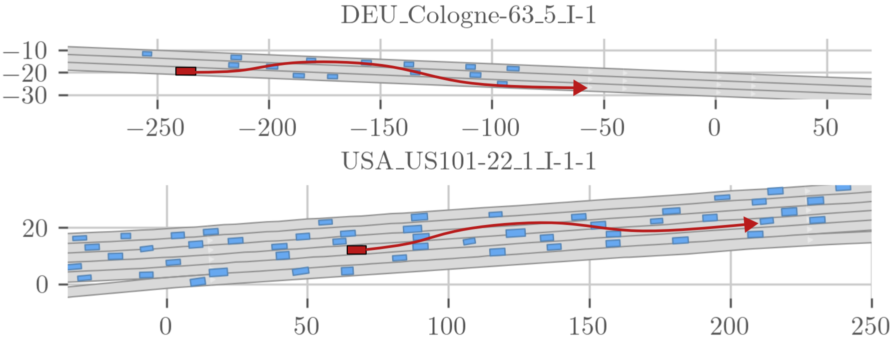
Equivariant Deep Learning of Mixed-Integer Optimal Control Solutions for Vehicle Decision Making and Motion Planning

Rudolf Reiter, Rien Quirynen, Moritz Diehl, Stefano Di Cairano

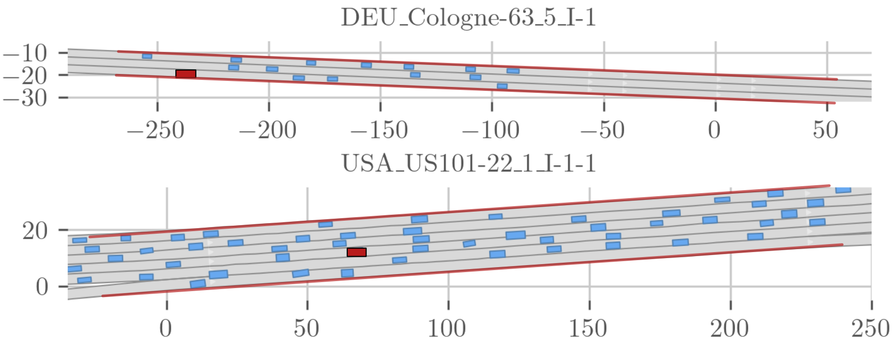
Systems Control and Optimization Laboratory, University of Freiburg

Group Reatreat
SYSCOP
June 10, 2024

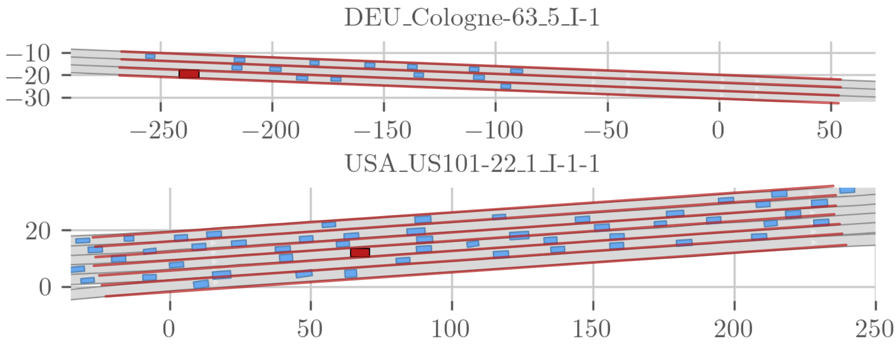
Task: Motion Planning on Highways



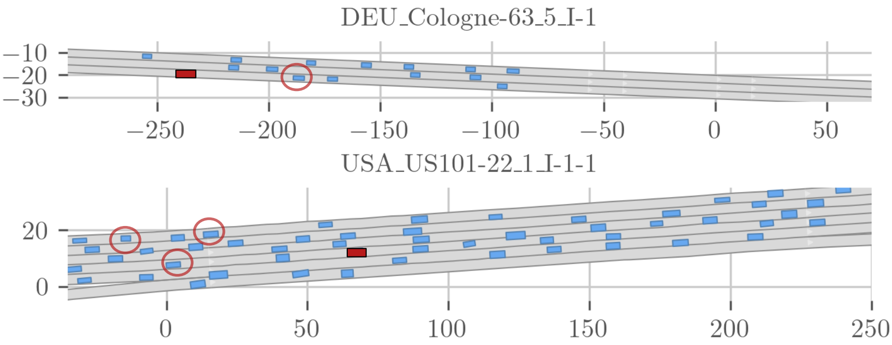
mostly straight



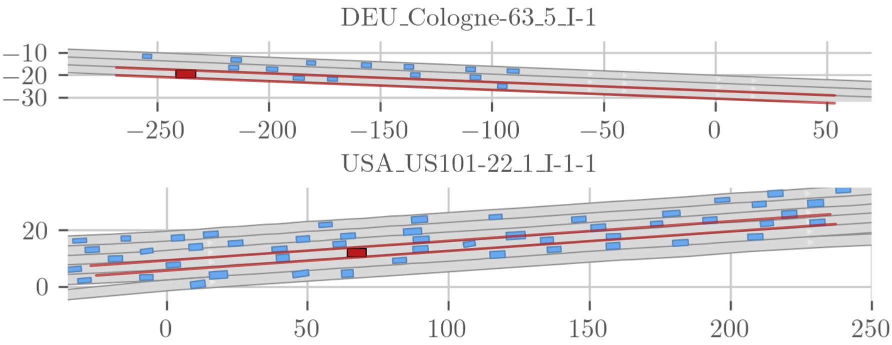
parallel lanes



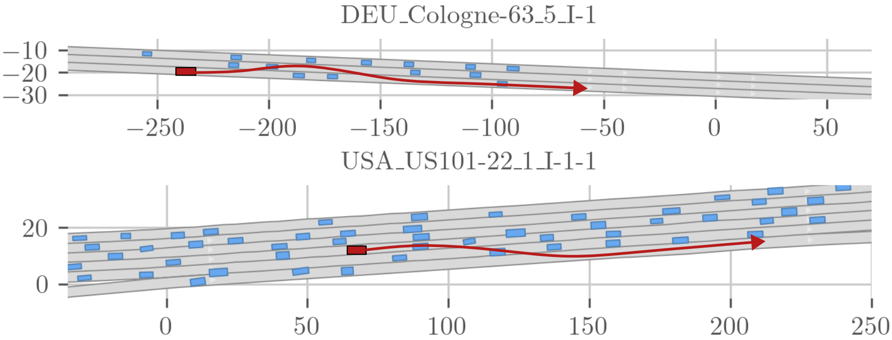
multiple similar obstacles



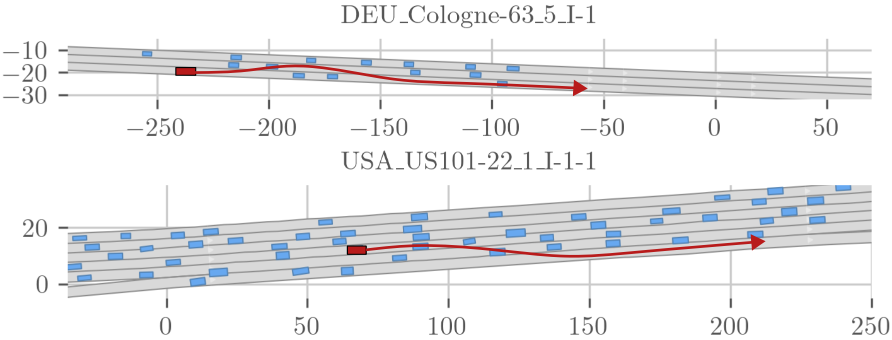
rules: lane keeping



rules: keep right, speed limit



objective: set speed, set lane



$$
\begin{aligned}
\min_{\substack{X\in\mathbb{R}^{n_x\times N},\\ U\in\mathbb{R}^{n_u\times N-1},\\ \beta\in\{0,1\}^{N_\mathrm{b}}}} \quad & J(X,U,\beta) \\
\text{s.t.} \quad & \\
& x_0 = \hat{x} \\
& x_{i+1} = Ax_i + Bu_i, \quad i = 1,\ldots,N-1, \\
& H(X,U) \geq 0, \\
& H_{\mathrm{bin}}(X,\beta) \geq 0
\end{aligned}
$$

[1] Rien Quirynen, Sleiman Safaoui, and Stefano Di Cairano. "Real-time Mixed-Integer Quadratic Programming for Vehicle Decision Making and Motion Planning". In: *ArXiv* (2023). arXiv: 2308.10069 [math.OC].

- Slow online computation time
- MIQP solvers are not usual for embedded hardware
- MIQP solvers are expensive

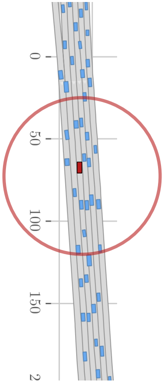
$$
\begin{aligned}
\min_{\substack{X \in \mathbb{R}^{n_x \times N},\\ U \in \mathbb{R}^{n_u \times N-1},\\ \beta \in \{0,1\}^{N_\mathrm{b}}}} \quad & J(X, U, \beta) \\
\text{s.t.} \quad & \\
& x_0 = \hat{x} \\
& x_{i+1} = A x_i + B u_i, \quad i = 1, \dots, N-1, \\
& H(X, U) \geq 0, \\
& H_{\mathrm{bin}}(X, \beta) \geq 0
\end{aligned}
$$

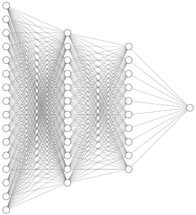
Introduction

Speed-up computation time

learn binary assignments through simulation by supervised learning

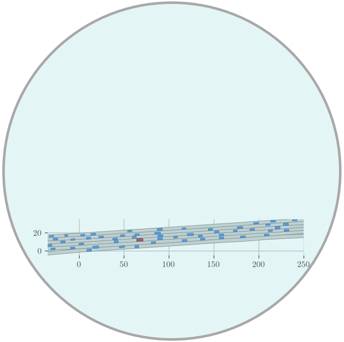
$\rightarrow$ only solve QP $\rightarrow$ much faster than solving MIQP





$$
\begin{aligned}
\min_{\substack{X \in \mathbb{R}^{n_x \times N},\\ U \in \mathbb{R}^{n_u \times N-1}}} \quad & J(X, U; \beta) \\
\text{s.t.} \quad & \\
& x_0 = \hat{x} \\
& x_{i+1} = Ax_i + Bu_i, \quad i = 1, \dots, N-1, \\
& H(X, U) \geq 0, \\
& H_{\text{bin}}(X; \beta) \geq 0
\end{aligned}
$$

motion planning on highways in autonomous driving stack



AD
Motion
Plannning

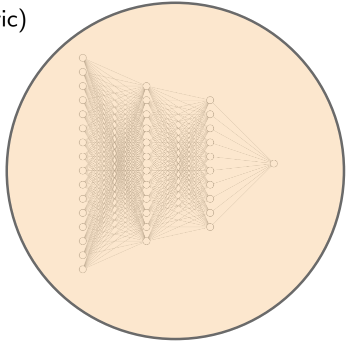
mixed integer optimization problem formulation

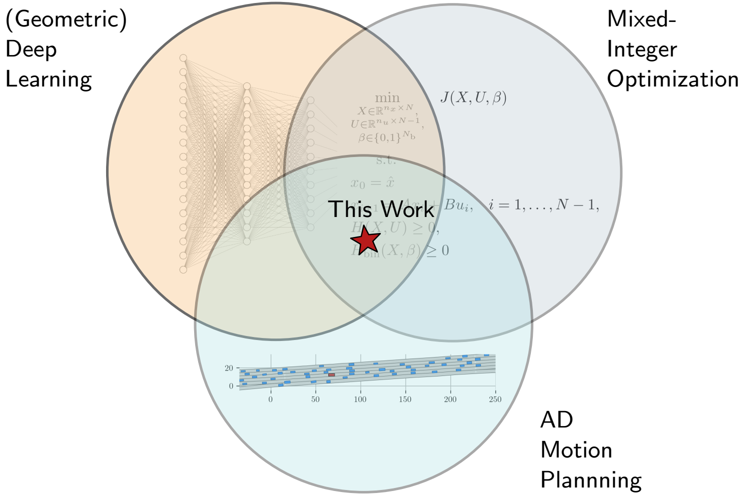
Mixed-Integer Optimization

$$
\begin{aligned}
\min_{\substack{X\in\mathbb{R}^{n_x\times N},\\ U\in\mathbb{R}^{n_u\times N-1},\\ \beta\in\{0,1\}^{N_\mathrm{b}}}} \quad & J(X,U,\beta) \\
\text{s.t.} \quad & \\
& x_0 = \hat{x} \\
& x_{i+1} = Ax_i + Bu_i, \quad i = 1,\dots,N-1, \\
& H(X,U) \geq 0, \\
& H_{\mathrm{bin}}(X,\beta) \geq 0
\end{aligned}
$$

structure-exploiting neural network architecture

(Geometric)
Deep
Learning





(Geometric) Deep Learning
Mixed-Integer Optimization
AD Motion Plannning
This Work
$\min_{X \in \mathbb{R}^{n_x \times N}, U \in \mathbb{R}^{n_u \times N-1}, \beta \in \{0,1\}^{N_b}} J(X,U,\beta)$
s.t.
$x_0 = \hat{x}$
$Bu_i, \quad i = 1, \ldots, N-1,$
$H(X,U) \geq 0,$
≥ 0
20
0
0
50
100
150
200
250

Outline

1. Mixed-Integer Problem Formulation
2. Learning Binary Variables
3. Geometric Deep Learning
4. Additional Concepts
5. Simulation Results

$$
\begin{aligned}
\min_{\substack{X\in\mathbb{R}^{n_x\times N},\\ U\in\mathbb{R}^{n_u\times N-1},\\ \beta\in\{0,1\}^{N_\mathrm{b}}}} \quad & J(X,U,\beta)\\
\text{s.t.} \quad & \\
& x_0 = \hat{x}\\
& x_{i+1} = Ax_i + Bu_i, \quad i=1,\dots,N-1,\\
& H(X,U)\geq 0,\\
& H_{\mathrm{bin}}(X,\beta)\geq 0
\end{aligned}
$$

Mixed-Integer Optimization

1. Mixed-Integer Problem Formulation

Two categories of binary variables:

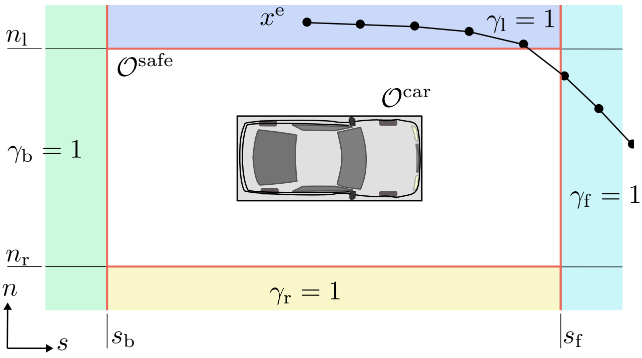
- Expression of nonconvex configuration space as a disjunction of convex sets
- Choice of lane

1. Mixed-Integer Problem Formulation

Two categories of binary variables:

- Expression of nonconvex configuration space as a disjunction of convex sets (**Why?**)
- Choice of lane

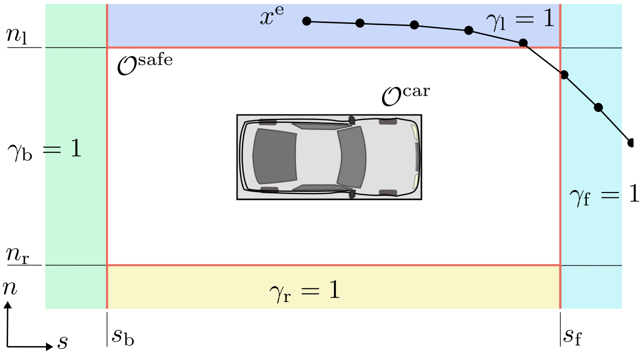
- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$



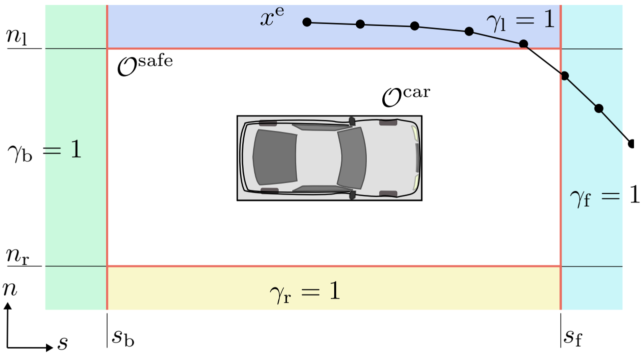
1. Mixed-Integer Problem Formulation

Nonconvex free configuration space

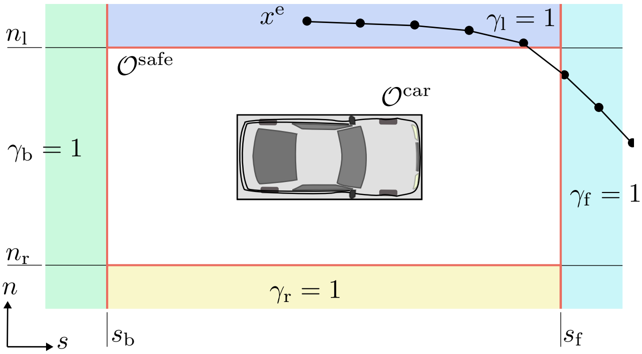
- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$
- Split configuration space $\mathcal{F} = \mathbb{R}^2 \backslash \mathcal{O}^{\mathrm{safe}}$ into 4 convex sets (left, right, front, back)



- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$
- Split configuration space $\mathcal{F} = \mathbb{R}^2 \backslash \mathcal{O}^{\mathrm{safe}}$ into 4 convex sets (left, right, front, back)
- Assign binary indicator variables $\gamma_{\mathrm{l}}, \gamma_{\mathrm{r}}, \gamma_{\mathrm{f}}, \gamma_{\mathrm{b}} \in \{0, 1\}$



- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$
- Split configuration space $\mathcal{F} = \mathbb{R}^2 \backslash \mathcal{O}^{\mathrm{safe}}$ into 4 convex sets (left, right, front, back)
- Assign binary indicator variables $\gamma_{\mathrm{l}}, \gamma_{\mathrm{r}}, \gamma_{\mathrm{f}}, \gamma_{\mathrm{b}} \in \{0, 1\}$
- 4 binary variables per obstacle per time step: $4N_{\mathrm{obs}}N$ binary variables



- Adding reference state as decision variable $\tilde{n}$, with $\tilde{X}_{\mathrm{n}} = [\tilde{n}_0, \dots, \tilde{n}_N]^\top$
- Adding binary lane change control variables $\lambda^{\mathrm{up}}, \lambda^{\mathrm{down}}$
- Reference dynamics:

$$\tilde{n}_{i+1} = \tilde{n}_i + d_{\mathrm{lane}} \lambda_i^{\mathrm{up}} - d_{\mathrm{lane}} \lambda_i^{\mathrm{down}}$$

- Tracking cost for lateral state n:

$$\sum_{i=0}^{N} w_n (\tilde{n}_i - n_i)^2$$

- adding $2N$ lane change binary variables to a total

$$N_{\mathrm{bin}} = 2N + 4N N_{\mathrm{obs}}$$

1. Mixed-Integer Problem Formulation

$$\min_{\substack{X\in\mathbb{R}^{n_x\times N}, U\in\mathbb{R}^{n_u\times N-1},\\ \tilde{X}_n\in\mathbb{R}^{N},\\ \Gamma\in\{0,1\}^{4NN_{\text{obs}}},\\ \Lambda\in\{0,1\}^{2N}}} \quad J(X,U,\tilde{X}_n)$$

$$\text{s.t.}$$

$$\begin{aligned}
& x_0 = \hat{x}, \quad \tilde{n}_0 = \hat{n}_0, && \\
& x_{i+1} = Ax_i + Bu_i, && i = 0,\dots,N-1, \\
& \tilde{n}_{i+1} = \tilde{n}_i + d_{\text{lane}}\lambda_i^{\text{up}} - d_{\text{lane}}\lambda_i^{\text{down}}, && i = 0,\dots,N-1, \\
& H(X,U) \geq 0, && \\
& H_{\text{obs}}(x_i, (\gamma_{\text{d}})_{i,j}) \geq 0 && i = 0,\dots,N-1, \\
& && j = 1,\dots,N_{\text{obs}}
\end{aligned}$$

2. Learning Binary Variables

Generate training data and class labels

- Randomize problem features p_i
  - ego state
  - obstacle states
  - vehicle dimensions
  - road geometry
- Solve MIQPs to obtain binary assignments $\beta_i^\star = (\Gamma_i^\star, \Lambda_i^\star)$
- If $\beta_i^\star$ not in data set $\mathcal{D}$: generate class label l_i and add $(p_i, l_i, \beta_i^\star)$ to $\mathcal{D}$
- If $\beta_i^\star$ in data set $\mathcal{D}$: use already existing label l_j, with $\beta_i^\star = \beta_j^\star$ and add $(p_i, l_j, \beta_j^\star)$ to $\mathcal{D}$

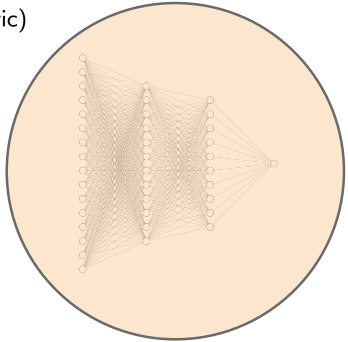
Use data set $\mathcal{D}$ to train a classifier that predicts $\beta^\star$

- Labels learned by classification as opposed to regression
- Number of assignments in theory $2^{4NN_{\mathrm{obs}}+2N}$
- If assignment was never seen in data, no label exists
- Shown to perform better than regression[2]

[2] Dimitris Bertsimas and Bartolomeo Stellato. "The voice of optimization". en. In: *Machine Learning* 110.2 (Feb. 2021), pp. 249–277. ISSN: 1573-0565. DOI: 10.1007/s10994-020-05893-5.

3. Geometric Deep Learning

(Geometric)
Deep
Learning



3. Geometric Deep Learning

Geometric Deep Learning
Grids, Groups, Graphs,
Geodesics, and Gauges

Michael M. Bronstein[1], Joan Bruna[2], Taco Cohen[3], Petar Veličković[4]

May 4, 2021

3. Geometric Deep Learning

Fundamentally, geometric deep learning involves encoding a geometric understanding of data as an inductive bias in deep learning models to give them a helping hand.

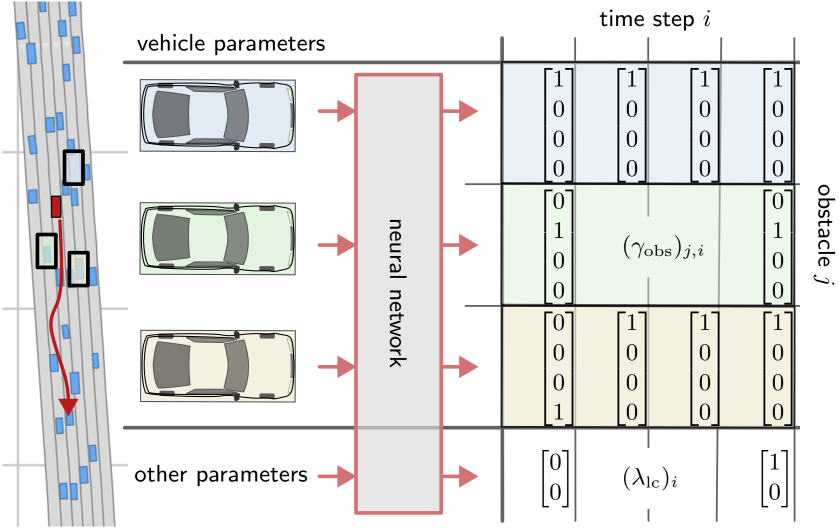
Definition

Let $f(x) : \mathbb{X}^M \to \mathbb{Y}$ be a function on a set of variables $x = \{x_1, \dots, x_M\} \in \mathbb{X}^M$ and let $\mathcal{G}$ be the permutation group on $\{1, \dots, M\}$. The function f is **permutation invariant**, if $f(g \cdot x) = f(x)$ for all $g \in \mathcal{G}, x \in \mathbb{X}^M$.

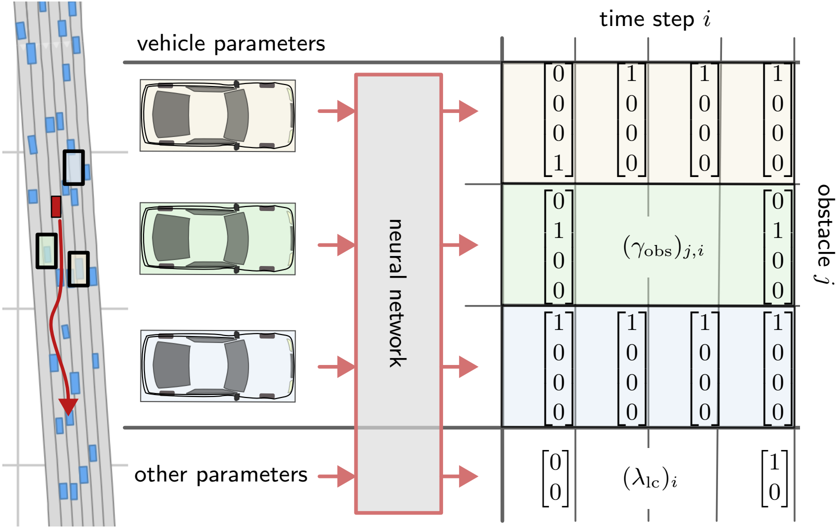
Definition

Let $f(x) : \mathbb{X}^N \to \mathbb{Y}^N$ be a function on a set of variables $x = \{x_1, \dots, x_N\} \in \mathbb{X}^N$ and let $\mathcal{G}$ be the permutation group on $\{1, \dots, N\}$. The function f is **permutation equivariant**, if $f(g \cdot x) = g \cdot f(x)$ for all $g \in \mathcal{G}, x \in \mathbb{X}^N$.

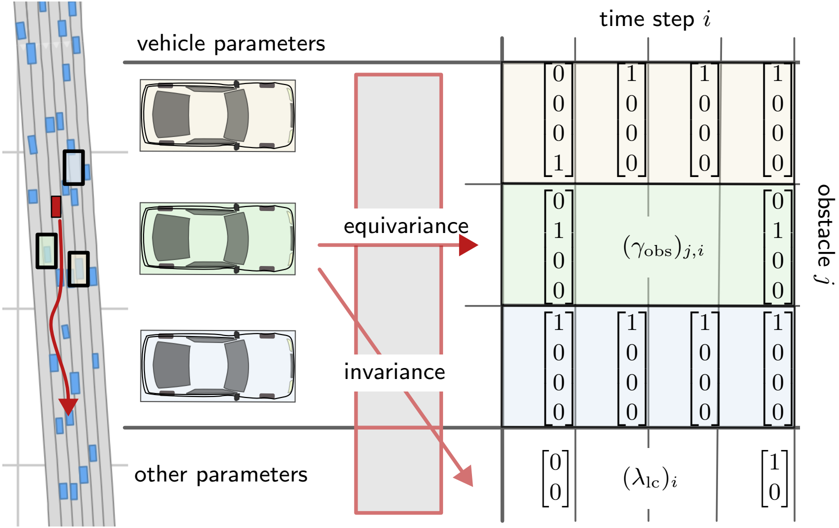
Why is invariance and equivariance interesting for our task?



time step i
vehicle parameters
neural network
obstacle j
$(\gamma_{\text{obs}})_{j,i}$
other parameters
$(\lambda_{\text{lc}})_i$



time step i
vehicle parameters
neural network
obstacle j
$(\gamma_{\mathrm{obs}})_{j,i}$
other parameters
$(\lambda_{\mathrm{lc}})_i$

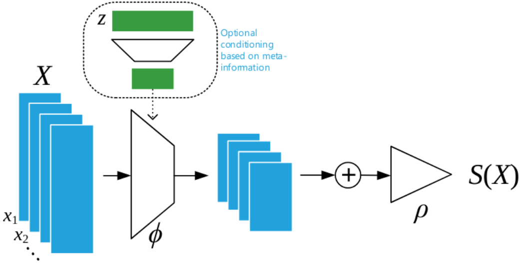


time step i
vehicle parameters
equivariance
invariance
$(\gamma_{\text{obs}})_{j,i}$
obstacle j
other parameters
$(\lambda_{\text{lc}})_i$

Number N_{p} of permutations of n elements is $N_{\mathrm{p}} = n!$
For 10 obstacles this would be $N_{\mathrm{p}} = 3628800$ different scenarios, while they all correspond to only one scenario

Let $x \in \mathbb{R}^D$ be features of a set element, $P \in \mathbb{R}^{M \times M}$ a permutation matrix and the matrix $X = (x_1, \dots, x_m)^\top \in \mathbb{R}^{M \times D}$ stacks the features as rows. A function $f(\cdot)$ is permutation invariant, iff $f(X) = f(PX)$. One permutation invariant function is

$$f(X) = \rho(\sum_{m=1}^{M} \phi(x_m))$$

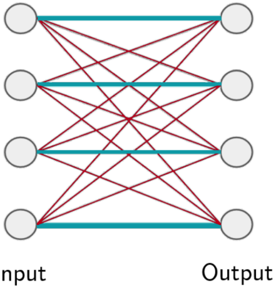


[3]Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

Permutation Equivariant Layers[4]

The function $f_\Theta(X) = \sigma(\Theta X)$, with $D = 1$, $\Theta \in \mathbb{R}^{M \times M}$, $X \in \mathbb{R}^M$ and $f_\Theta : \mathbb{R}^M \to \mathbb{R}^M$ is permutation equivariant iff all the off-diagonal elements of Θ are tied together and all the diagonal elements are equal as well. That is,

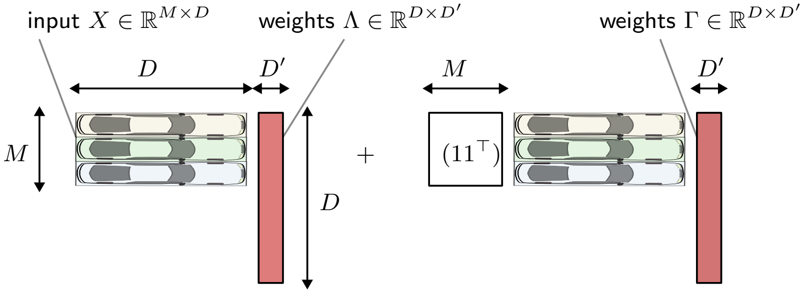
$$\Theta = \lambda I + \gamma(11^\top), \quad \lambda, \gamma \in \mathbb{R}, \quad 1^\top = [1, \dots, 1]^\top \in \mathbb{R}^M, \quad I \in \mathbb{R}^{M \times M} \text{ is identity}$$



[4] Zaheer et al., "Deep Sets".

This result can be easily extended to higher dimensions, i.e., D input and D' output channels. Then, $X \in \mathbb{R}^{M\times D}$, $y \in \mathbb{R}^{M\times D'}$, λ, γ become matrices $\Lambda, \Gamma \in \mathbb{R}^{D\times D'}$.

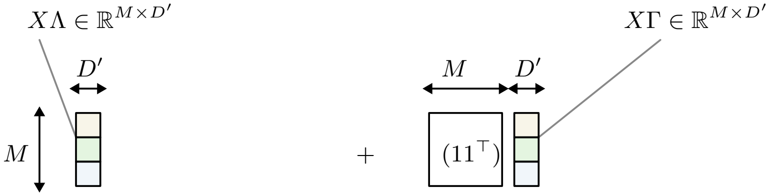
Layer function: $f(X) = \sigma(X\Lambda - (11^\top)X\Gamma)$



[5] Zaheer et al., "Deep Sets".

This result can be easily extended to higher dimensions, i.e., D input and D' output channels.
Then, $X \in \mathbb{R}^{M \times D}$, $y \in \mathbb{R}^{M \times D'}$, λ, γ become matrices $\Lambda, \Gamma \in \mathbb{R}^{D \times D'}$.

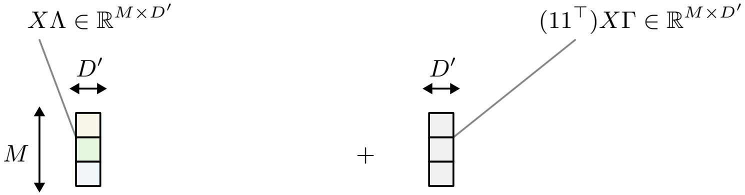
Layer function: $f(X) = \sigma(X\Lambda - (11^\top)X\Gamma)$



[6] Zaheer et al., "Deep Sets".

This result can be easily extended to higher dimensions, i.e., D input and D' output channels. Then, $X \in \mathbb{R}^{M \times D}$, $y \in \mathbb{R}^{M \times D'}$, λ, γ become matrices $\Lambda, \Gamma \in \mathbb{R}^{D \times D'}$.

Layer function: $f(X) = \sigma(X\Lambda - (11^\top)X\Gamma)$

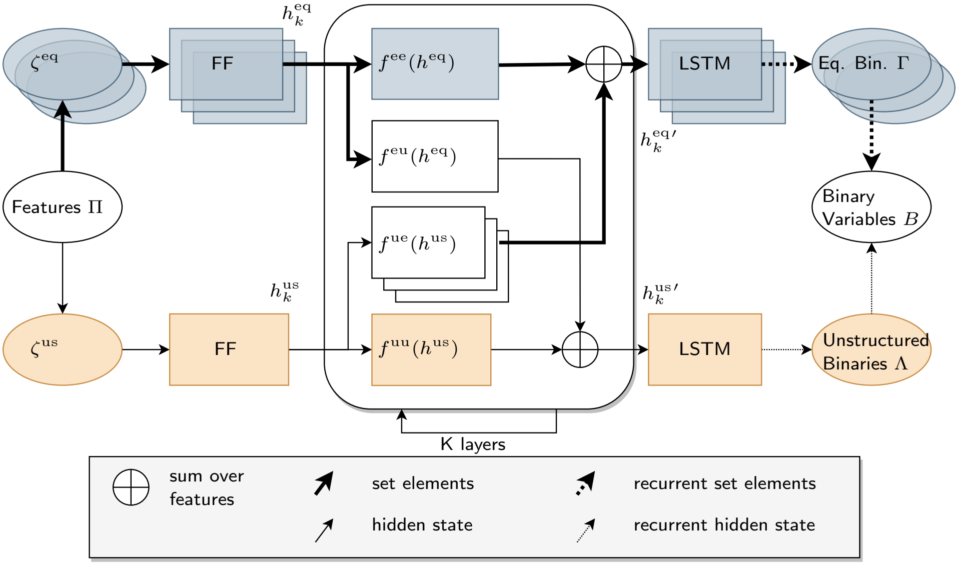


[7] Zaheer et al., "Deep Sets".

- The prediction is a time series of binary assignments $\rightarrow$ using a recurrent *decoder* to generate a time series[8]
- Allows for variable length predictions in addition to the variable number of obstacles

[8] Abhishek Cauligi et al. "PRISM: Recurrent Neural Networks and Presolve Methods for Fast Mixed-integer Optimal Control". In: *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*. Vol. 168. Proceedings of Machine Learning Research. PMLR, 2022, pp. 34–46.

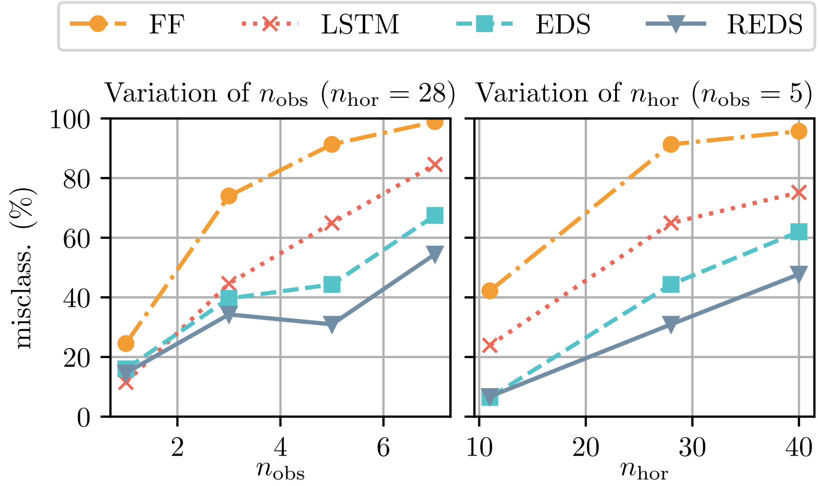
Final Recurrent Equivariant Deep Set Architecture



4. Additional Concepts

- Slacked QP: After predicting the binary variables, the remaining QP is solved with slacks on the fixed binary variables
- NN Ensemble: Several differently trained neural networks and slacked QPs are solved in parallel $\rightarrow$ lowest-cost solution is chosen
- Feasibility Projection: To enhance safety, an additional NLP is solved with nonlinear obstacle constraints to project possibly unsafe trajectories
- Lowest-level MPC: Lowest-level MPC tracks the planned trajectory

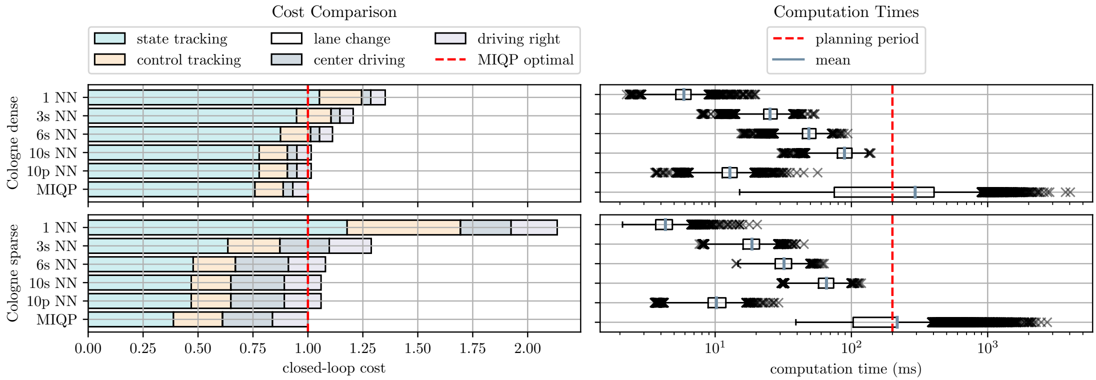
- Comparing the share of wrong predictions (misclassification) of all binary variables on test data set
- Architectures
  - Feed Forward (FF)
  - Long Short Term Memory (LSTM)
  - Equivariant and Invariant Deep Sets (EDS)
  - Equivariant, Invariant Layers and LSTM decoder (REDS)


FF
LSTM
EDS
REDS
Variation of n_{obs} ($n_{\mathrm{hor}} = 28$)
Variation of n_{hor} ($n_{\mathrm{obs}} = 5$)
misclass. (%)
100
80
60
40
20
0
2
4
6
n_{obs}
10
20
30
40
n_{hor}

- Comparing expert MIQP with proposed stack:
  - ensemble of $(1$ to $10)$ REDS networks for predictions of binaries
  - slacked QP
  - feasibility projector
- Both variants followed by a lowest-level NMPC tracking controller
- On randomized CommonRoad Cologne highway scenarios with SUMO backend

5. Results

Comparison in closed-loop simulations



Outlook

Interesting further work

- Diving deeper into geometric deep learning
  - Using geometric deep learning for other control systems tasks (e.g., exploiting invariances to other groups, such as Euclidean group)
  - Finding more generic layers for any MIQP (graph neural networks, transformers)
- More applications
  - Applying structure to large SUMO simulations for coordinating traffic
  - Multi-agent coordination of e.g., drones
- Improving the algorithm
  - Conditioned predictions along time axis to generate multiple prediction candidates
  - "Sandwiching" equivariant and recurrent layers

Thanks to the Coauthors!

Thank you for your attention!