The power of random

By RandomBaseline

Ideas

- Dynamical Systems ✗ → not achievable in a weekend
- Road prediction + linear approximation ✗ → good road prediction not feasible
- Backpropagate the target function directly ✗ → no differentiable objective function
- ANN trained with Genetic Algorithm ✓

Genetic Algorithm – Advantages

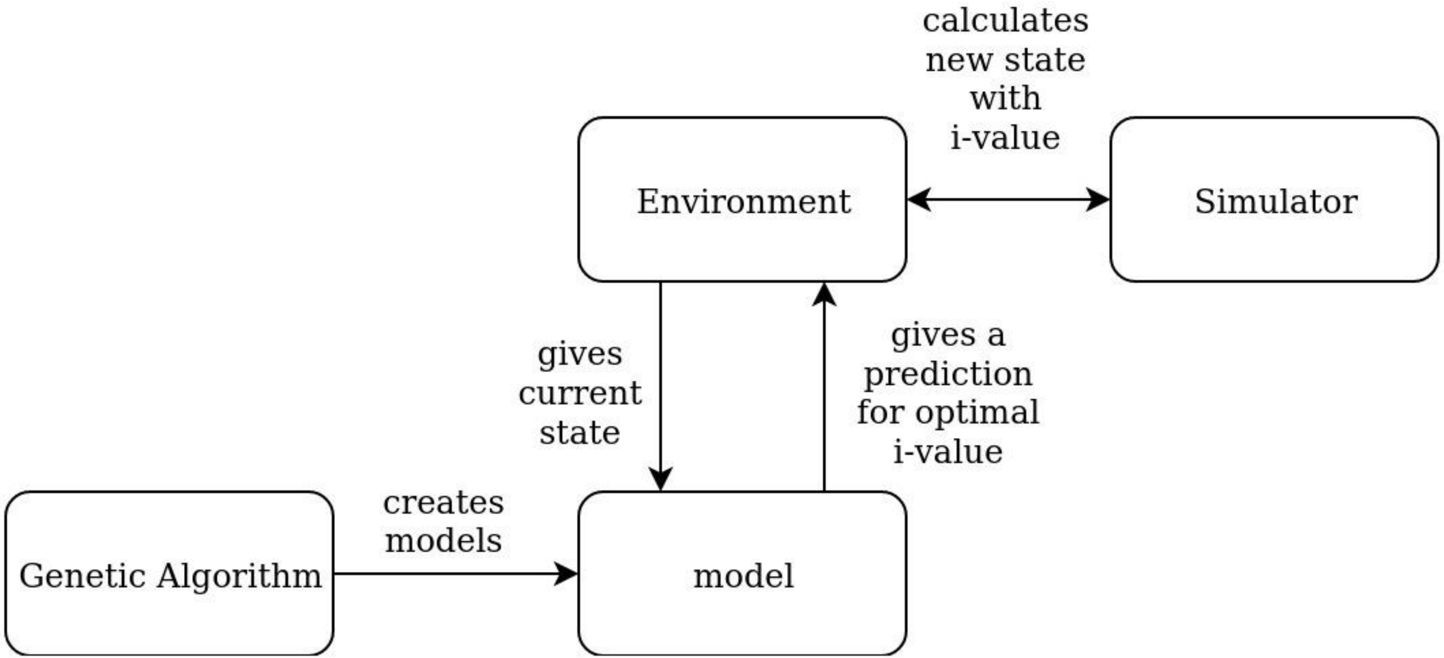
- non-gradient based (T_{Target} is not differentiable)

- parallel search (less likely to get stuck in local optima)

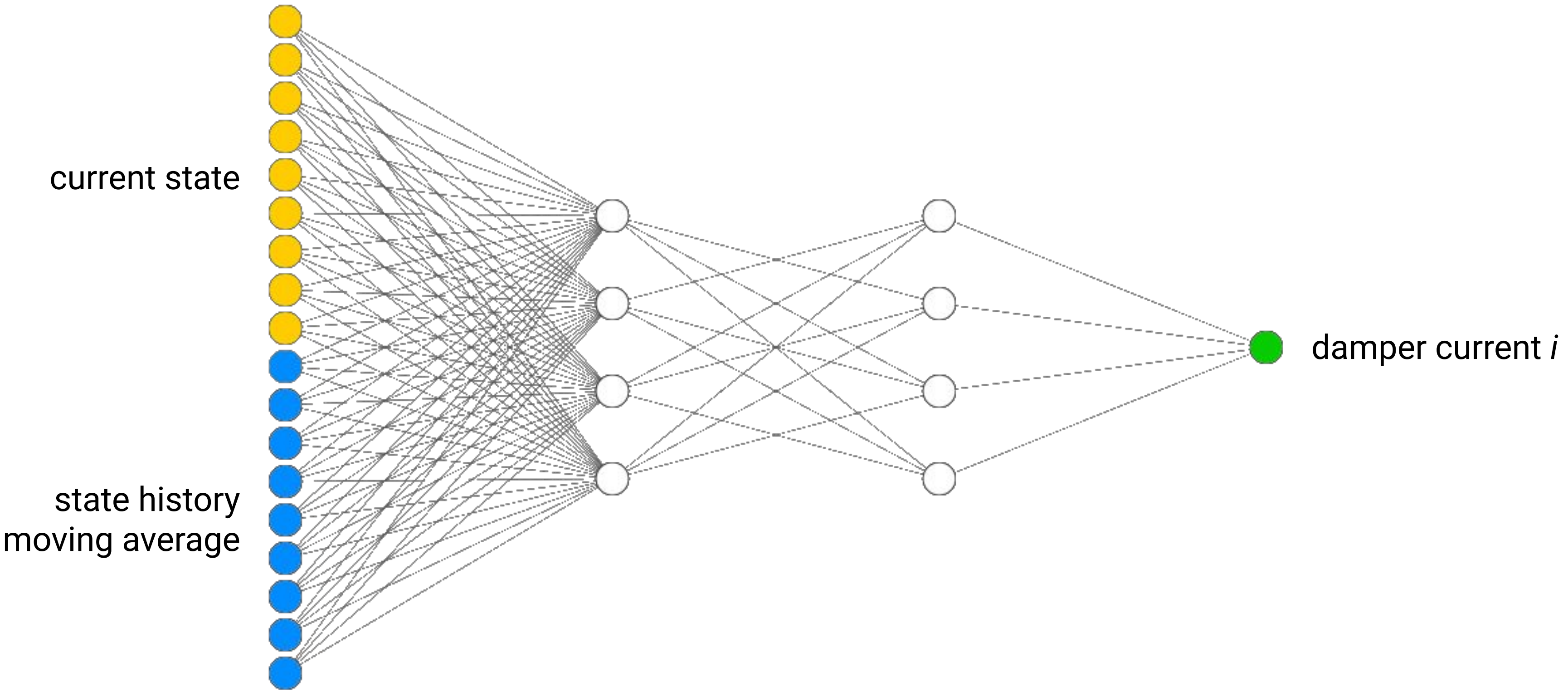
- can optimize for multiple objectives (constraint & target)

- explainable method

Design – Code


calculates new state with i-value
Environment
Simulator
gives current state
gives a prediction for optimal i-value
creates models
Genetic Algorithm
model

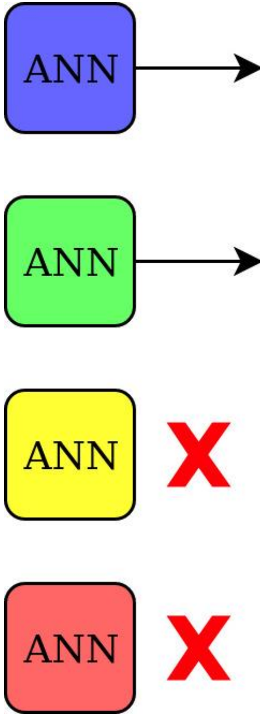
Design - Artificial Neural Network (ANN)



Design – Genetic Algorithm

Fitness function:

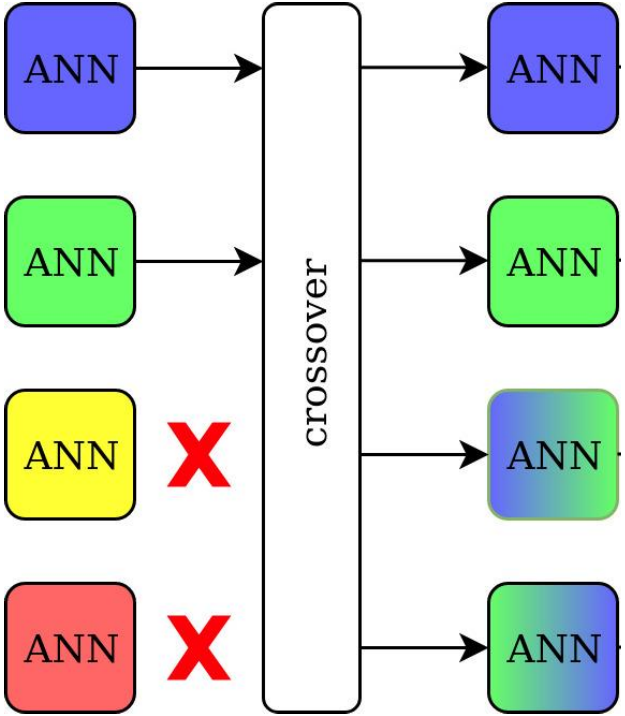
$$f(x) = \begin{cases} T_{Target} & constraint = true \\ T_{Target} \cdot 10 & else \end{cases}$$



Design – Genetic Algorithm

Fitness function:

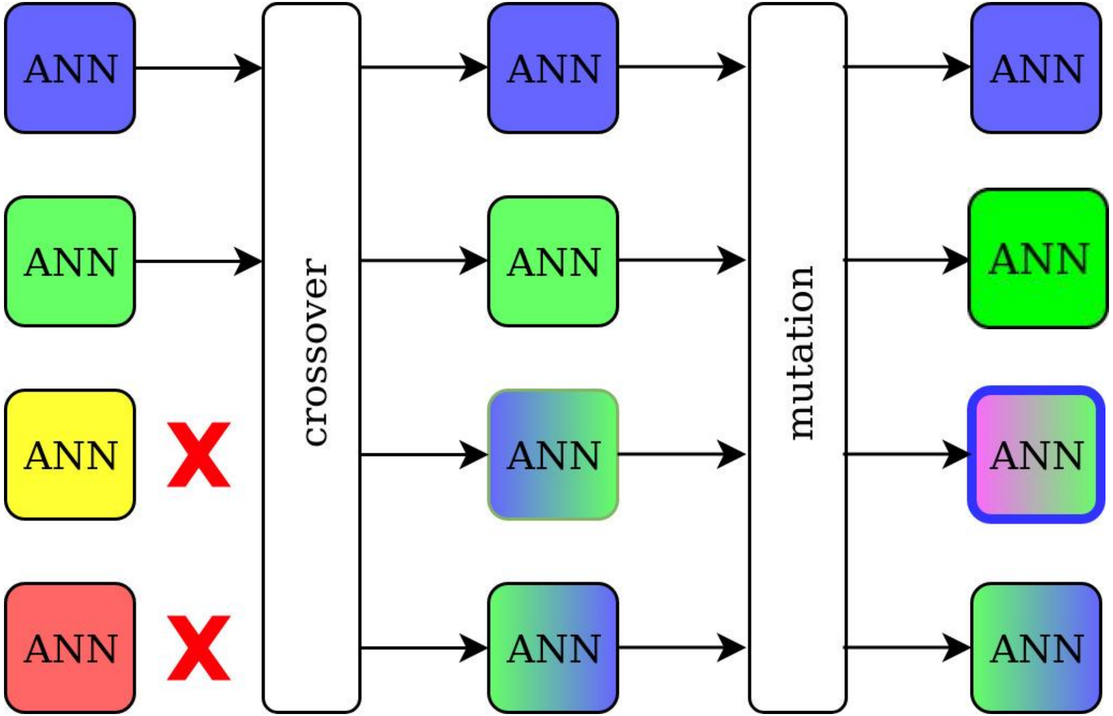
$$f(x) = \begin{cases} T_{Target} & constraint = true \\ T_{Target} \cdot 10 & else \end{cases}$$



Design – Genetic Algorithm

Fitness function:

$$f(x) = \begin{cases} T_{Target} & constraint = true \\ T_{Target} \cdot 10 & else \end{cases}$$

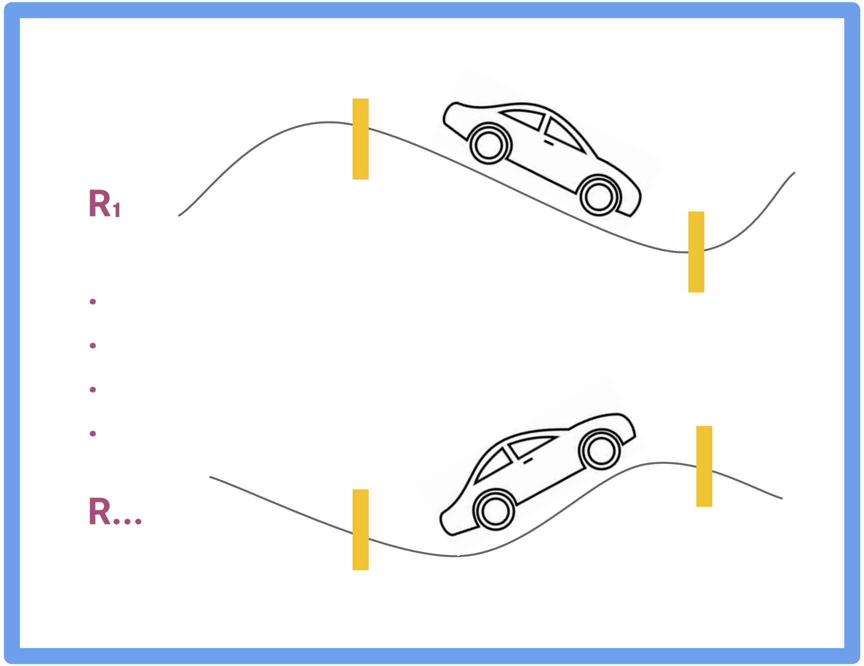


Training - Hyperparameters

- **EPOCHS:** in an epoch a population is evaluated and updated

- **EVALUATION REPEATS:** how many road positions we look at for each epoch

- **EVALUATION STEPS:** how many steps we train from the picked road position onward

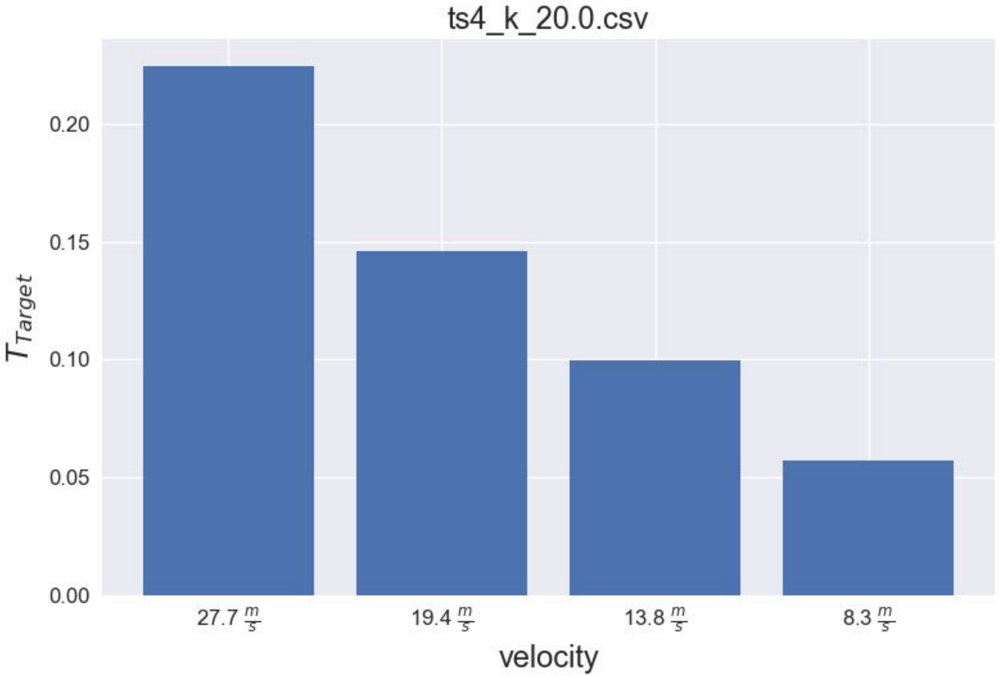


Results

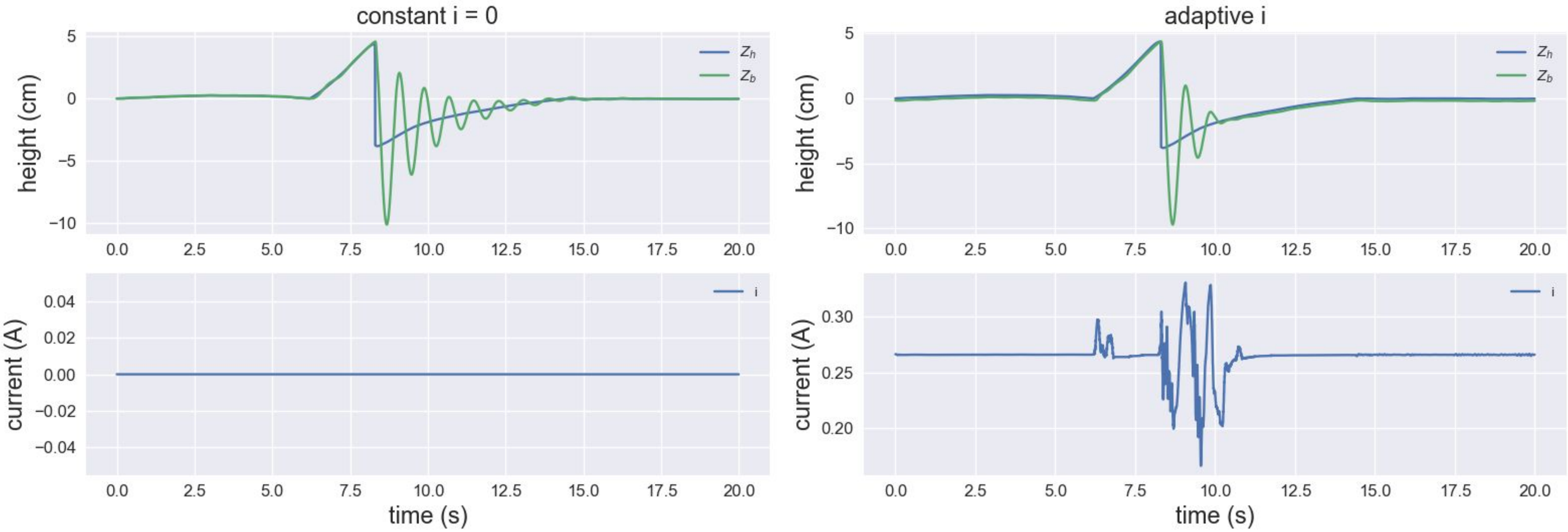
- including a moving average improved performance

- adding convolution did not improve performance directly
    - might achieve better results after longer training

- our model achieves T_{Target} values lower than simulated passive dampers (constant *i*)



Constant vs adaptive i – Example



ANN inference speed

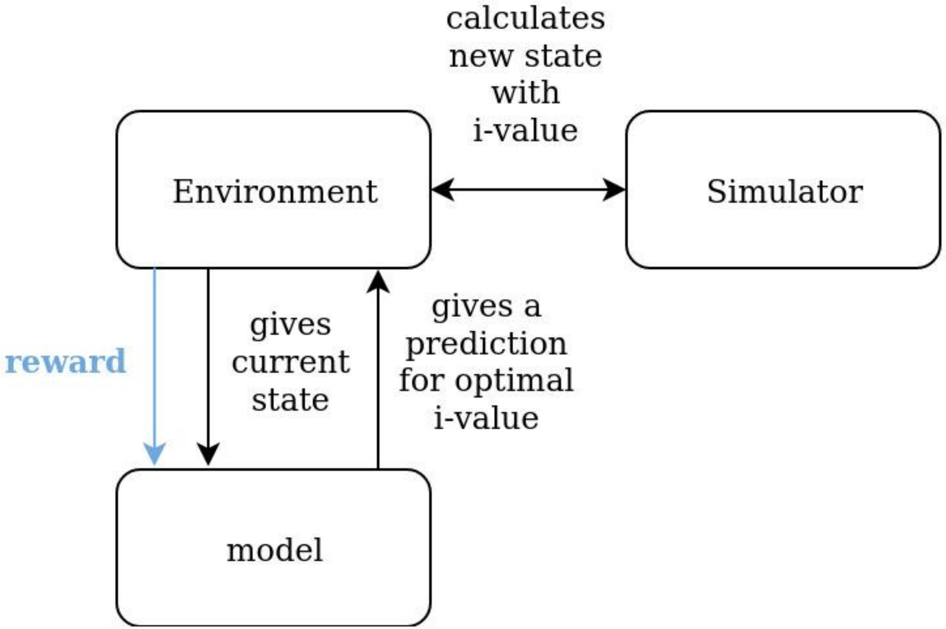
0,25ms

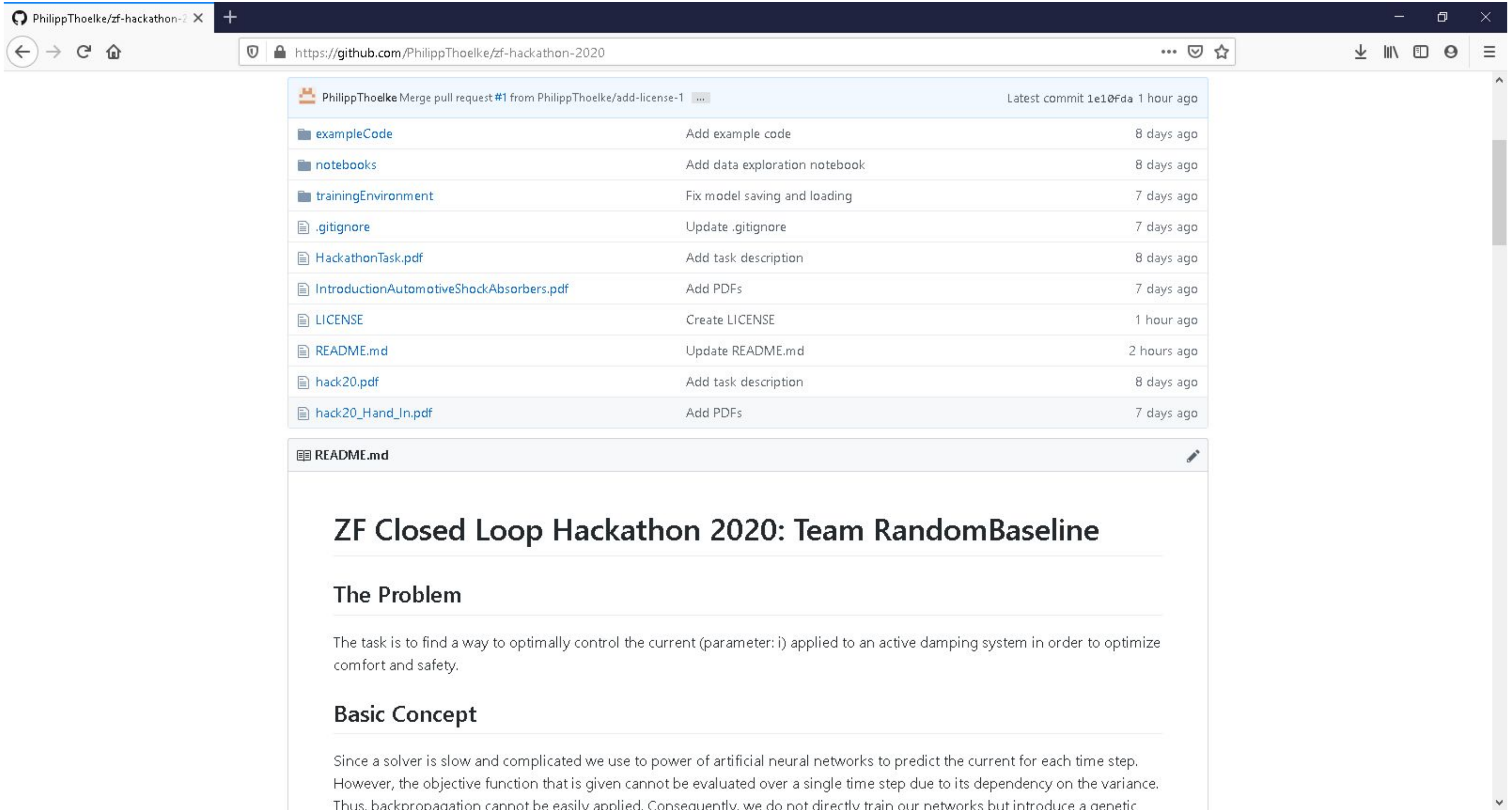
Outlook – Reinforcement Learning

- same environment can be used

- more directed learning → might converge faster

- challenge: continuous action space, delayed rewards


calculates new state with i-value
Environment
Simulator
reward
gives current state
gives a prediction for optimal i-value
model


PhilippThoelke/zf-hackathon-2
https://github.com/PhilippThoelke/zf-hackathon-2020

PhilippThoelke Merge pull request #1 from PhilippThoelke/add-license-1 … Latest commit 1e10fda 1 hour ago

exampleCode	Add example code	8 days ago
notebooks	Add data exploration notebook	8 days ago
trainingEnvironment	Fix model saving and loading	7 days ago
.gitignore	Update .gitignore	7 days ago
HackathonTask.pdf	Add task description	8 days ago
IntroductionAutomotiveShockAbsorbers.pdf	Add PDFs	7 days ago
LICENSE	Create LICENSE	1 hour ago
README.md	Update README.md	2 hours ago
hack20.pdf	Add task description	8 days ago
hack20_Hand_In.pdf	Add PDFs	7 days ago

README.md

ZF Closed Loop Hackathon 2020: Team RandomBaseline

The Problem

The task is to find a way to optimally control the current (parameter: i) applied to an active damping system in order to optimize comfort and safety.

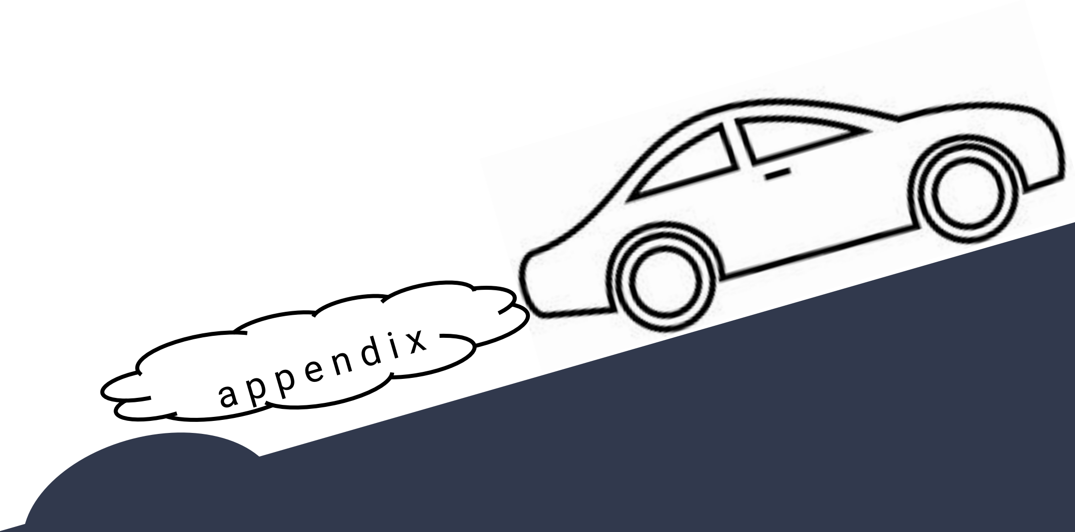
Basic Concept

Since a solver is slow and complicated we use to power of artificial neural networks to predict the current for each time step. However, the objective function that is given cannot be evaluated over a single time step due to its dependency on the variance. Thus, backpropagation cannot be easily applied. Consequently, we do not directly train our networks but introduce a genetic

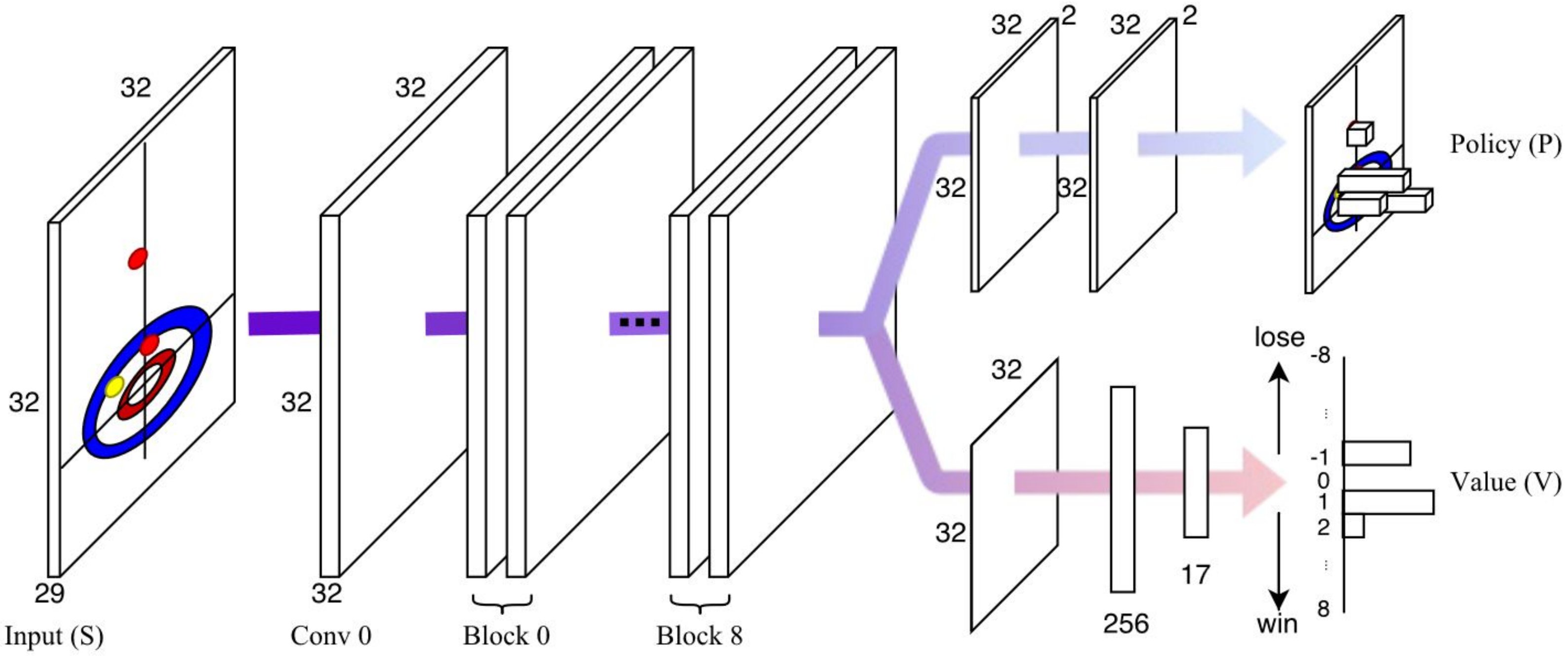
GitHub: https://github.com/PhilippThoelke/zf-hackathon-2020

Thank you.

Questions?



RL continuous action space - Model architecture



http://proceedings.mlr.press/v80/lee18b/lee18b.pdf