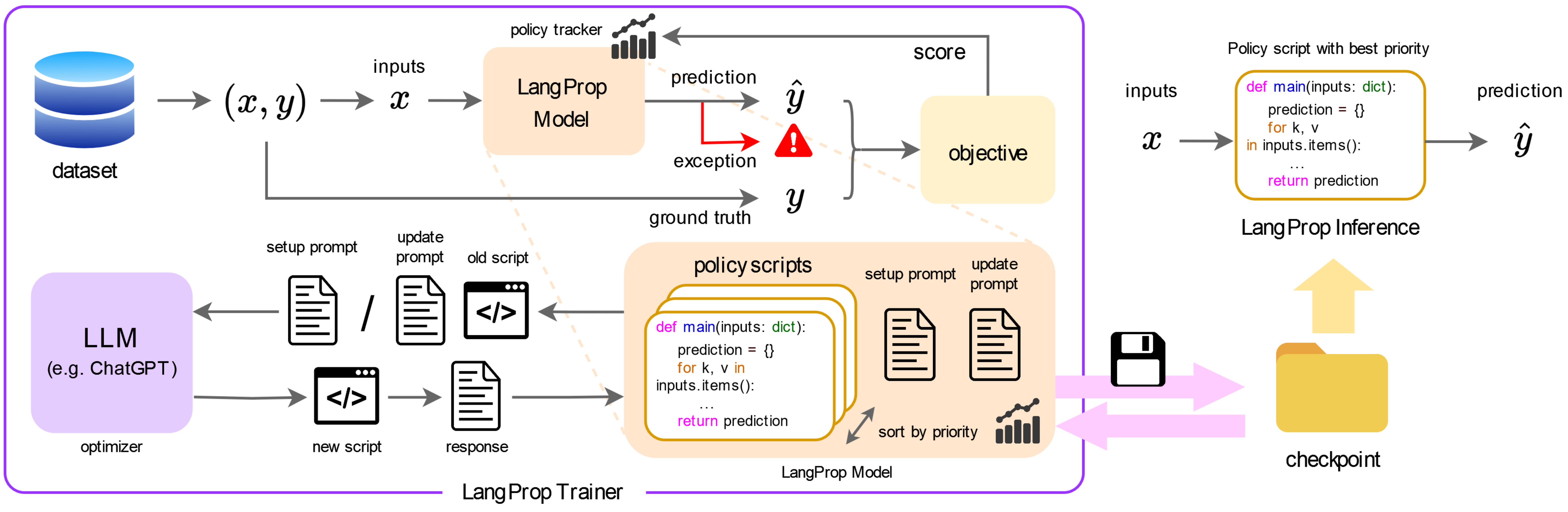
LangProp: A code optimization framework using Large Language Models applied to driving

Shu Ishida, Gianluca Corrado, George Fedoseev, Hudson Yeo, Lloyd Russell, Jamie Shotton, João F. Henriques, Anthony Hu

research@wayve.ai



Training symbolic systems with LangProp

- LangProp iteratively optimizes code with LLMs to maximize an objective.
- Analogy from classical ML training: LLM = ***optimizer***; code = ***parameters***.
- Code is run on a dataset, reranked by scores, and updated with an LLM.
- LangProp supports both **supervised** and **reinforcement learning**.

github.com/shuishida/LangProp

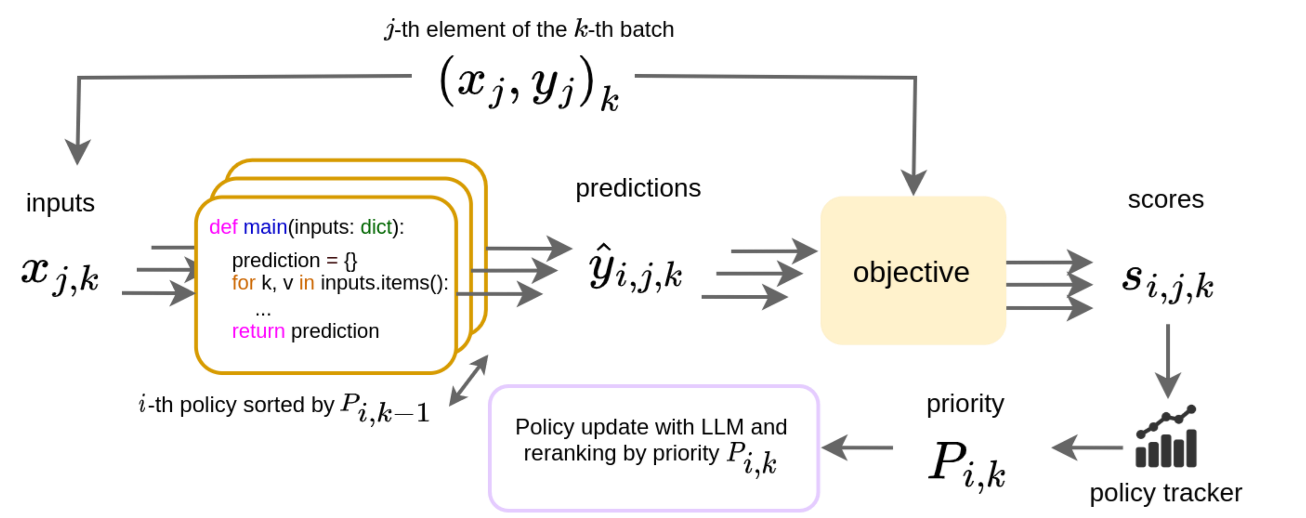
```python
train_loader = DataLoader(train_data, batch_size, shuffle=True, collate_fn=lambda x: x)
val_loader = DataLoader(val_data, batch_size, shuffle=True, collate_fn=lambda x: x)
model = LPModule.from_template(name, root)
trainer = LPTrainer(model, RunConfig(run_name))
trainer.fit(train_loader, val_loader, epochs=epochs)
```

Training a LangProp model (similar interface to PyTorch Lightning)

```python
model = LPModule.from_checkpoint(checkpoint)
model.setup(config=RunConfig())
prediction = model(*input_args, **input_kwargs)
```

Inference with a LangProp model checkpoint

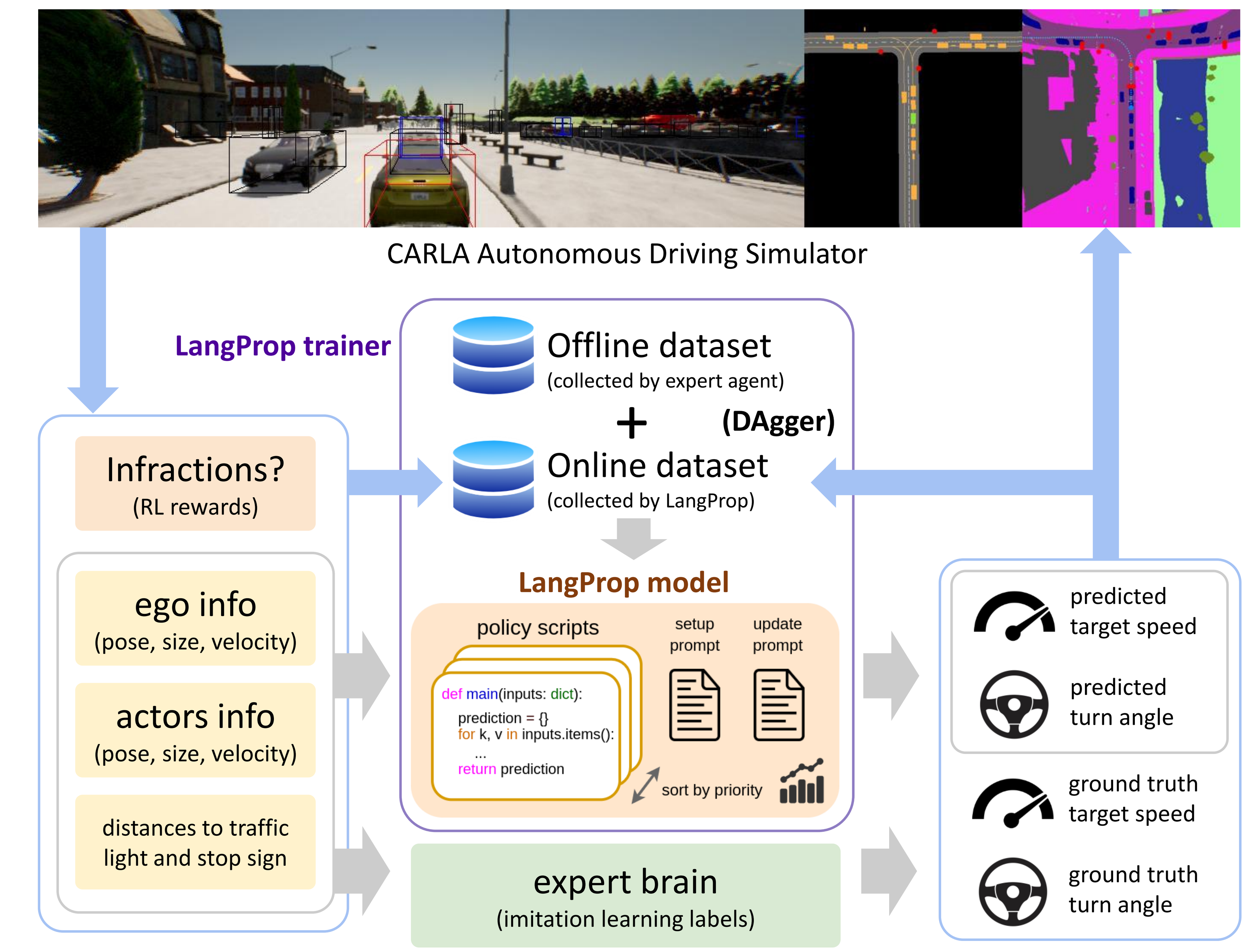
LangProp policy update mechanism



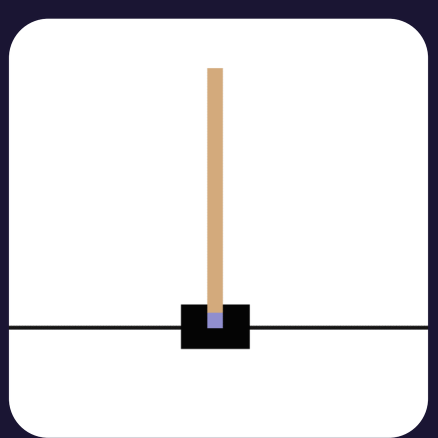
$$P_{i,k} = \frac{\left(\sum_{j=1}^{N_k^B} s_{i,j,k}\right) + W_{i,k-1}P_{i,k-1}}{N_k^B + W_{i,k-1}}, \quad W_{i,k} = \gamma\left(N_k^B + W_{i,k-1}\right).$$

N_k^B, $P_{i,k}$ and $W_{i,k}$ are the batch size, priority and priority weighting

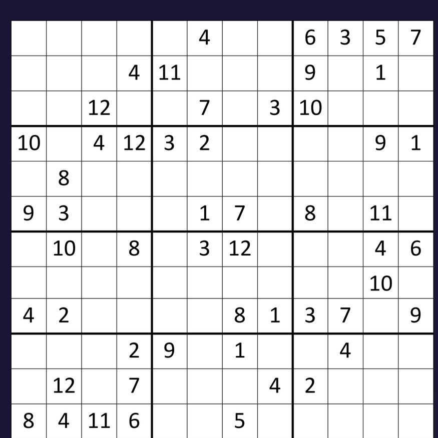
Training pipeline of the LangProp driving agent



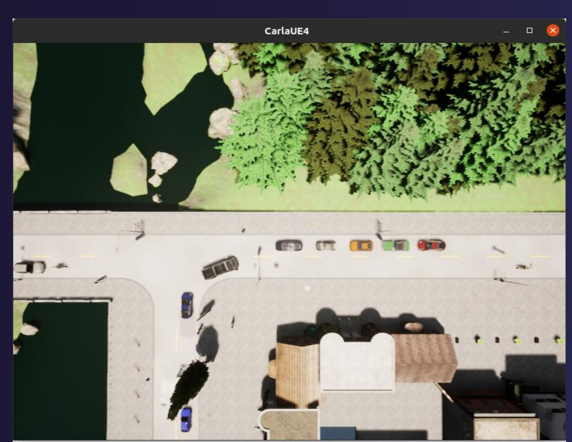
Experiments



Gym CartPole



Generalized Sudoku



CARLA autonomous driving

Results

LangProp successfully solved Sudoku and CartPole, as well as generated driving code with comparable or superior performance to human-implemented expert systems in the CARLA driving benchmark.
LangProp can generate interpretable and transparent policies that can be verified and improved, driven by metrics and data.

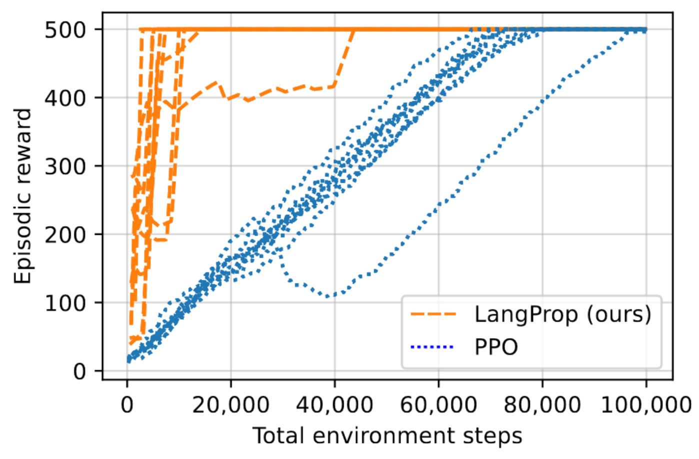


Figure 1: The total number of *environment* steps required to learn CartPole-v1 (10 seeds per method) in comparison to a RL method (PPO). Most seeds converged to an optimal solution within 10 LangProp updates.

Table 1: Driving performance of expert drivers in CARLA. The driving score is a product of the route completion percentage $\bar{R}$ and infraction factor $\bar{I}$. DAgger uses both online and offline data.

Method	Training routes			Testing routes			Longest6		
	Score ↑	$\bar{R}$ ↑	$\bar{I}$ ↑	Score ↑	$\bar{R}$ ↑	$\bar{I}$ ↑	Score ↑	$\bar{R}$ ↑	$\bar{I}$ ↑
Roach expert	57.8	95.9	0.61	63.4	98.8	0.64	54.9	81.7	0.67
TCP expert	64.3	92.3	0.71	72.9	93.2	0.77	46.9	63.1	0.76
TransFuser expert	69.8	94.5	0.74	73.1	91.3	0.80	70.8	81.2	0.88
InterFuser expert	69.6	83.1	0.86	78.6	81.7	0.97	48.0	56.0	0.89
TF++ expert	**90.8**	95.9	0.94	86.1	91.5	0.94	**76.4**	84.4	0.90
Our expert	88.9	92.8	0.95	**95.2**	98.3	0.97	72.7	78.6	0.92
LangProp: Offline IL	0.07	0.37	0.97	0.00	0.00	1.00	0.00	0.00	1.00
LangProp: DAgger IL	36.2	94.5	0.40	41.3	95.3	0.44	22.6	87.4	0.30
LangProp: DAgger IL/RL	64.2	90.0	0.72	61.2	95.2	0.64	43.7	71.1	0.65
LangProp: Online IL/RL	**70.3**	90.5	0.78	**80.9**	92.0	0.89	**55.0**	75.7	0.73