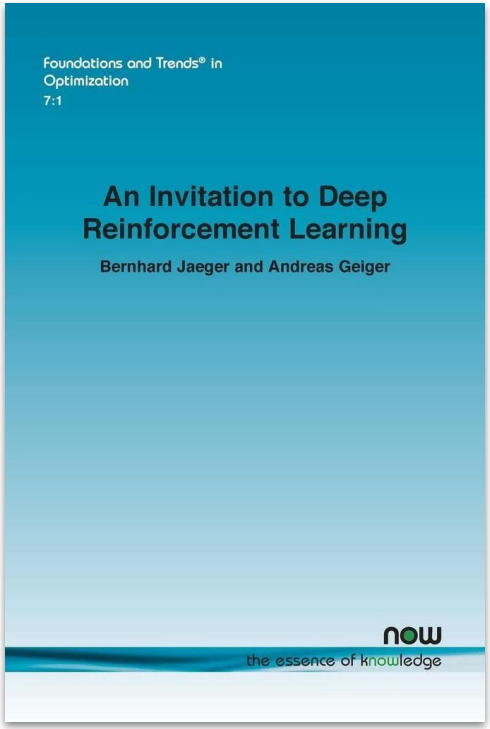
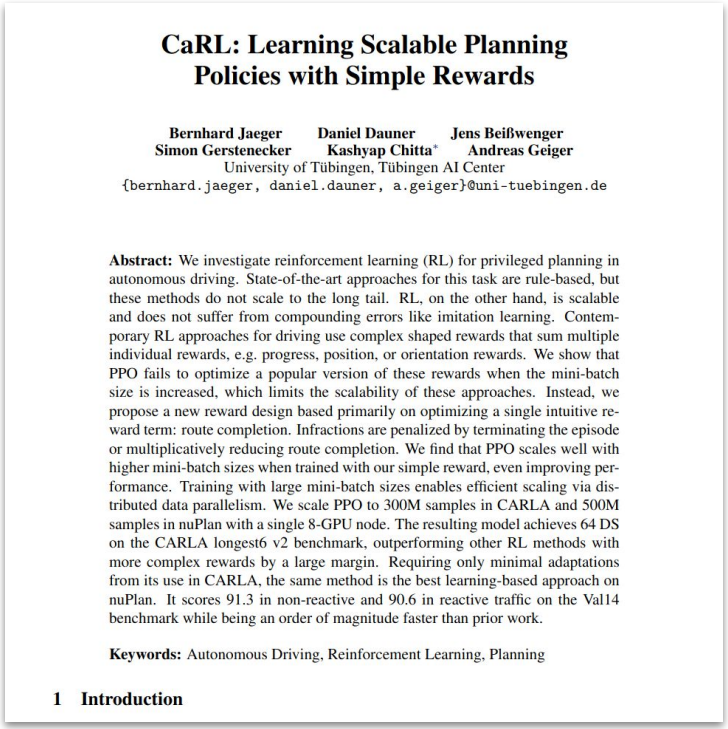
Optimizing non-differentiable metrics

And applications in autonomous driving

Plan for Today



Bernhard Jaeger, Andreas Geiger: “An Invitation to Deep Reinforcement Learning”, Foundations and Trends® in Optimization 2024



CaRL: Learning Scalable Planning Policies with Simple Rewards

Bernhard Jaeger **Daniel Dauner** **Jens Beißwenger**
Simon Gerstenecker **Kashyap Chitta*** **Andreas Geiger**
University of Tübingen, Tübingen AI Center
{bernhard.jaeger, daniel.dauner, a.geiger}@uni-tuebingen.de

Abstract: We investigate reinforcement learning (RL) for privileged planning in autonomous driving. State-of-the-art approaches for this task are rule-based, but these methods do not scale to the long tail. RL, on the other hand, is scalable and does not suffer from compounding errors like imitation learning. Contemporary RL approaches for driving use complex shaped rewards that sum multiple individual rewards, e.g. progress, position, or orientation rewards. We show that PPO fails to optimize a popular version of these rewards when the mini-batch size is increased, which limits the scalability of these approaches. Instead, we propose a new reward design based primarily on optimizing a single intuitive reward term: route completion. Infractions are penalized by terminating the episode or multiplicatively reducing route completion. We find that PPO scales well with higher mini-batch sizes when trained with our simple reward, even improving performance. Training with large mini-batch sizes enables efficient scaling via distributed data parallelism. We scale PPO to 300M samples in CARLA and 500M samples in nuPlan with a single 8-GPU node. The resulting model achieves 64 DS on the CARLA longest6 v2 benchmark, outperforming other RL methods with more complex rewards by a large margin. Requiring only minimal adaptations from its use in CARLA, the same method is the best learning-based approach on nuPlan. It scores 91.3 in non-reactive and 90.6 in reactive traffic on the Val14 benchmark while being an order of magnitude faster than prior work.

Keywords: Autonomous Driving, Reinforcement Learning, Planning

1 Introduction

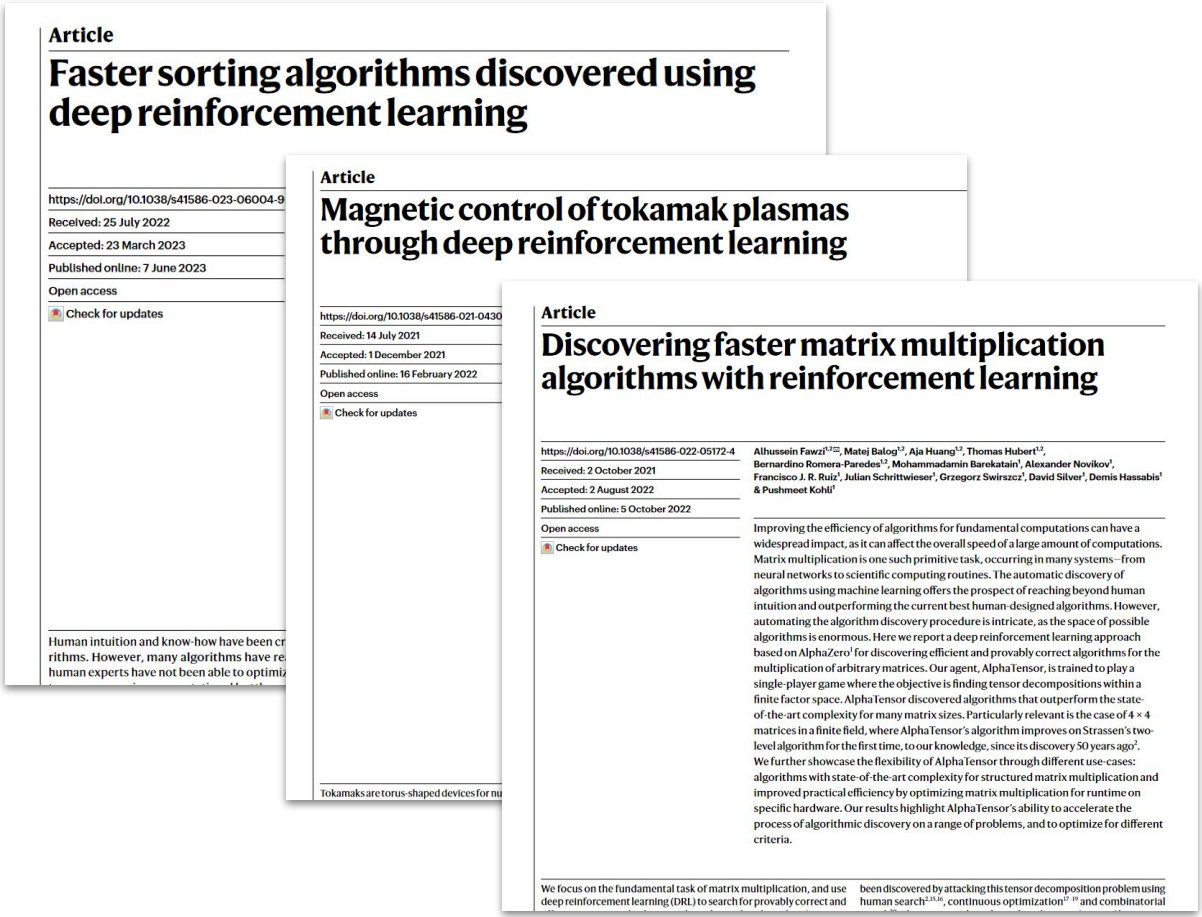
Bernhard Jaeger, Daniel Dauner, Jens Beißwenger, Simon Gerstenecker, Kashyap Chitta, Andreas Geiger: “CaRL: Learning Scalable Planning Policies with Simple Rewards”, ArXiv 2025

GPT-3

+Reinforcement Learning

GPT-3

ChatGPT



Article

Faster sorting algorithms discovered using deep reinforcement learning

https://doi.org/10.1038/s41586-023-06004-9
Received: 25 July 2022
Accepted: 23 March 2023
Published online: 7 June 2023
Open access
Check for updates

Human intuition and know-how have been cr
rithms. However, many algorithms have re
human experts have not been able to optimiz

Article

Magnetic control of tokamak plasmas through deep reinforcement learning

https://doi.org/10.1038/s41586-021-0430
Received: 14 July 2021
Accepted: 1 December 2021
Published online: 16 February 2022
Open access
Check for updates

Tokamaks are torus-shaped devices for nu

Article

Discovering faster matrix multiplication algorithms with reinforcement learning

https://doi.org/10.1038/s41586-022-05172-4
Received: 2 October 2021
Accepted: 2 August 2022
Published online: 5 October 2022
Open access
Check for updates

Alhussein Fawzi[1,2]✉, Matej Balog[1,2], Aja Huang[1,2], Thomas Hubert[1,2], Bernardino Romera-Paredes[1,2], Mohammadamin Barekatain[1], Alexander Novikov[1], Francisco J. R. Ruiz[1], Julian Schrittwieser[1], Grzegorz Swirszcz[1], David Silver[1], Demis Hassabis[1] & Pushmeet Kohli[1]

Improving the efficiency of algorithms for fundamental computations can have a widespread impact, as it can affect the overall speed of a large amount of computations. Matrix multiplication is one such primitive task, occurring in many systems—from neural networks to scientific computing routines. The automatic discovery of algorithms using machine learning offers the prospect of reaching beyond human intuition and outperforming the current best human-designed algorithms. However, automating the algorithm discovery procedure is intricate, as the space of possible algorithms is enormous. Here we report a deep reinforcement learning approach based on AlphaZero[1] for discovering efficient and provably correct algorithms for the multiplication of arbitrary matrices. Our agent, AlphaTensor, is trained to play a single-player game where the objective is finding tensor decompositions within a finite factor space. AlphaTensor discovered algorithms that outperform the state-of-the-art complexity for many matrix sizes. Particularly relevant is the case of 4 × 4 matrices in a finite field, where AlphaTensor's algorithm improves on Strassen's two-level algorithm for the first time, to our knowledge, since its discovery 50 years ago[2]. We further showcase the flexibility of AlphaTensor through different use-cases: algorithms with state-of-the-art complexity for structured matrix multiplication and improved practical efficiency by optimizing matrix multiplication for runtime on specific hardware. Our results highlight AlphaTensor's ability to accelerate the process of algorithmic discovery on a range of problems, and to optimize for different criteria.

We focus on the fundamental task of matrix multiplication, and use deep reinforcement learning (DRL) to search for provably correct and

been discovered by attacking this tensor decomposition problem using human search[2,15,16], continuous optimization[17–19] and combinatorial

MetricOpt: Learning to Optimize Black-Box Evaluation Metrics

Chen Huang Shuangfei Zhai Pengsheng Guo Josh Susskind
Apple Inc.

We study the problem o
non-differentiable task evalu
sification rate and recall. C
operates in a black-box setti
tails of the target metric are
learning a differentiable val
pact task-specific model para
The learned value function i
optimizers like SGD and Ad
finetuning a pre-trained moa
provements since the value f
ric supervision during finetu
potential bias of loss-only su
state-of-the-art performance
age) classification, image r
Solid benefits are found over
ten involve complex loss des
also generalizes well to new

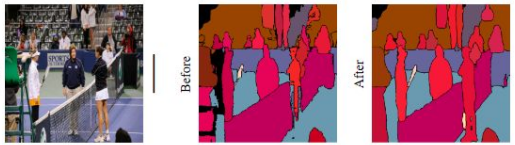
Tuning computer vision models with task rewards

André Susano Pinto [*1] **Alexander Kolesnikov** [*1]
Yuge Shi [1 2] **Lucas Beyer** [1] **Xiaohua Zhai** [1]

Abstract

Misalignment between model predictions and intended usage can be detrimental for the deployment of computer vision models. The issue is exacerbated when the task involves complex structured outputs, as it becomes harder to design procedures which address this misalignment. In natural language processing, this is often addressed using reinforcement learning techniques that align models with a task reward. We adopt this approach and show its surprising effectiveness across multiple computer vision tasks, such as object detection, panoptic segmentation, colorization and image captioning. We believe this approach has the potential to be widely useful for better aligning models with a diverse range of computer vision tasks.

(a) Optimize mAP: 39 → 54, results in a much high recall and learns box prediction confidences.



(b) Optimize PQ: 43.1 → 46.1, removes many incoherent predictions, especially for small-scale objects.

Main issue with standard supervised learning:

What we optimize ≠ what we care about

What do we care about?

"Good results"

Bold Metrics*

What is a metric* $m(x, y) \rightarrow \mathbb{R}$

What is a reward? $r(s, a) \rightarrow \mathbb{R}$

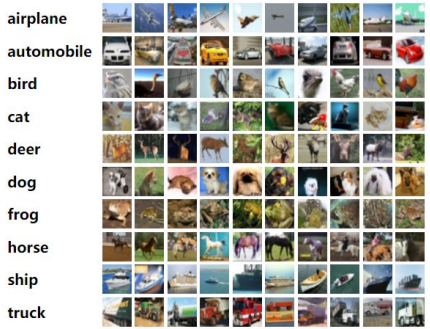
Metrics* and rewards are the same thing

Reinforcement Learning (RL) allows us to optimize what we care about

*metric in the colloquial sense. I am talking about measures

J = Performance
π = neural network
R = metric / reward func.
S = Dataset
s = state / image
a = action / label

Example: Classification



1. Write down performance measure.

$$J(\pi) = \frac{1}{|S|} \sum_{(s,a) \in \mathcal{S}} R(s,a)\pi(a|s)$$

2. Compute the gradient that maximizes performance: “Policy gradient”

$$\nabla_\pi J(\pi) = \frac{1}{|\mathcal{S}|} \sum_{(s,a) \in \mathcal{S}} R(s,a)\nabla_\pi \pi(a|s)$$

J = Performance
π = neural network
R = metric / reward func.
S = Dataset
s = state / image
a = action / label
a* = correct class /action

3. Use accuracy as reward

$$\nabla_\pi J(\pi) = \frac{1}{|\mathcal{S}|} \sum_{(s,a)\in\mathcal{S}} acc(s,a) \nabla_\pi \pi(a|s)$$

4. Simplify equation

$$\nabla_\pi J(\pi) = \frac{1}{|\mathcal{S}|} \sum_{(s,a^\star)\in\mathcal{S}} \nabla_\pi \pi(a^\star|s)$$

5. Optimize log probability instead (doesn't change global optimum)

$$L_\pi = \underbrace{-\sum_{(s,a^\star)\in\mathcal{S}} \log \pi(a^\star|s)}_{\text{Cross Entropy}} = \underbrace{-\log \prod_{(s,a^\star)\in\mathcal{S}} \pi(a^\star|s)}_{\text{Negative Log-Likelihood}}$$

Cross-entropy is the policy gradient that maximizes accuracy!

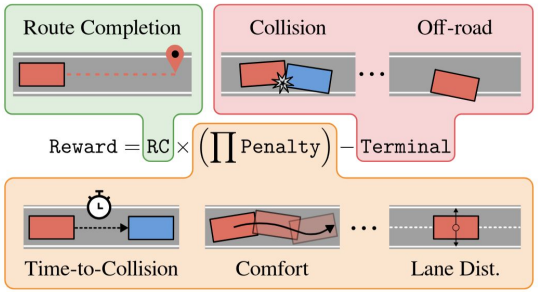
EVER WROTE A CLASSIFIER?

So how did we apply this idea in autonomous driving?

Take your driving metric and make it a reward

(3) **Driving Score (DS)**: weighted average of the route completion with infraction multiplier P_i

$$\text{DS} = \frac{1}{N}\sum_{i}^{N} R_i P_i \tag{8}$$



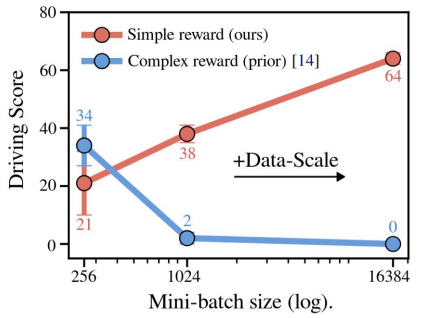
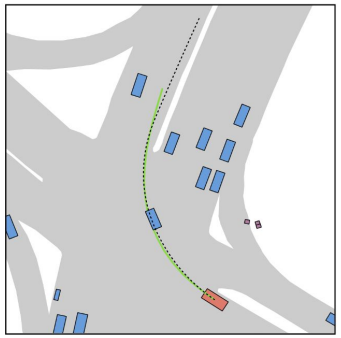


Figure 1: **Simple rewards scale with mini-batch size.** Typical rewards in driving consist of com-

Also happens to scale much better than traditional reward shaping

First time we beat Supervised Learning with RL in planning



Method	Type	Non-Reactive			Reactive			Time ↓
		CLS ↑	Col. ↑	RC ↑	CLS ↑	Col. ↑	RC ↑	
Log Replay (LQR)	*Human*	*93.5*	*98.8*	*99.0*	*80.3*	*85.6*	*99.0*	-
PlanTF [8]	IL	84.6	94.2	90.7	76.1	95.2	77.2	107
Diff. Planner [32]	IL	89.6	95.9	94.2	82.7	93.1	85.9	138
CaRL (Ours)	RL	**91.3**	**97.4**	**94.4**	**90.6**	**97.1**	**91.3**	**14**

Table 6: **Performance on Val14** (nuPlan)

Do you have an unusual metric in your problem?

Maybe you should optimize it with RL.