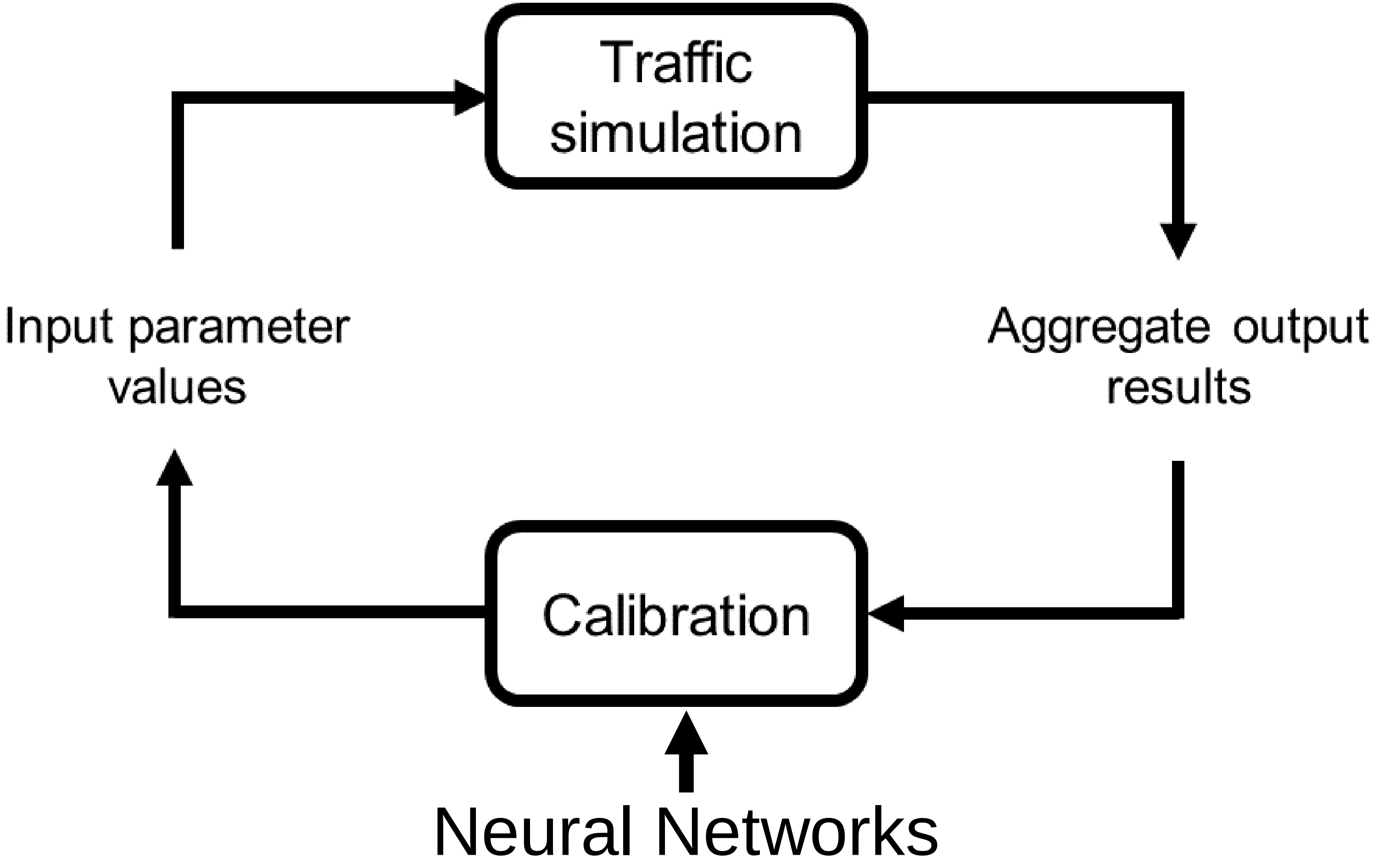
Similar simulations



Calibration model reuse



Simulation and calibration functions



Dataset variables

Example of dataset format

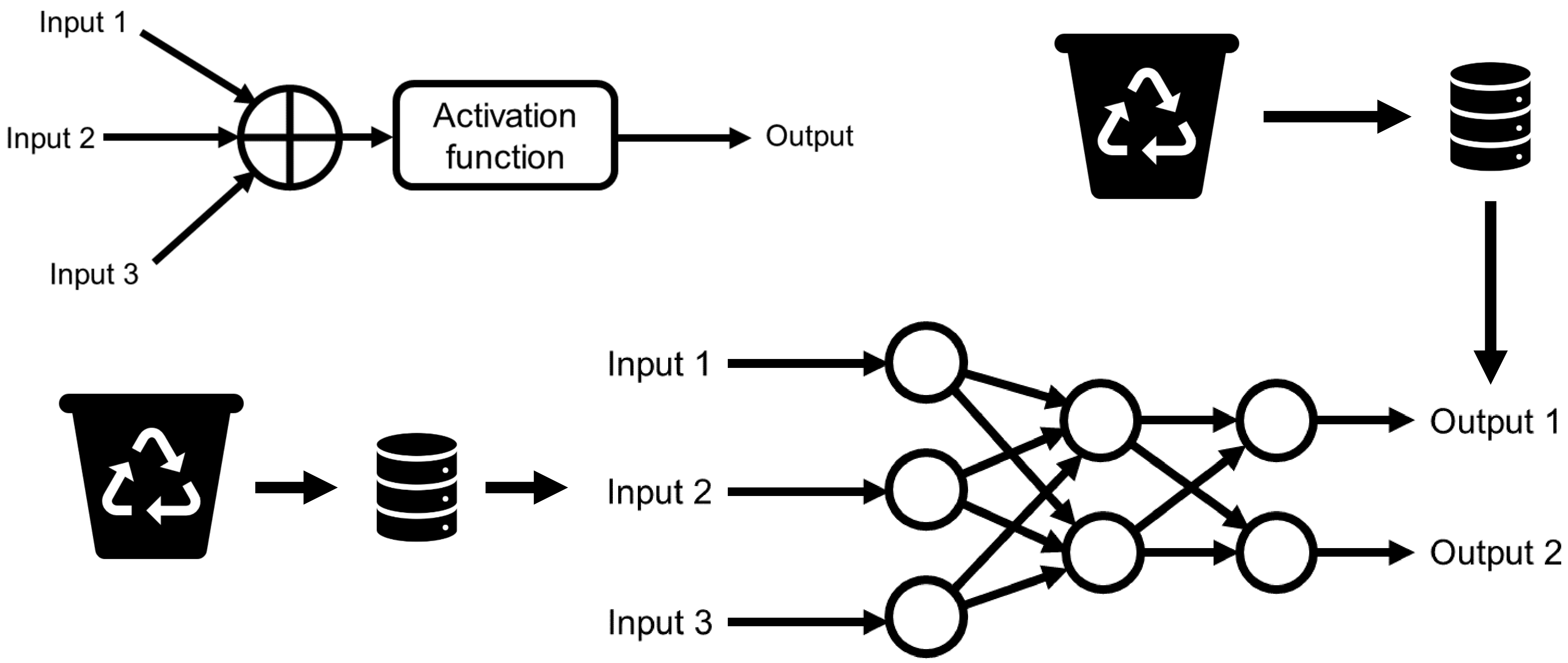
Vehicle volume 1 (1/h)	**Vehicle volume 2 (1/h)**	**Route Decision 1-3**	**Travel Time 1 (s)**	**Average speed 1 (km/h)**	**Vehicle count 1**
600	120	0.4	45	26.55	357
550	2300	0.5	140	5.88	12
1200	100	0.1	30	34.5	689

Simulation

Parameters $\rightleftarrows$ Aggregate results

Calibration

Artificial Neural Networks



Methodology

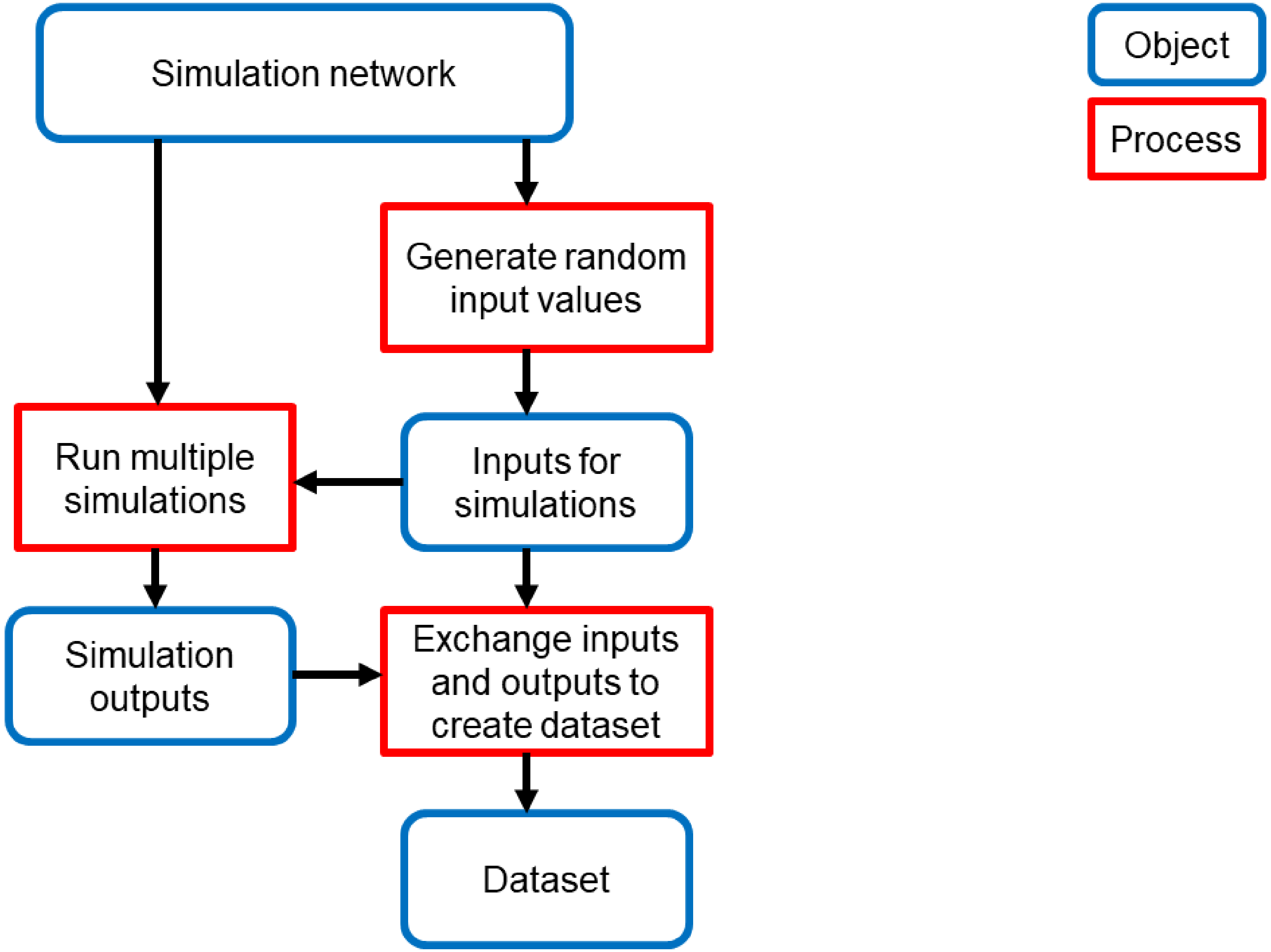
Start with: Simulation network.

1. **Dataset creation**
    1. Generate random values for inputs to be calibrated.
    2. Run multiple simulations.
    3. Exchange simulation inputs and outputs to create dataset.

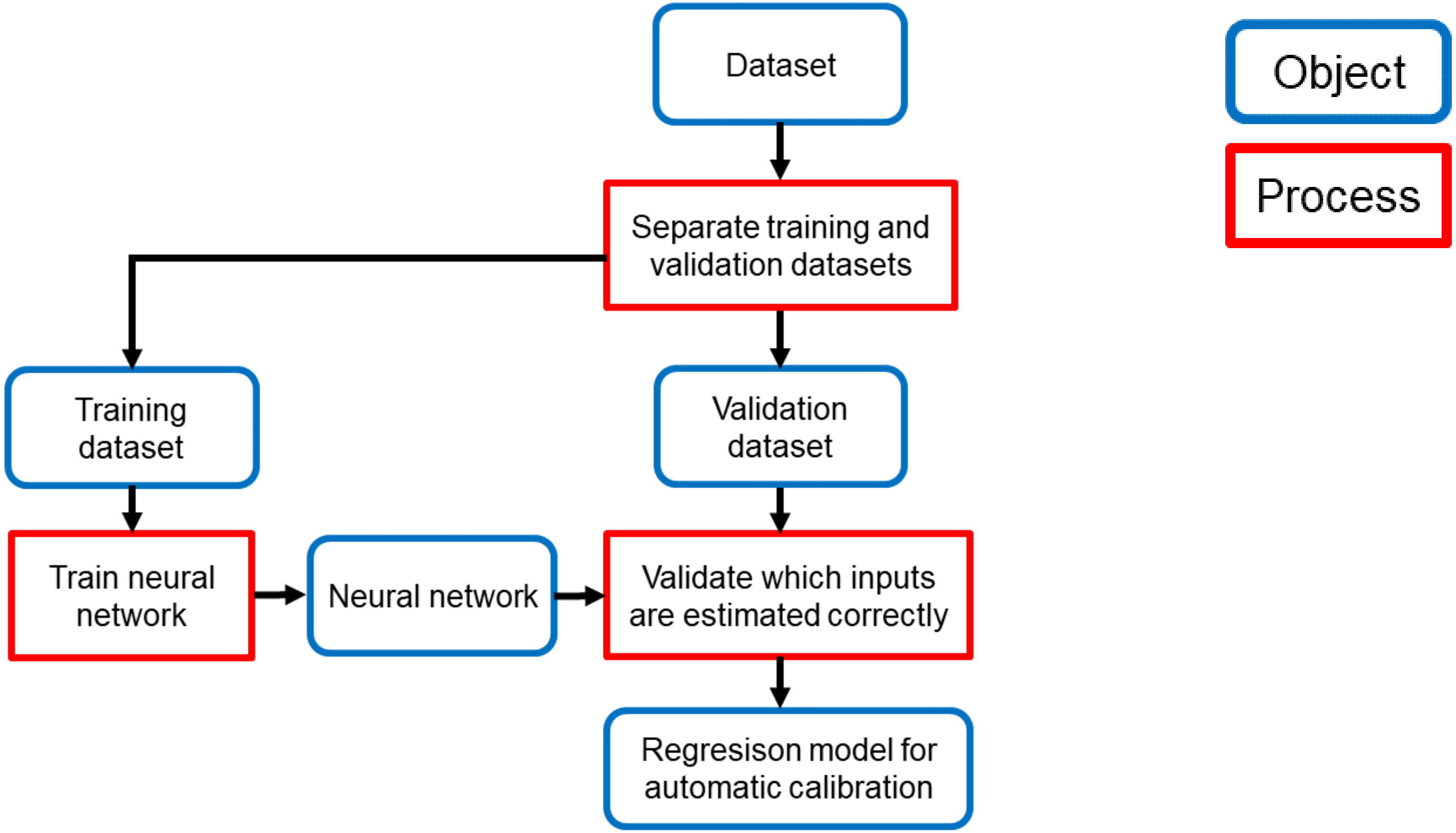
2. **Neural network training**
    1. Separate training and validation datasets.
    2. Train neural network as a regression model.
    3. Validate which inputs are estimated correctly.

End with: Regression model for automatic calibration

Flowchart – Dataset creation



Flowchart – Neural network training



Experiments

Simulator

Inputs

- Vehicle volumes
- Route decisions
- Behavioral parameters
  - W74, W99 car-following
  - Headway for lane change

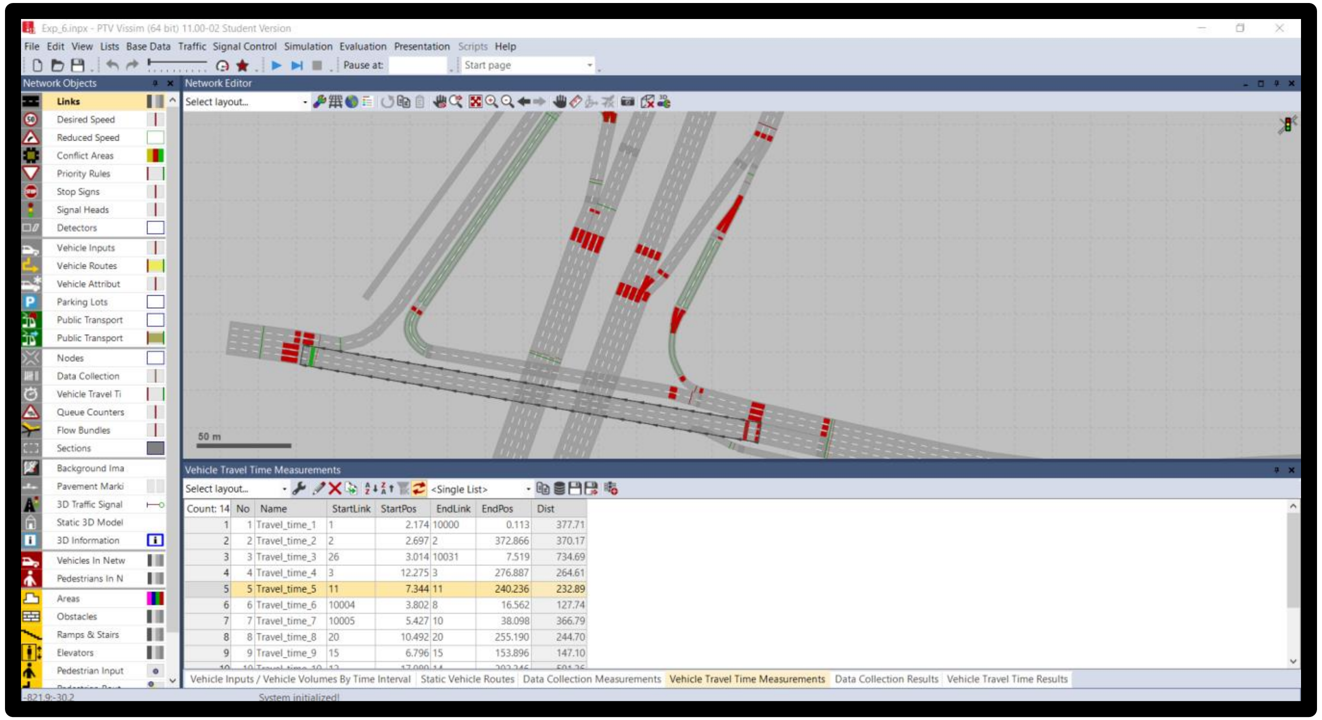
Outputs (aggregate results)

- Vehicle counts
- Travel times
- Harmonic average speeds

COM interface

~4000 simulations

PTV VISSIM



Simulations run by Python scripts I

```
…
all_flows = Vissim.Net.VehicleInputs.GetAll()
all_routes = Vissim.Net.VehicleRoutingDecisionsStatic.GetAll()
for k in range(4000):
        this_line = myfile.next()
        index = 0
        for t in range(len(all_flows)):
                all_flows[t].SetAttValue("Volume(1)", this_line[index])
                index+=1
        for m in range(len(all_routes)):
                options = all_routes[m].VehRoutSta
                for n in range(len(options)):
                        options[n].SetAttValue("RelFlow(1)", this_line[index])
                        index+=1
        Vissim.Simulation.RunContinuous()
```

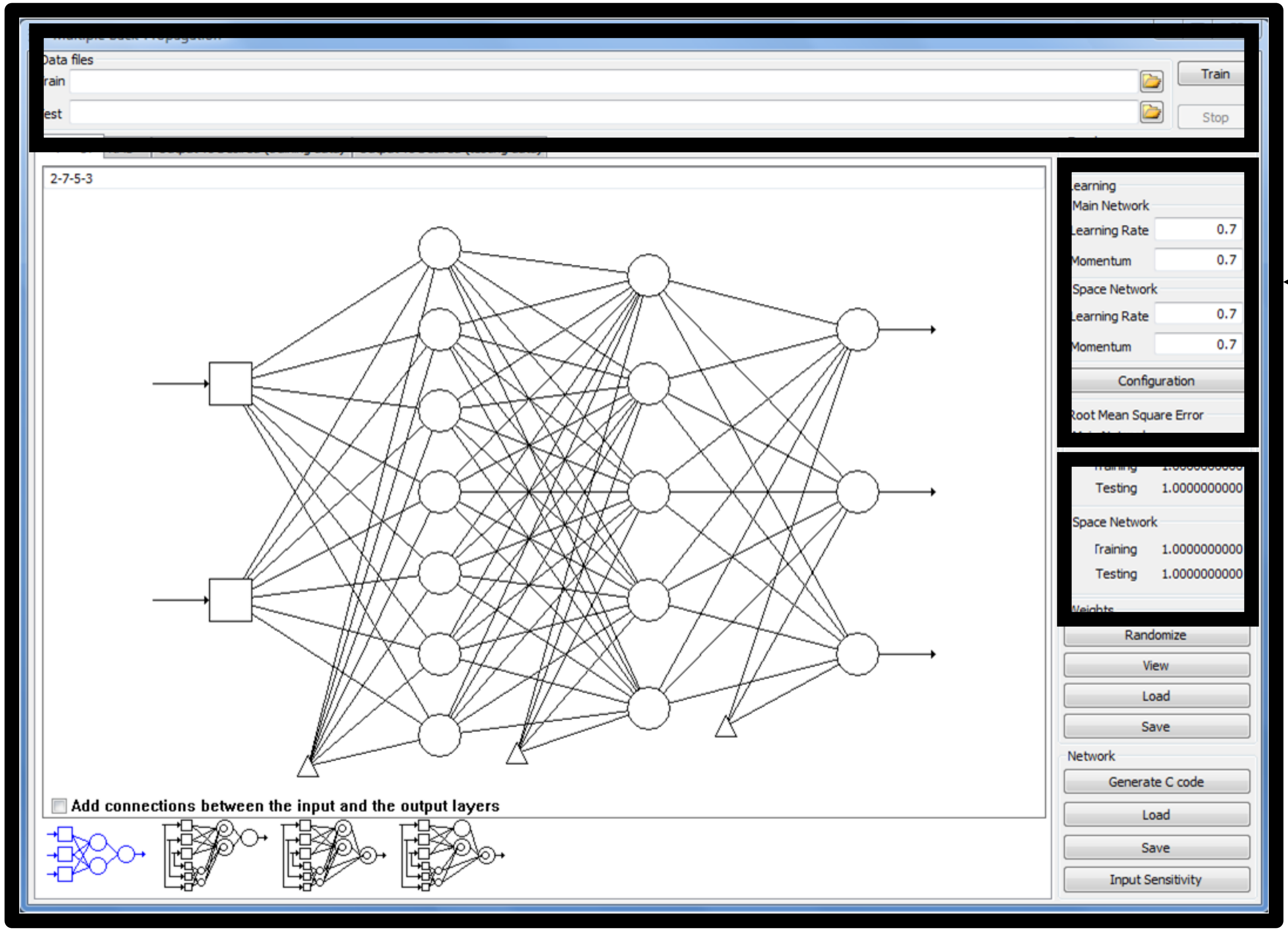
Simulations run by Python scripts II

```
…
for k in range(3000):
        this_line = myfile.next()
        driving_behavior_list[0].SetAttValue("W74ax", this_line[0])
        driving_behavior_list[0].SetAttValue("W74bxAdd", this_line[1])
        driving_behavior_list[0].SetAttValue("W74bxMult", this_line[2])
        driving_behavior_list[0].SetAttValue("MinHdwy", this_line[3])
        Vissim.Simulation.RunContinuous()
```

***Similar command for W99cc parameters**

Multiple Backpropagation software

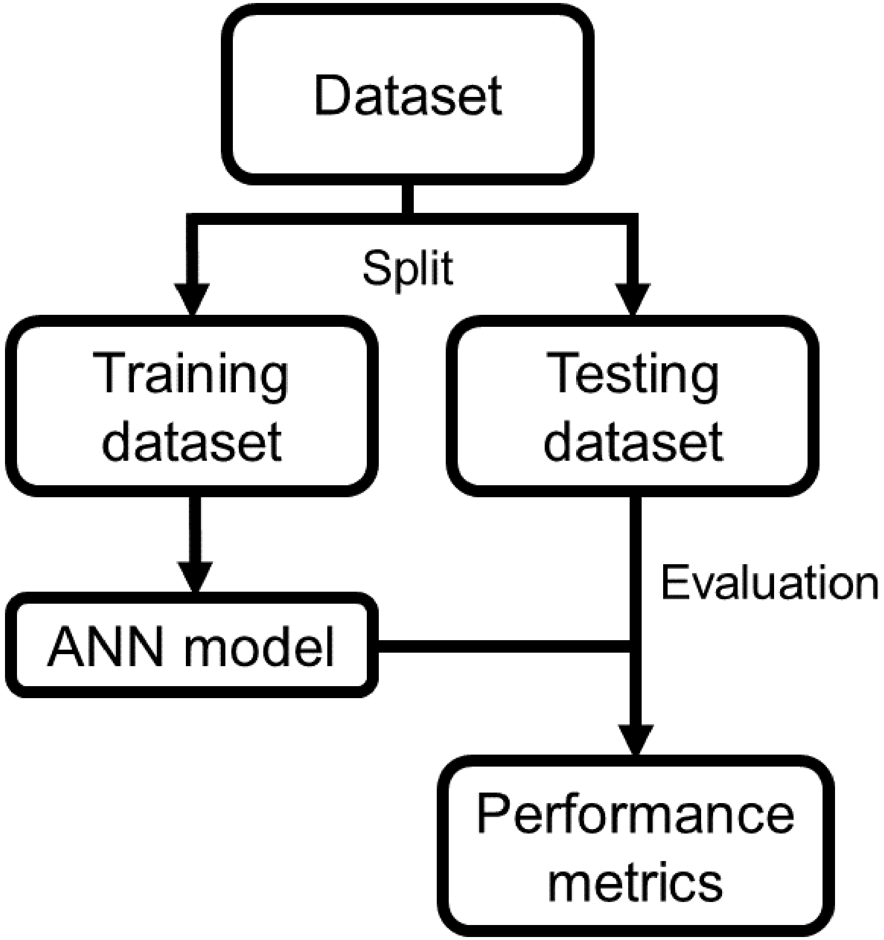
Load data

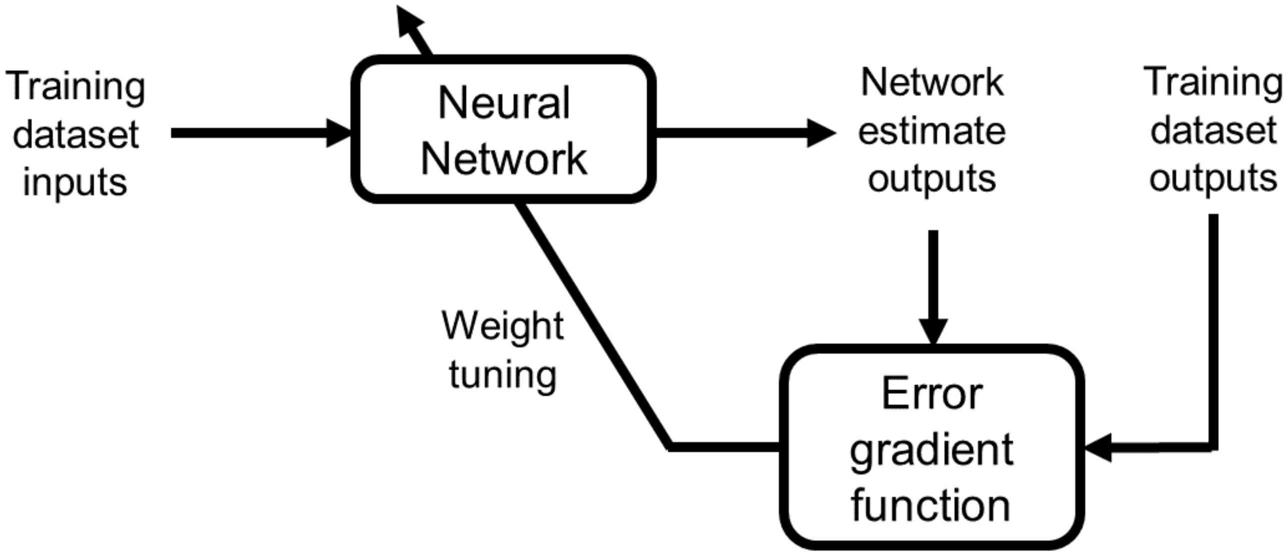


Configuration

Results

Training the Neural Network





Performance evaluation I

- Neural network minimizes RMS error of training dataset.

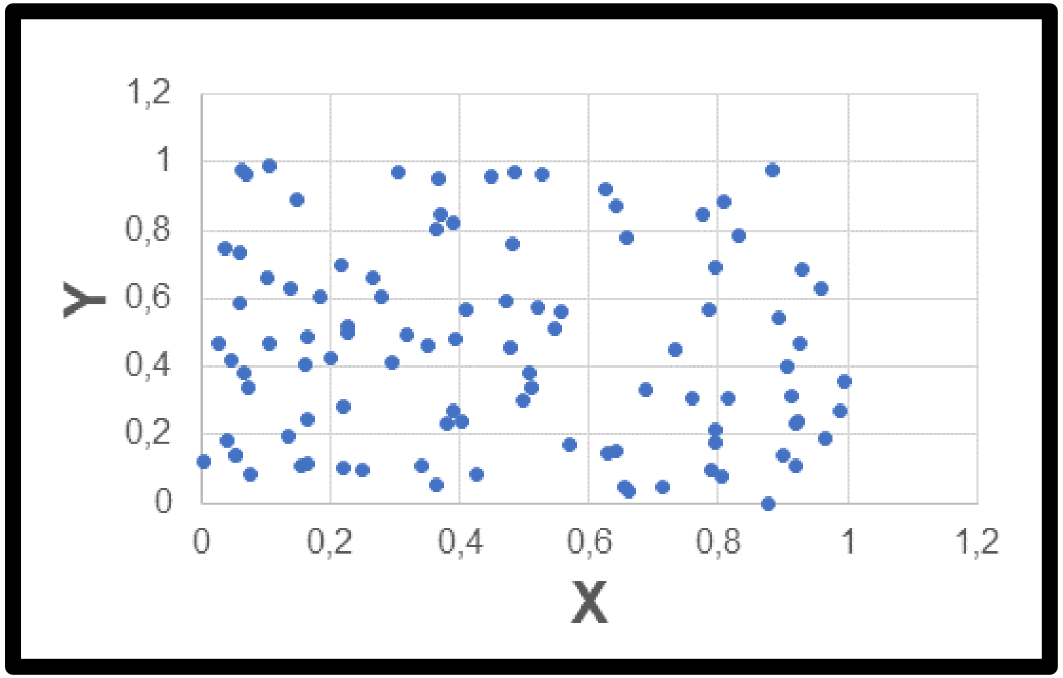
$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2}$$

- Strong correlation is desired
  - Between desired and estimated values of the validation dataset.
  - Indicates successful calibration capabilities.

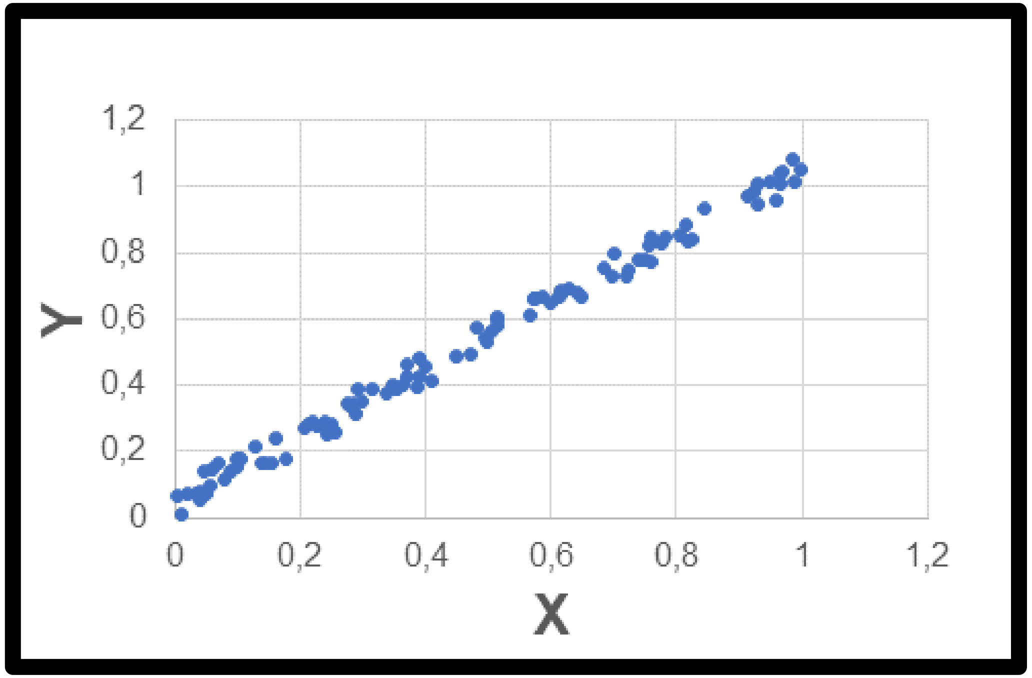
$$r_{xy} = \frac{\sum(x_i - \bar{x})(y - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 (y_i - \bar{y})^2}}$$

$$-1 \le r_{xy} \le 1$$

Performance evaluation II

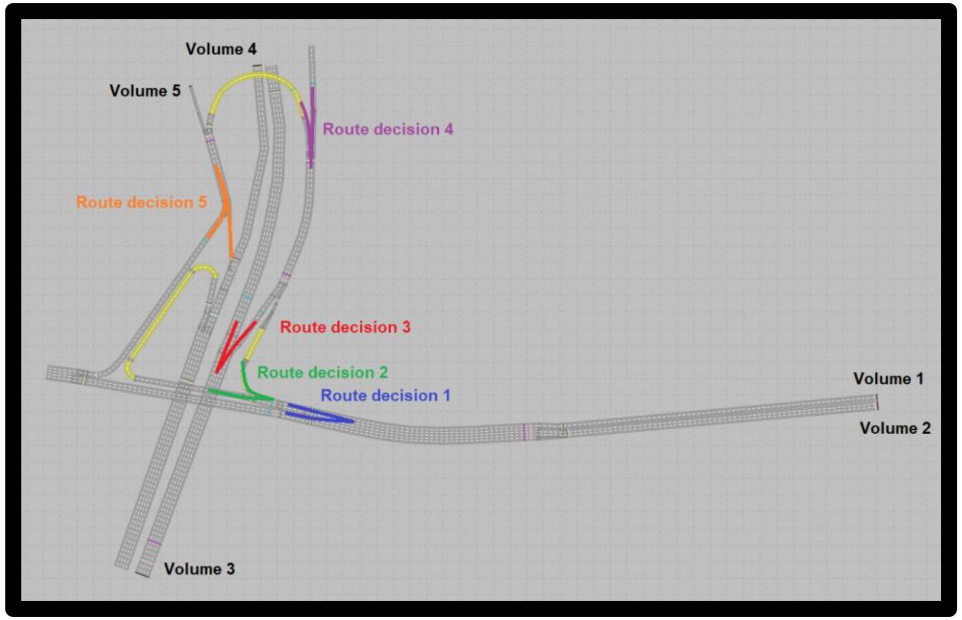


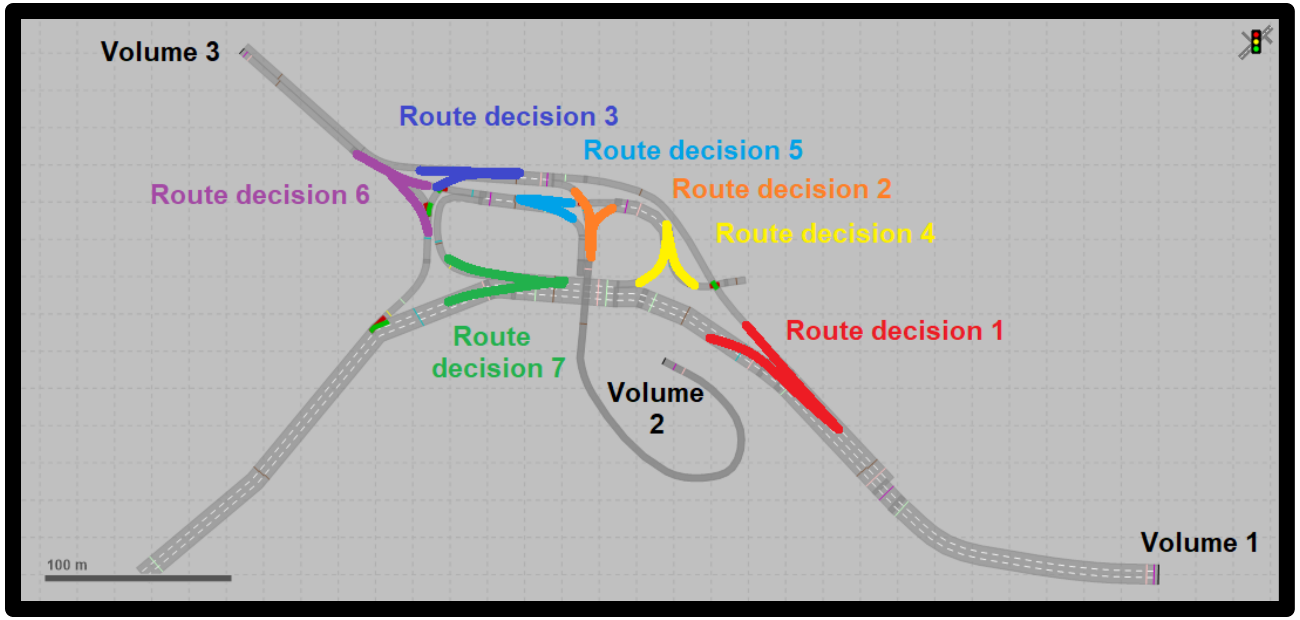
Weak correlation
close to 0



Strong correlation
close to 1

Experiments





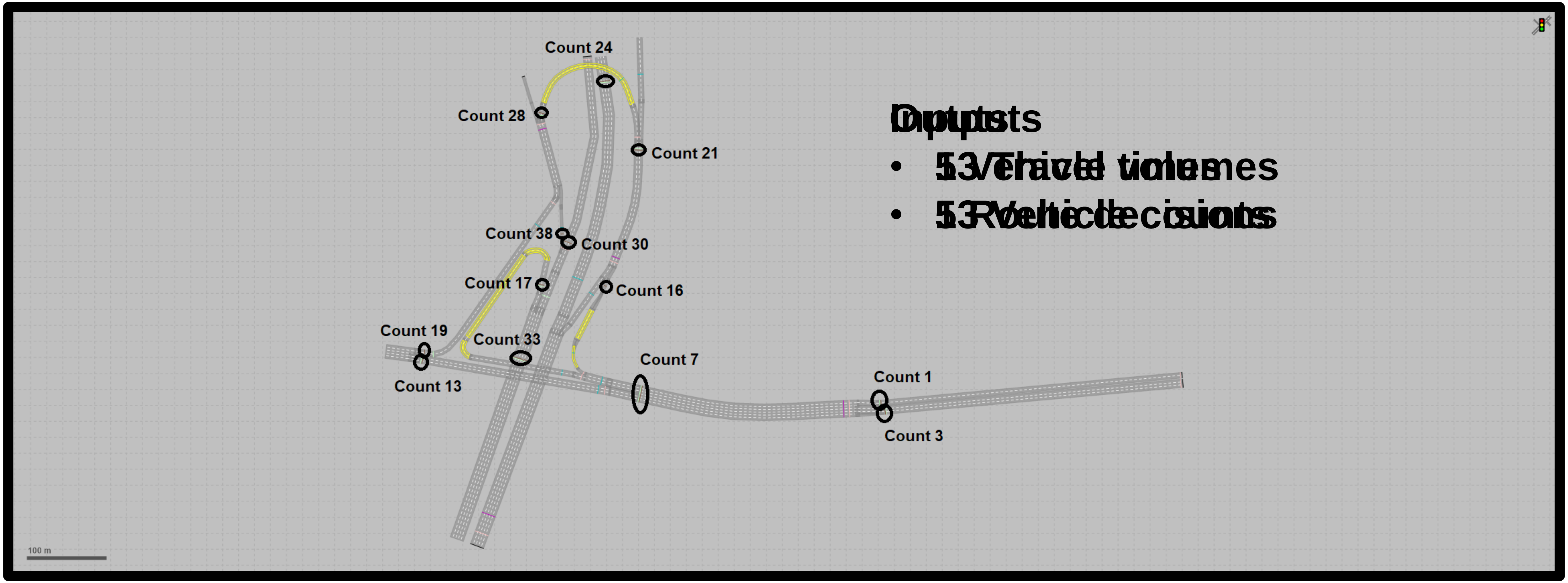
Experiment 8

- 23 de Maio/Radial Leste
  - Vehicle volumes and route decisions

Experiments 5, 6 and 7

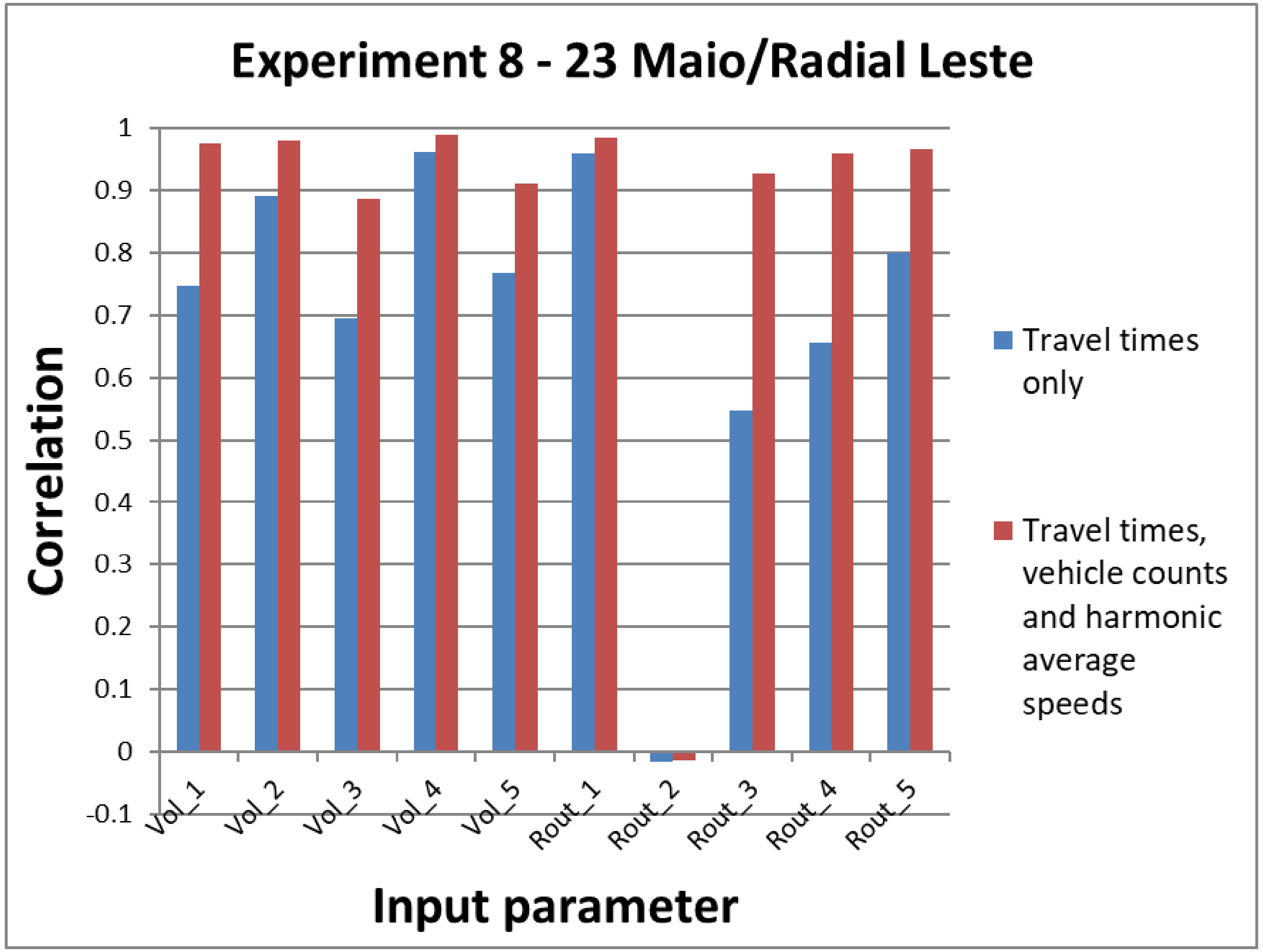
- Raposo Tavares km 23
  - Vehicle volumes and route decisions
  - W74 car-following and lane change
  - W99 car-following

Setup for experiment 8



Experiment 8 results I

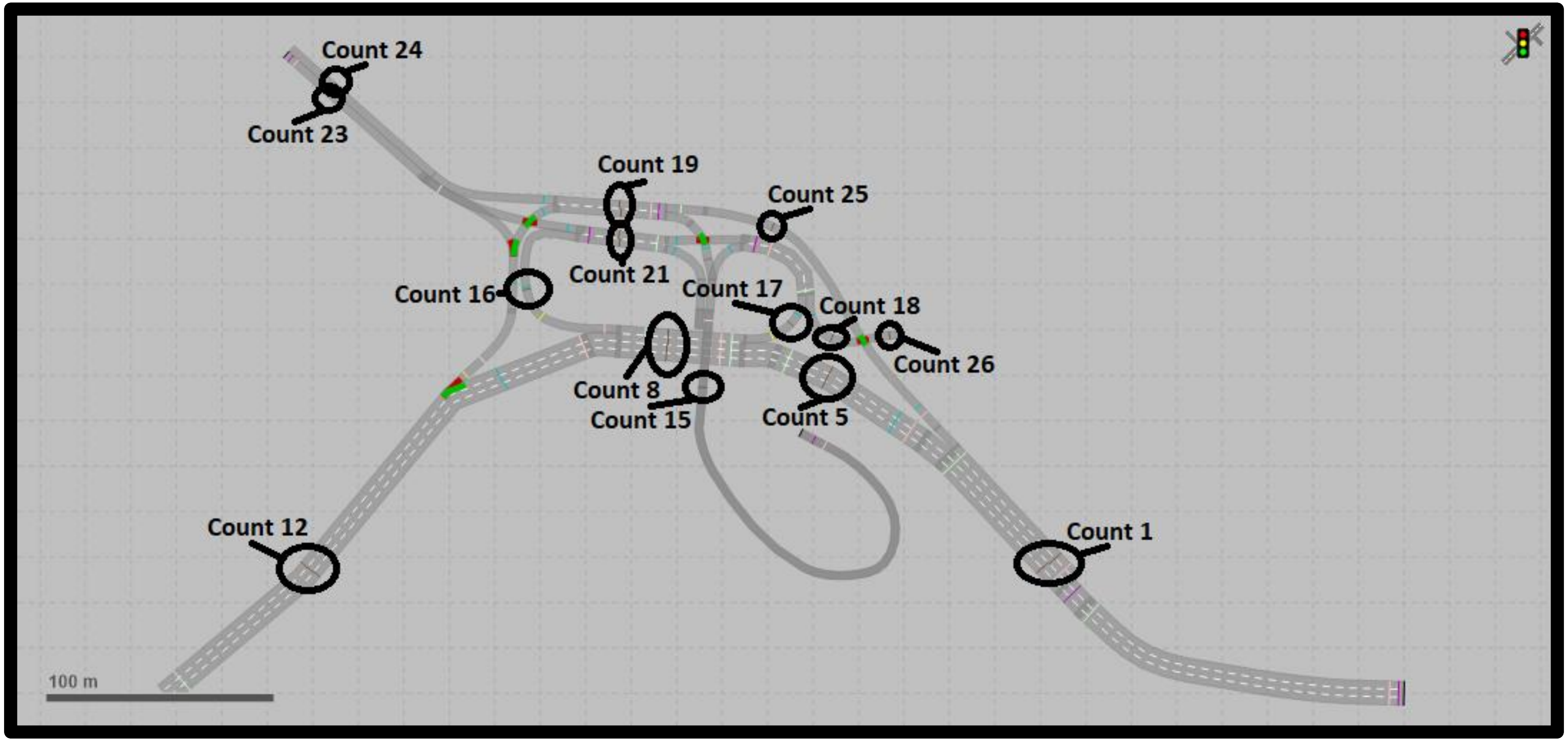
- 4000 simulations
- Duration of 60 minutes
- Excludes 10 initial minutes
- Neural network with 2 hidden layers of 50 neurons (empirical)



Experiment 8 results II

Outputs used	Avg correlation	Worst correlation	Avg correlation w/o worst	RMS test error
• Travel times	0.7012671	-0.01641	0.781009	9.60%
• Travel times • Vehicle counts • Harmonic average speeds	0.8569168	-0.01379	0.953662	6.30%

Setup for experiments 5, 6 and 7 - I



Setup for experiments 5, 6 and 7 - II

- 4000 simulations
- Duration of 30 minutes
- Excludes 5 initial minutes
- Default neural network with 2 hidden layers of 50 neurons (empirical)

Inputs:

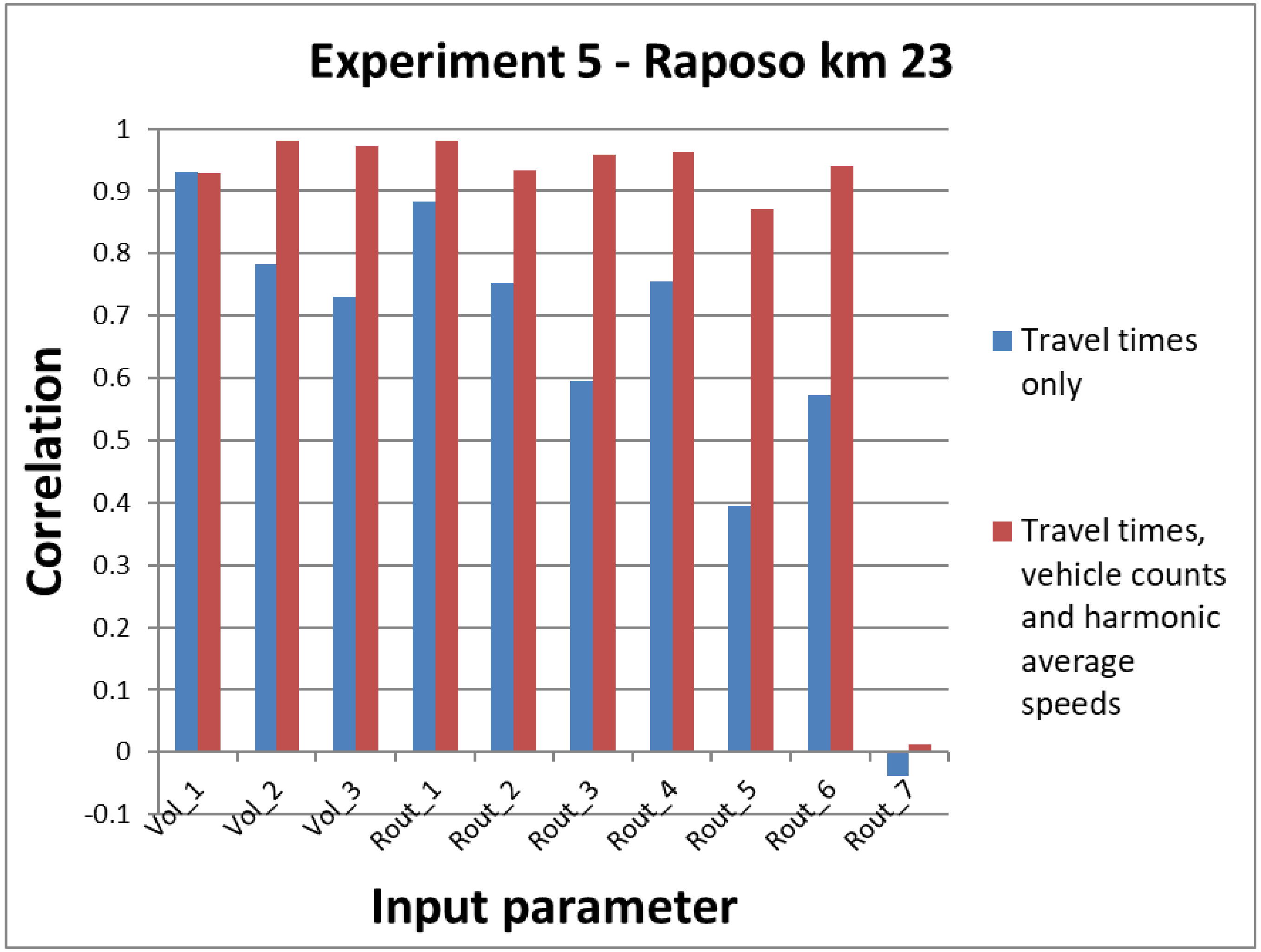
Experiment 5
- 3 Vehicle volumes
- 7 Route decisions

Experiment 6
- 3 W74 parameters
- Minimal headway for lane change

Experiment 7
- 9 W99 parameters

Experiment 5 results I

- 3 Vehicle volumes
- 7 Route decisions

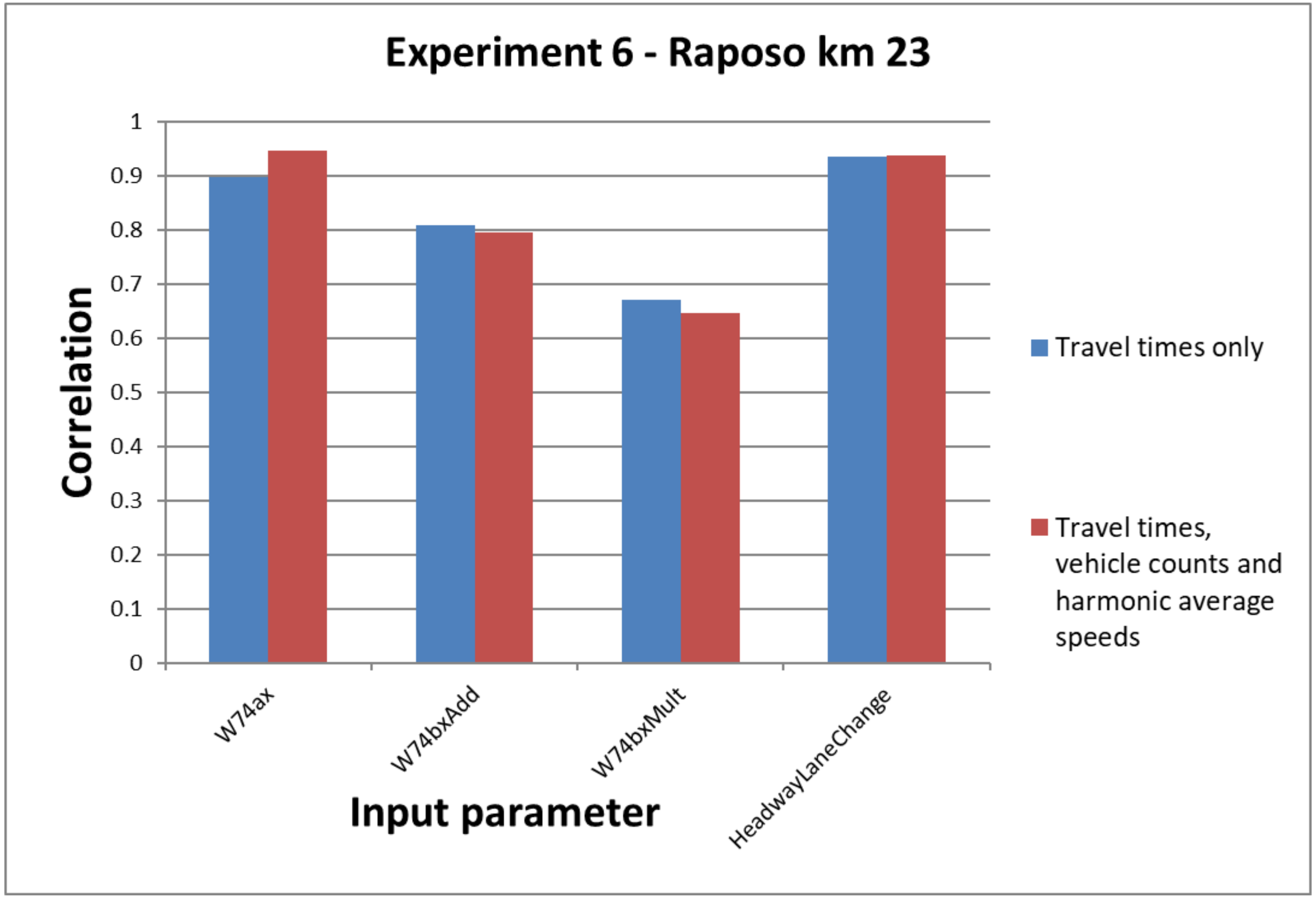


Experiment 5 results II

Outputs used	Avg correlation	Worst correlation	Avg correlation w/o worst	RMS test error
• Travel times	0.636267	-0.03746	0.711125556	10.50%
• Travel times • Vehicle counts • Harmonic average speeds	0.8539175	0.012659	0.947390667	5.50%

Experiment 6 results I

- W74ax
- W74bxAdd
- W74bxMult
- Minimal headway for lane change



Experiment 6 results II

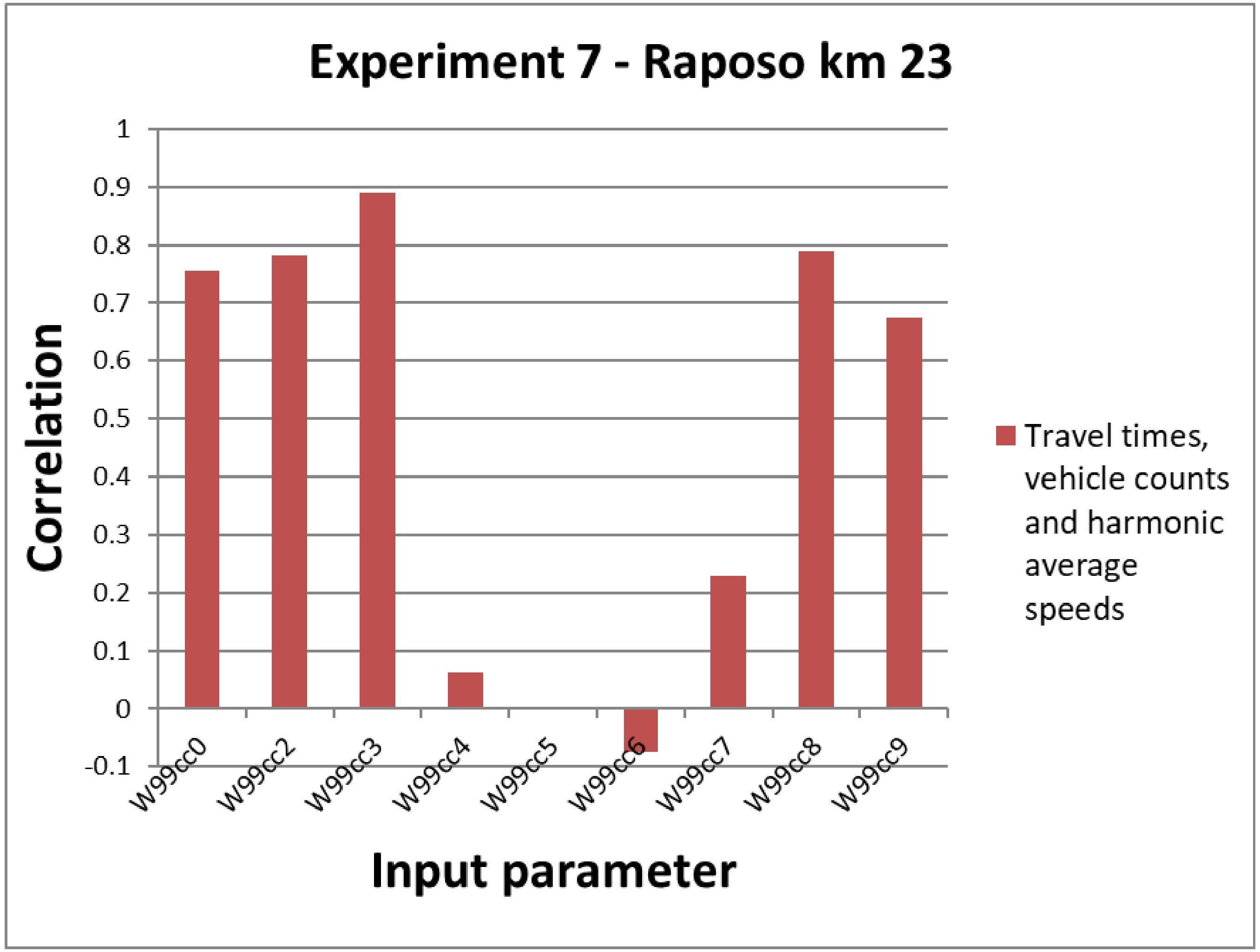
Outputs used	Avg correlation	Worst correlation	Avg correlation w/o worst	RMS test error
• Travel times	0.827453	0.669339	0.880157667	7.90%
• Travel times • Vehicle counts • Harmonic average speeds	0.83053725	0.646007	0.892047333	7.60%

Experiment 7 results I

- W99cc0

- W99cc(2-9)

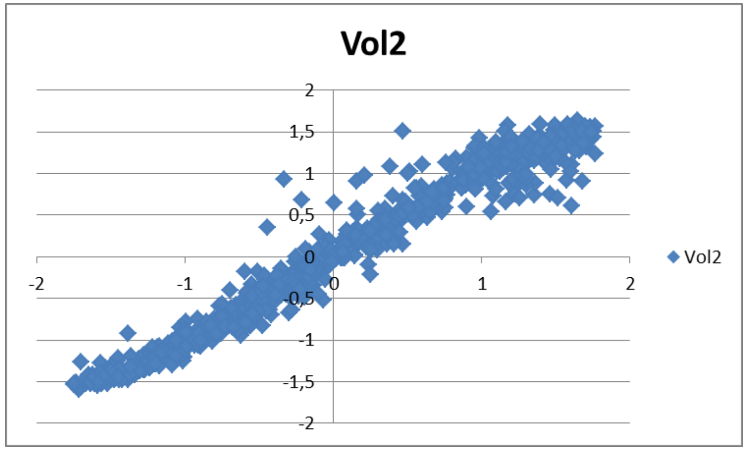
*W99cc1 drop-down menu, not value box

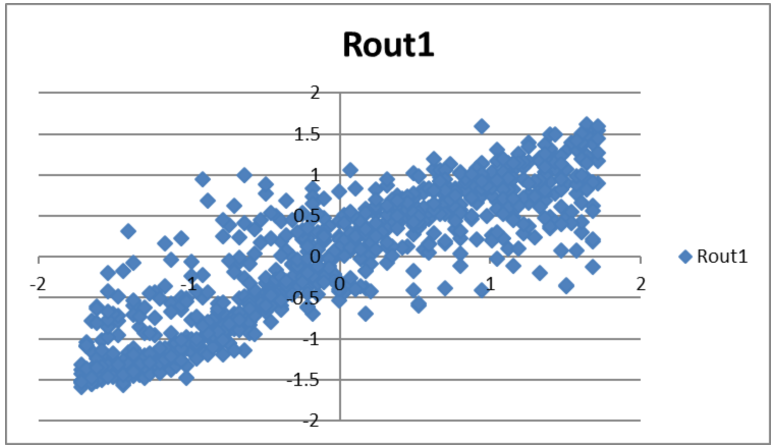


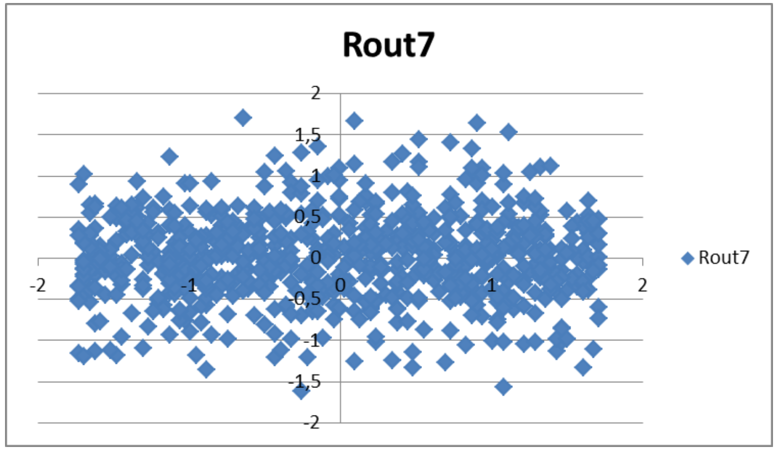
Experiment 7 results II

Outputs used	Avg correlation	Worst correlation	RMS test error
• Travel times • Vehicle counts • Harmonic average speeds	0.456564021	-0.073783329	12.20%

Examples of correlation visualization







- Strong correlation
- Variable can be calibrated automatically by the model

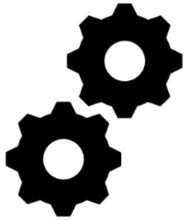
- Degenerate estimates
- Successful automatic calibration depends on tolerable error

- Very weak correlation
- Model can not calibrate the variable

Insights for future work

Initial conclusions

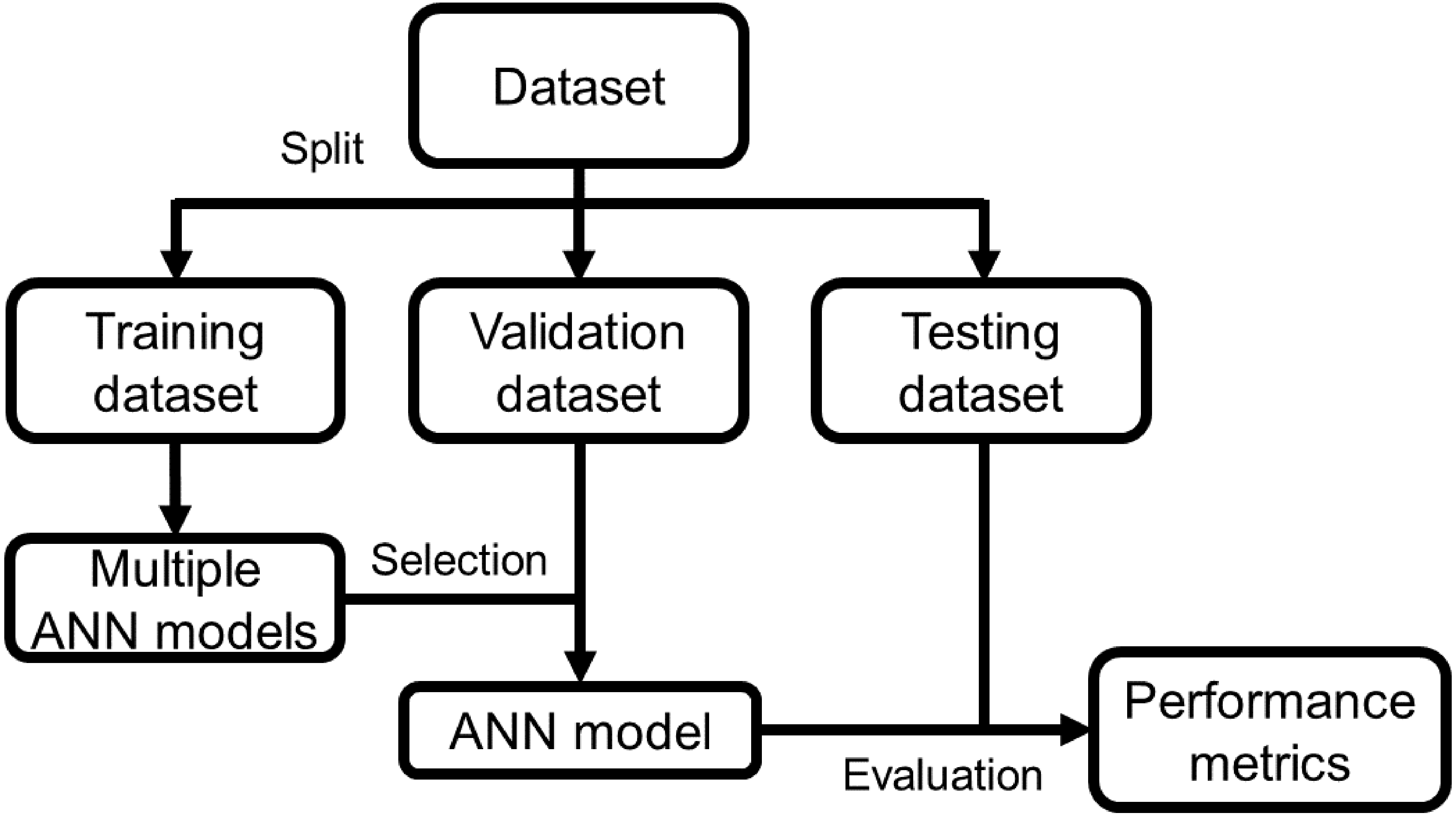
- Automation hypothesis confirmed



- Complexity is adequate for neural networks

- Refinements needed to conclude proof of concept

Neural network topology investigation



New script development environment I

python™

ANACONDA®

TensorFlow

K Keras

New script development environment II

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Acknowledgements

I would like to thank my family, friends and professors for the support in this journey.

Special thanks to Prof. Dr. Claudio Marte, PhD; and PhD student Olímpio Barros for the help from the Traffic Engineering perspective.

References

Stock photos with free license for use: www.pexels.com

ABADI, Martín et al. Tensorflow: a system for large-scale machine learning. In: **OSDI**. 2016. p. 265-283.

CHOLLET, François et al. **Keras: Deep learning library for theano and tensorflow.** Available at <https://keras. io>. Accessed on 24 Oct. 2018.

HOLLANDER, Yaron; LIU, Ronghui. The principles of calibrating traffic microsimulation models. **Transportation**, v. 35, n. 3, p. 347-362, 2008.

References

LOPES, Noel; RIBEIRO, Bernardete. GPU implementation of the multiple back-propagation algorithm. In: **International Conference on Intelligent Data Engineering and Automated Learning**. Springer, Berlin, Heidelberg, 2009. p. 449-456.

RRECAJ, Arlinda Alimehaj; MBOMBOL, Kristi. Calibration and Validation of the VISSIM Parameters-State of the Art. **TEM Journal**, v. 4, n. 3, p. 255, 2015.

Planung Transport Verkehr AG. **PTV Vissim.** Available at <http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/>. Accessed on 24 Oct. 2018.

PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825-2830, 2011.

References

Python Software Foundation. **Python Language Reference**, version 2.7. Available at <http://www.python.org>. Accessed on 24 Oct. 2018.

MOHANTY, Saraju P.; CHOPPALI, Uma; KOUGIANOS, Elias. Everything you wanted to know about smart cities: The internet of things is the backbone. **IEEE Consumer Electronics Magazine**, v. 5, n. 3, p. 60-70, 2016.

XIONG, Zhang et al. Intelligent transportation systems for smart cities: a progress review. **Science China Information Sciences**, v. 55, n. 12, p. 2908-2914, 2012.

Thanks for your attention