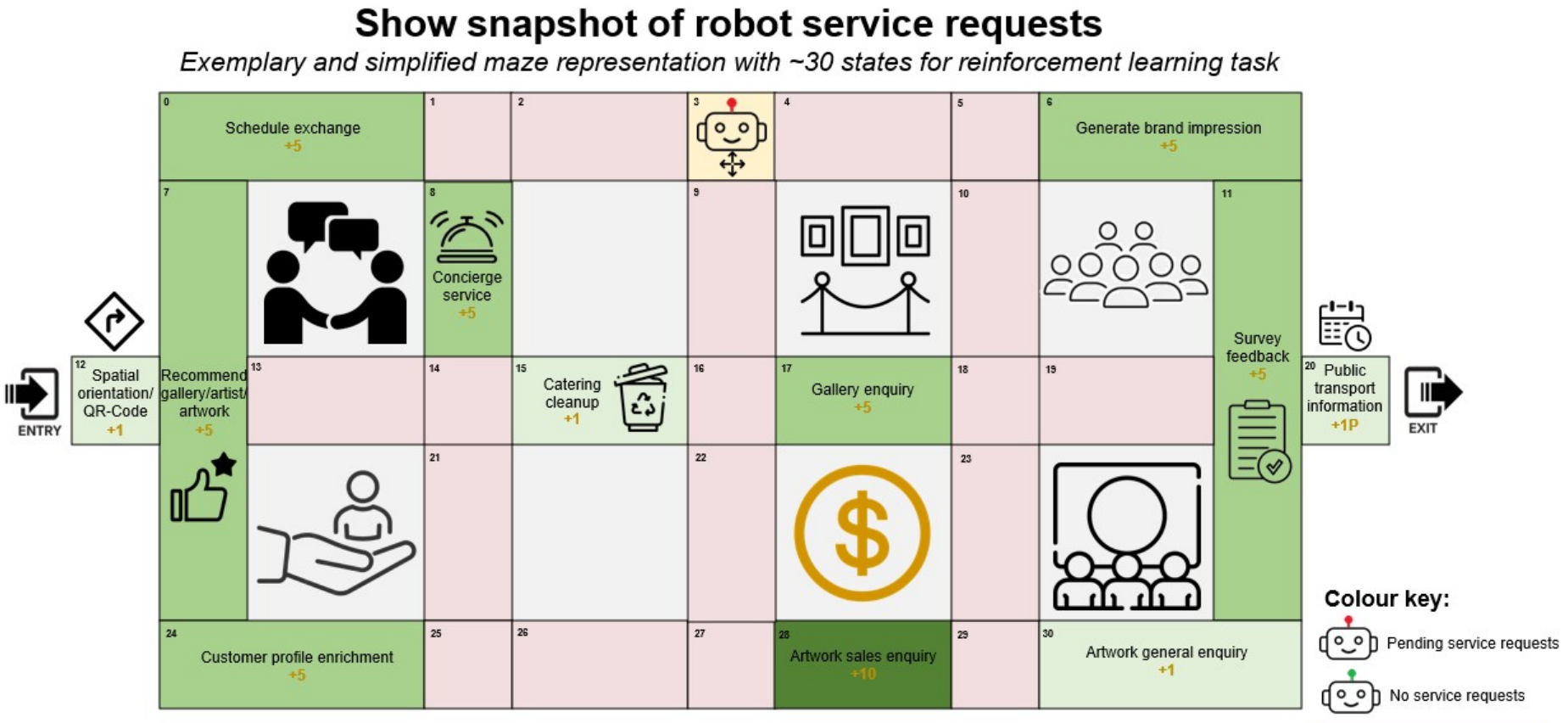
Maschinelles Lernen
- Informatik -

Rapp, DHBW Lörrach

16.02.2024

Inhaltsübersicht

Prüfungsaufgabe 6: Reinforcement Learning - Übersicht



Service examples

- Spatial orientation and provision of QR-code to view online map
- Survey feedback e.g. overall satisfaction
- Customer profile enrichment (e.g. badge scan) to add artwork to favourites collection
- Exchange scheduling between e.g. collector and gallery employee
- Gallery ad-hoc enquiry e.g. „How many high potential customers are at the show right now?“
- Concierge service e.g. gallery guidance

Prüfungsaufgabe 6: Reinforcement Learning

Entwicklung eines Dienstleistungsroboters für Kunstmessen

1. Trainieren Sie die Q-Tabelle für die in der Übersicht dargestellte Momentaufnahme mithilfe der Belohnungsmatrix (Reward_Matrix_Show_Snapshot.xlsx) und geben ein Beispiel für einen idealen Pfad an. Hinweise:
   - Ein analoges Beispiel wurde in der Vorlesung "Verstärkendes Lernenïm 5. Semester auf Folie 39 besprochen.
   - Der Zustandsraum besteht aus 31 Flächenabschnitten, die in der Belohnungsmatrix als Zeilen dargestellt werden.
   - Der Aktionsraum besteht aus der 4-elementigen Menge left, right, up, down, die in der Belohnungsmatrix als Spalten dargestellt werden.
   - Die Belohnungsmatrix ist dem Agenten unbekannt und wird von der Umgebung vorgegeben.
2. Programmieren Sie eine Python-Funktion, die für einen gegebenen Zustand innerhalb der drei Serviceanfrage-Phasen während des Messeverlaufs die optimale Strategie zurückgibt:
   - def optimal_strategy_function(current_timestep, current_phase_number, current_state_number, current_reward_matrix_dataframe)
   - current_timestep: Integer increment
   - current_phase_number in [1,2,3]
   - current_state_number in [0,...,30]
   - current_reward_matrix_dataframe als 31x31 Dataframe
   - return updated_timestep, updated_phase_number, next_state_number, updated_reward_matrix_dataframe

   Die drei aufeinanderfolgenden Serviceanfrage-Phasen sind:
   1. Es sind keine Serviceanfragen vorhanden. Hinweise:
      - Verwenden Sie die Strategie aus Aufgabe 1 (vgl. Exploitation Prinzip) in Verbindung mit einer zufälligen Komponente bei der Auswahl der nächsten Aktion (Exploration).
      - Die Dauer der Phase 1 beträgt 100 Zeitschritte (z.B. Minuten)
   2. Die zu Beginn vorhandenen Serviceanfragen (vgl. Reward_Matrix_Show_Snapshot.xlsx) werden sukzessive abgearbeitet, bis alles erledigt ist. Hinweise:
      - Nachdem eine Dienstleistung durch den Roboter erbracht wurde, wird die entsprechende Belohnung in der Belohnungsmatrix auf -1 zurückgesetzt und die Q-Tabelle aus Aufgabe 1 wird neu berechnet.
      - Die Dauer der Phase 2 hängt von der Anzahl Serviceanfragen ab.
   3. Es sind immer Serviceanfragen vorhanden und neue treffen zufällig ein:
      - Vorgehen analog zu Phase 2 durch jeweiliges Aktualisieren der Belohnungsmatrix und Neuberechnen der Q-Tabelle.
      - Die Dauer der Phase 3 beträgt 10 Zeitschritte
3. Erstellen Sie eine einfache Visualisierung der resultierenden Bewegungsabläufe des Roboters in der Maze-Darstellung für jeweils eine Beispiel-Sequenz in den drei Phasen.
4. Beurteilen Sie, inwiefern der Einsatz von Dataset Aggregation zur Effizienzsteigerung Ihres Roboters bei einer realen Kunstmesse beitragen kann.

Abschlusspräsentation u. Liefertermine auf Moodle

Abschlusspräsentation am 08.03.24

- 30 Minuten Präsentationszeit pro Gruppe

Gruppe	Präsentationszeit
C	09:00-09:30
B	09:30-10:00
D	10:00-10:30
A	10:30-11:00

Aktualisierung vom 16.2.

- entspricht ca. 5 Minuten pro Prüfungsaufgabe für Präsentation und ggf. Live-Demo
- Rückfragen durch den Dozenten während der Präsentation

Liefertermine auf Moodle

- **Fr 01.03.2024:** 2 Wahlaufgaben aus den Aufgaben 5, 6 und 7
- **Do 07.03.2024:** Folien für Abschlusspräsentation (z.B. PDF)

Imitation Learning

Anwendungen

Imitation Learning

- Lernen durch “Vormachen“

Anwendungsbeispiele

- Physikalische Mensch-Roboter Interaktion
- Autonomes Fahren
- Beantwortung von Fragen
- Maschinelle Übersetzung

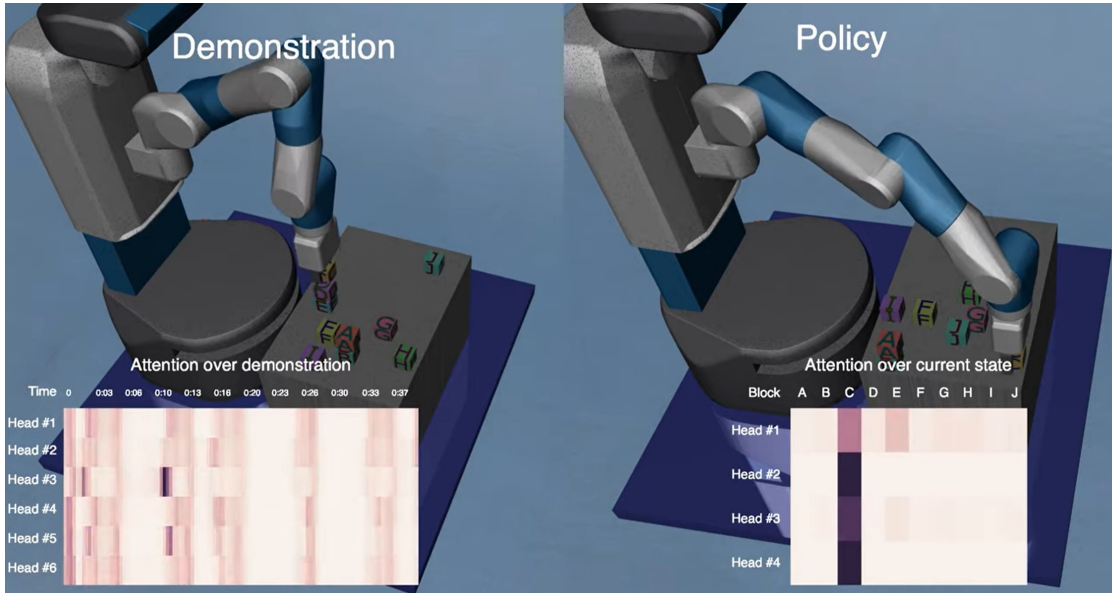
Imitation Learning in a nutshell

Gegeben: Demonstrationen oder Demonstrator
Ziel: Trainiere eine Strategie für die Nachahmung der Demonstrationen
Beispiel Super Tux Kart

Quelle: Ross, Carnegie Mellon University
https://deepai.org/publication/a-reduction-of-imitation-learning-and-structured-prediction-to-no-regret-online-learning

Beispiel Roboterarm: One Shot Imitation Learning

Aufgabe: Stapeln der Blöcke als Türme *ab, cde, fg und hij*



Duan et al., NIPS '17, *https://www.youtube.com/watch?v=oMZwkIjZzCM*

- **Links:** Demonstration
- **Rechts:** Gelernte Strategie wird auf einer neuen Situation ausgeführt
- **Unten links:** Attention Gewichte zu verschiedenen Zeitschritten (x-Achse) während der Demonstration mit Query Heads (y-Achse)
- **Unten rechts:** Attention Gewichte zu verschiedenen Blöcken im aktuellen Zustand

Wiederholung: RL Formalismus

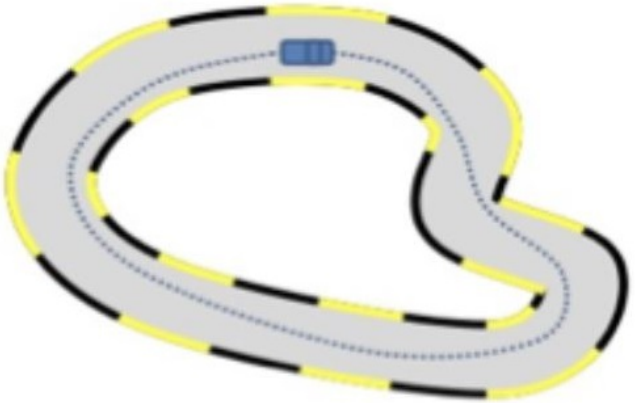
Grundlegender Formalismus im Imitation Learning entspricht dem des Reinforcement Learnings

- s : Zustand
- a : Aktion
- π_θ : Strategie ordnet Zuständen
  - Aktionen zu: $\pi_\theta(s) \rightarrow a$
  - bzw. Verteilungen von Aktionen: $\pi_\theta(s) \rightarrow P(a)$
- Erzeugung der Trajektorie $\tau := (s_0, a_0, s_1, a_1, ...)$ im *Rollout* durch sukzessive Ausführung von π auf einem initialen Zustand s_0

Beispiel 1

Rennspiel *(Super Tux Kart)*

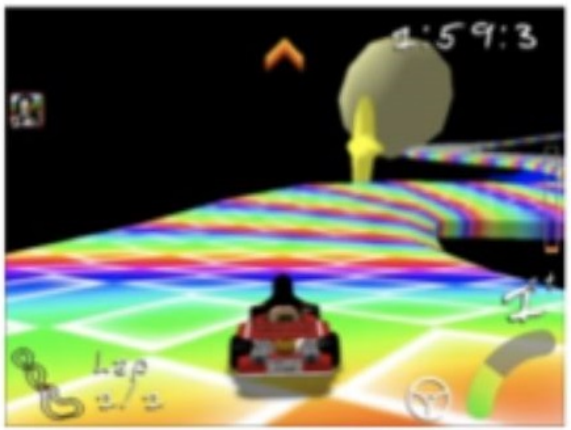
- $s =$ *Spielbildschirm*
- $a =$ *Lenkradwinkel*



Trainingsdatensatz

$$D = \{\tau := (s, a)\} \text{ von } \pi^*$$

- $s =$ Sequenz von s
- $a =$ Sequenz von a



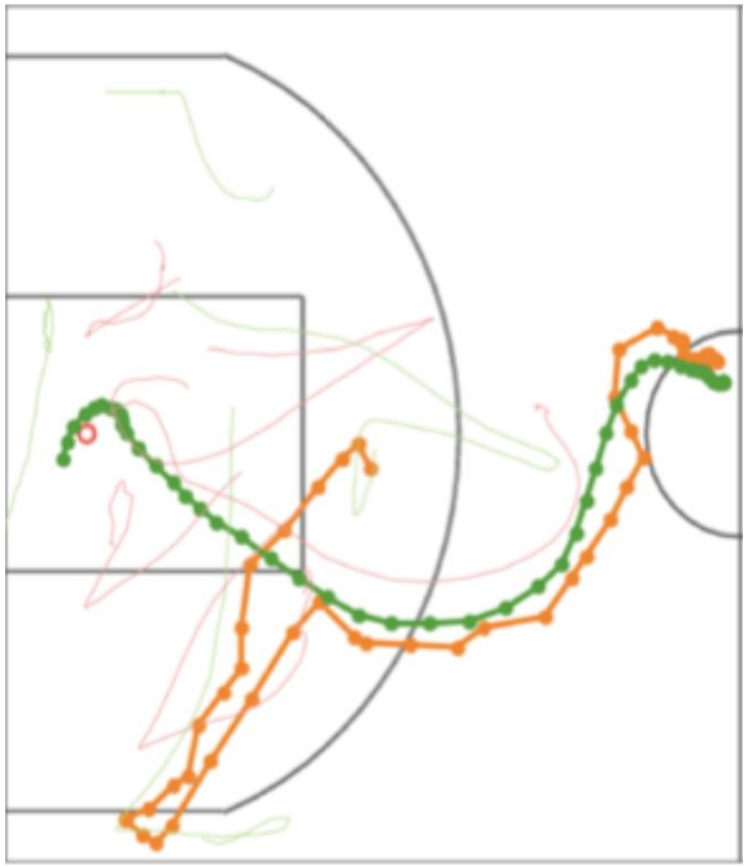
Ziel: Lernen einer (optimalen) Strategie

Lerne $\pi_\theta(s) \rightarrow a$

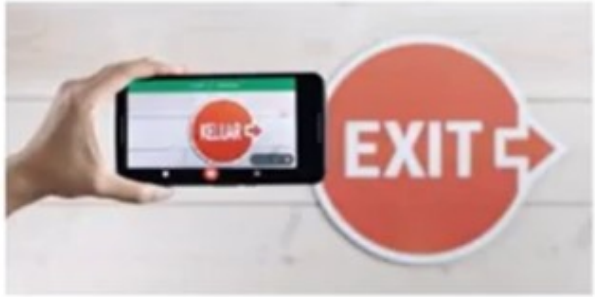
Beispiel 2

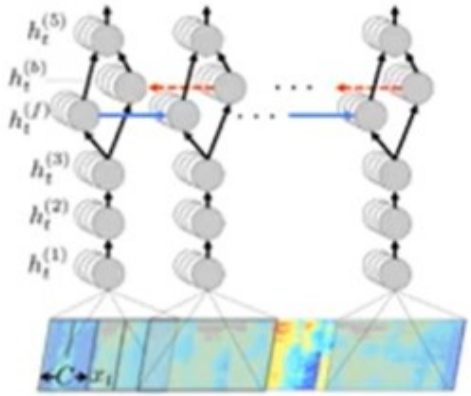
Basketball

- s = Position von Spieler und Ball
- a = nächste Position des Spielers



Bisher: Lernen, Vorhersagen zu treffen





What about learning to **control**?

Folien aus: Robotics AI Learning Lab
UC Berkeley, S. Levine
Imitation Learning Lecture (z.B. Youtube)

Von Vorhersage- zu Control-Problemstellungen

Prediction

Control

Unabhängig und identisch verteilte Zufallsvariablen

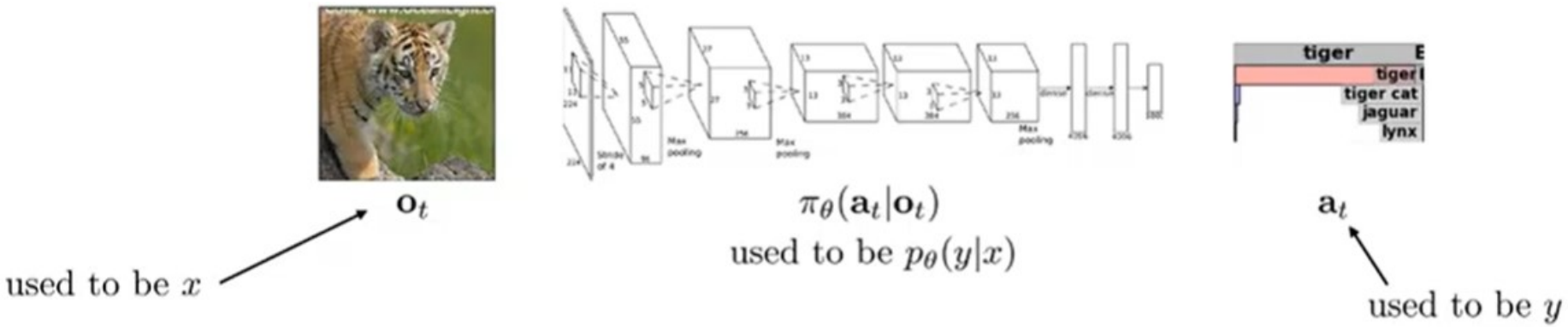
Besitzen alle dieselbe Verteilung, nehmen also mit gleicher Wahrscheinlichkeit gleiche Werte an, beeinflussen sich aber nicht (engl. **IID**).

- Annahme im **überwachten Lernen:** Zustand-Aktion-Paare sind IID
- Ground Truth Supervision
- Ziel ist die Vorhersage des richtigen Labels

ABER...

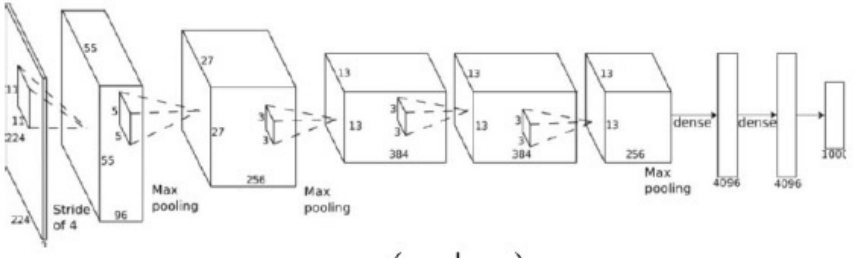
- Jede Entscheidung kann zukünftige Inputs beeinflussen (d.h. nicht unabhängig)
- Supervision kann High-Level sein (z.B. Ziel)
- Ziel ist die Erfüllung einer Aufgabe

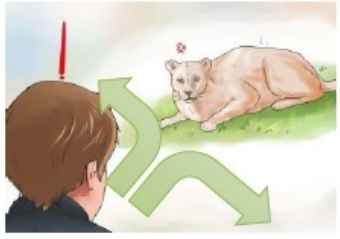
Vergleich bisherige und neue Terminologie und Notation



Bisher: ConvNet Klassifikator mit einem Bild als Input und einer Wahrscheinlichkeit für die Klassenzugehörigkeit als Output (unten). Neu: Gegeben ist eine Beobachtung und gesucht die Abbildung auf eine Aktion (oben).

Neue Terminologie und Notation





1. run away
2. ignore
3. pet

$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)

$\mathbf{o}_t$ – observation

$\mathbf{s}_t$ – state

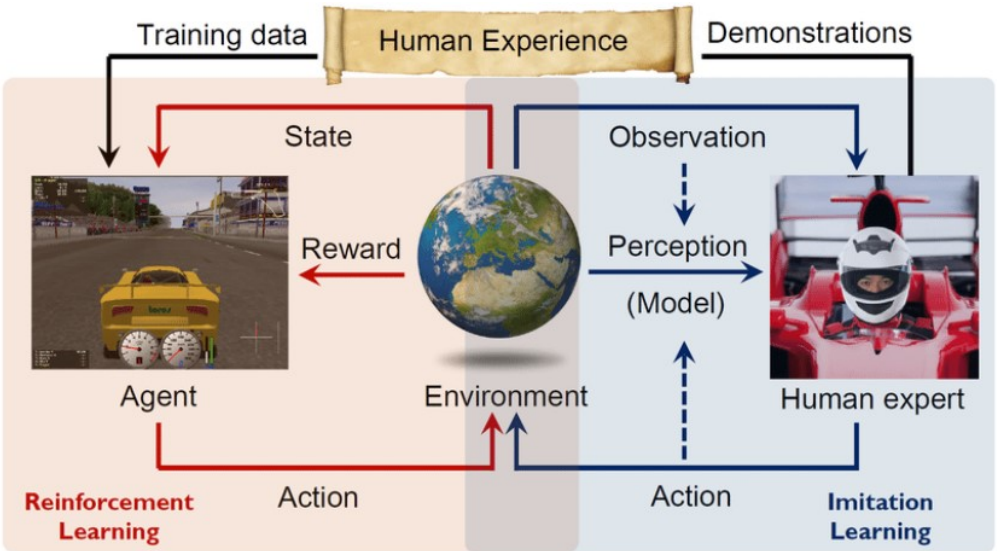
Neu: Wahrscheinlichkeit für die Klassenzugehörigkeit zu einer (z.B. diskreten) Aktion als Output. Die Strategie $\pi_\theta(a_t|o_t)$ wird durch das Modell (z.B. ConvNet) gelernt.

Imitation Learning

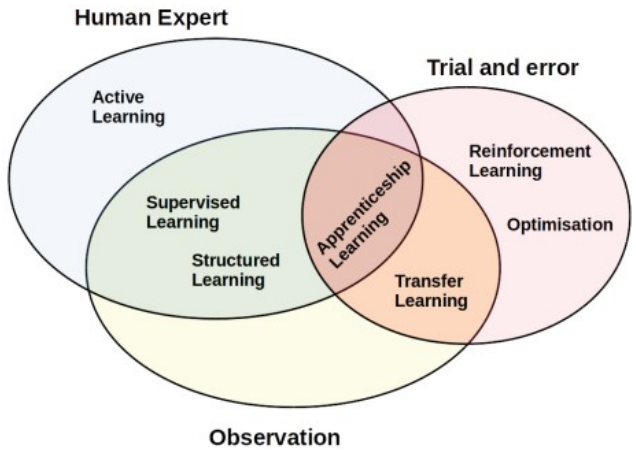
Imitation Learning und Reinforcement Learning

Imitation Learning

Der Agent lernt die Strategie *(Policy)* durch Nachahmung der Aktionen des menschlichen Demonstrators.



Continuous Reinforcement Learning From Human Demonstrations With Integrated Experience Replay for Autonomous Driving, Zuo et al., 2017



An agent can learn from **teacher demonstrations, observing other agent's actions** or through **trial and error.** *(Aus: Learning methods from different sources, Hussein et. al, Imitation Learning: A Survey of Learning Methods, 2016)*

Imitation Learning wird gegenüber Reinforcement Learning bevorzugt, wenn es für den menschlichen Experten z.B. einfacher ist, das gewünschte Verhalten zu demonstrieren, als eine Belohnungsfunktion zu definieren bzw. die Strategie zu erlernen.

Die Demonstrationen menschlicher Experten werden häufig für das Vortrainieren eines Modells für Reinforcement verwendet, um z.B. eine Belohnungsfunktion bzw. Strategie zu formen.

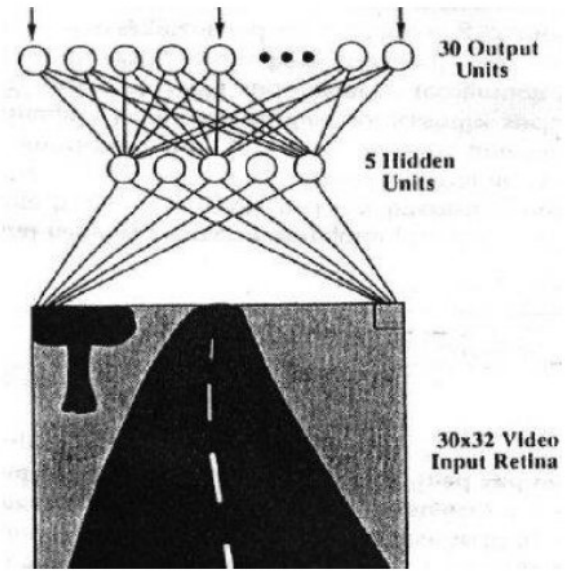
Anwendungen

- Many successes:
  - Legged locomotion [Ratliff 06]
  - Outdoor navigation [Silver 08]
  - Helicopter flight [Abbeel 07]
  - Car driving [Pomerleau 89]
  - etc...

Aus: Adaptive Control and Reinforcement Learning
Georgia Institute of Technology, B. Boots

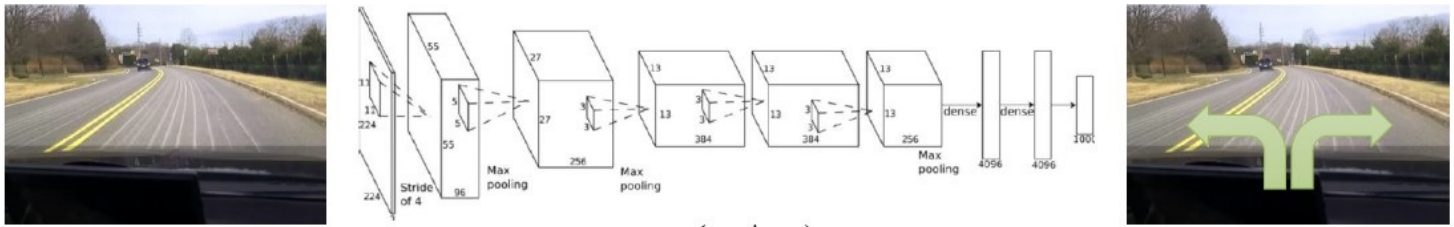
Ursprüngliches Deep Imitation Learning System

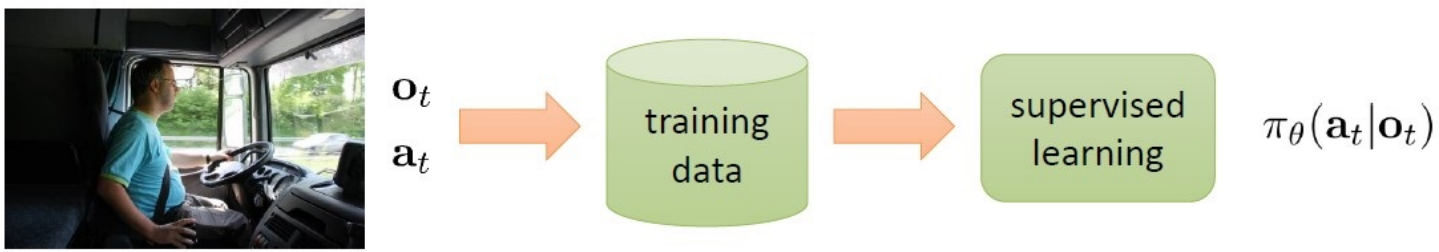
ALVINN: **A**utonomous **L**and **V**ehicle **I**n a **N**eural **N**etwork
1989



„When driving for itself, the network may occasionally stray from the road center, so it must be prepared to recover by steering the vehicle back to the center of the road.“ — Pomerleau, 1989

Imitation Learning





Bojarski et al. '16, NVIDIA

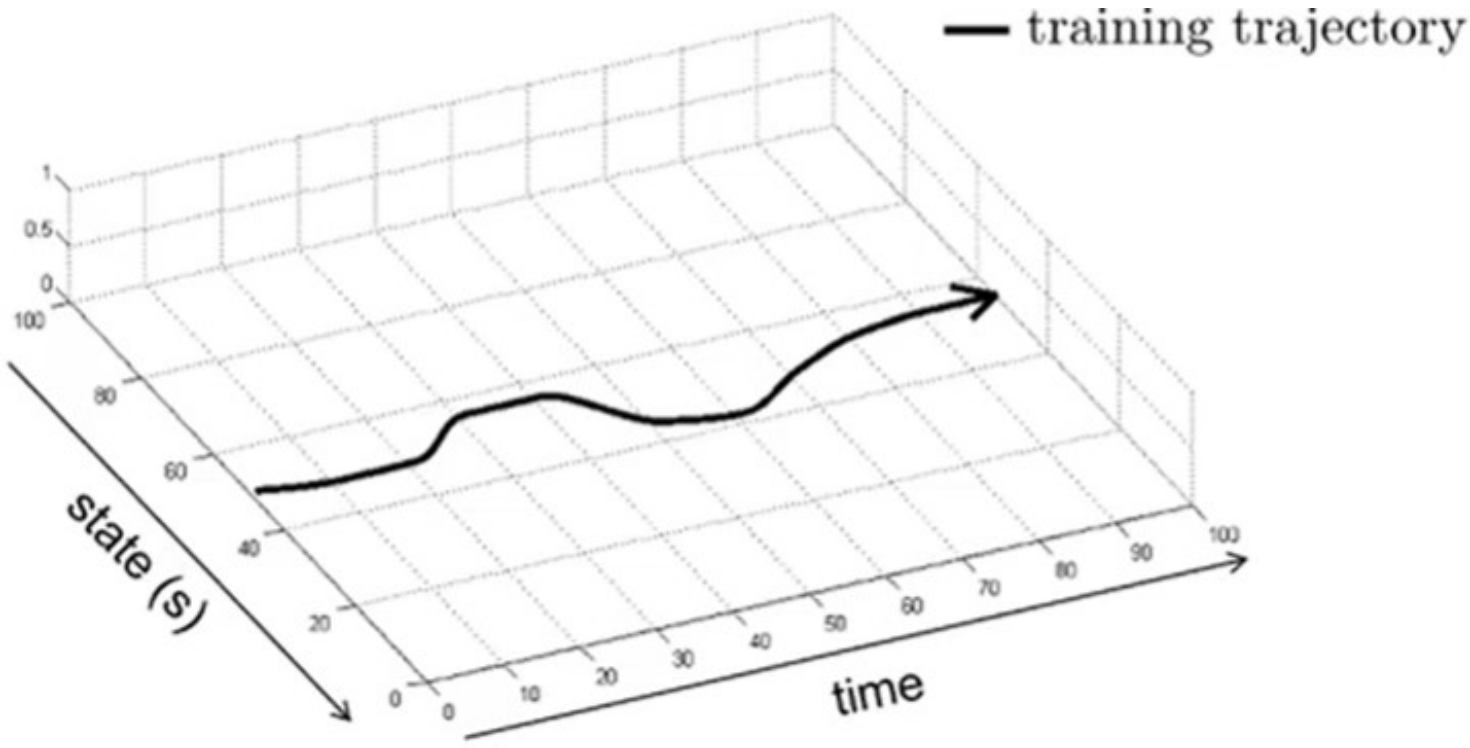
Behavior Cloning *(einfachste Form von Imitation Learning)*

- **Beobachtungen:** Kamerabilder
- **Aktionen:** Links oder rechts steuern
- **Annahme:** Gelabelte Trainingsdaten stehen als Ground Truth zur Verfügung
- **Ziel:** Trainiere das (z.B. Conv-)Net im überwachten Lernen so, dass ein Bild als Input eine Aktion als Output liefert

Distributional Shift

Funktioniert es in der Theorie?

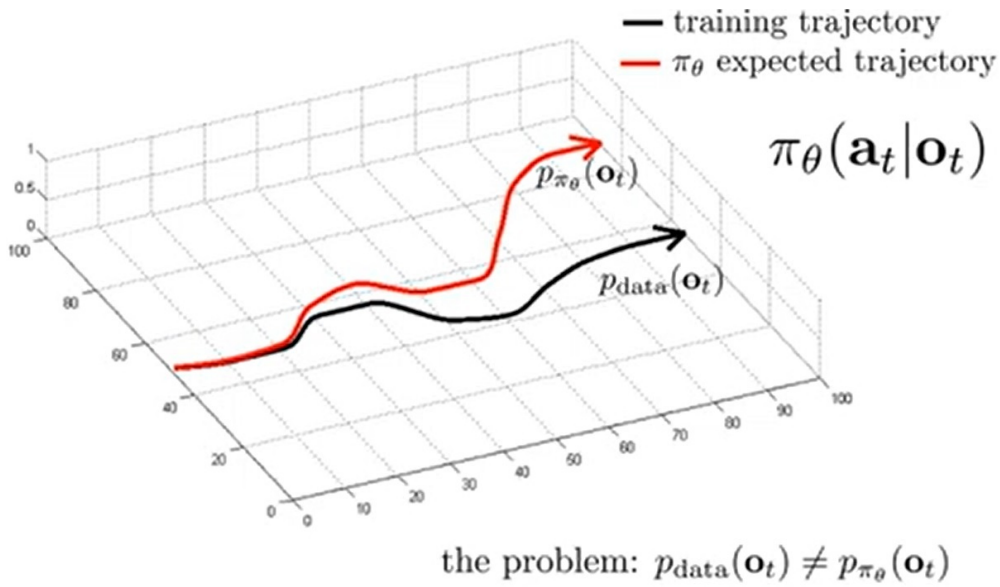
Nein!



Wahrscheinlichkeitsverteilung der Beobachtungen $p_{data}(o_t)$ *(z-Achse)* aus den **Trainingsdaten,** dargestellt als Trajektorie am Beispiel eines eindimensionalen Zustands *(y-Achse)* über die Zeit *(x-Achse)*.

Distributional Shift

Nein!



The problem: this is a training/test discrepancy: the network always saw **true** sequences as inputs, but at test-time it gets as input its own (potentially incorrect) predictions

This is called **distributional shift**, because the input distribution **shifts** from true strings (at training) to synthetic strings (at test time)

Distributional Shift

Die Input Verteilung driftet von der wahren Verteilung aus den Trainingsdaten $p_{data}(o_t)$ hin zu synthetischen Daten zur Testzeit $p_{\pi_\theta}(o_t)$ ab.

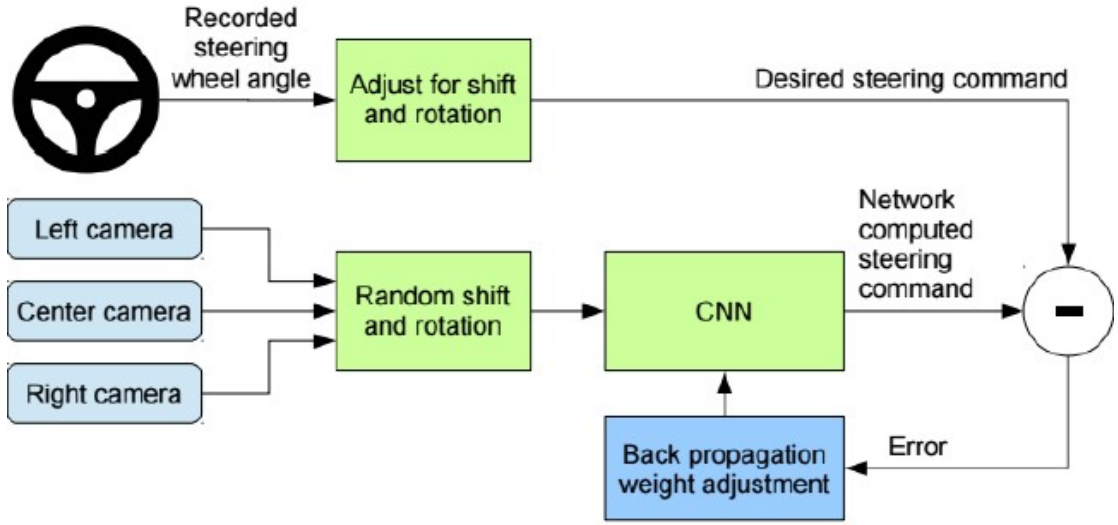
$\Rightarrow$ Erwartungswert des Fehlers steigt **quadratisch(!)** mit den Zeitschritten *(S. Ross, 2010)*

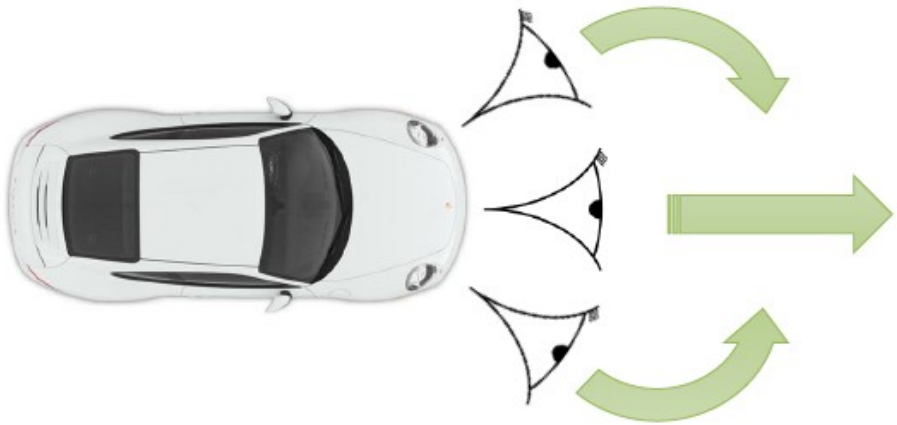
Funktioniert es in der Praxis?

Ja!

Bojarski et al. '16, NVIDIA, https://www.youtube.com/watch?v=qhUvQiKec2U

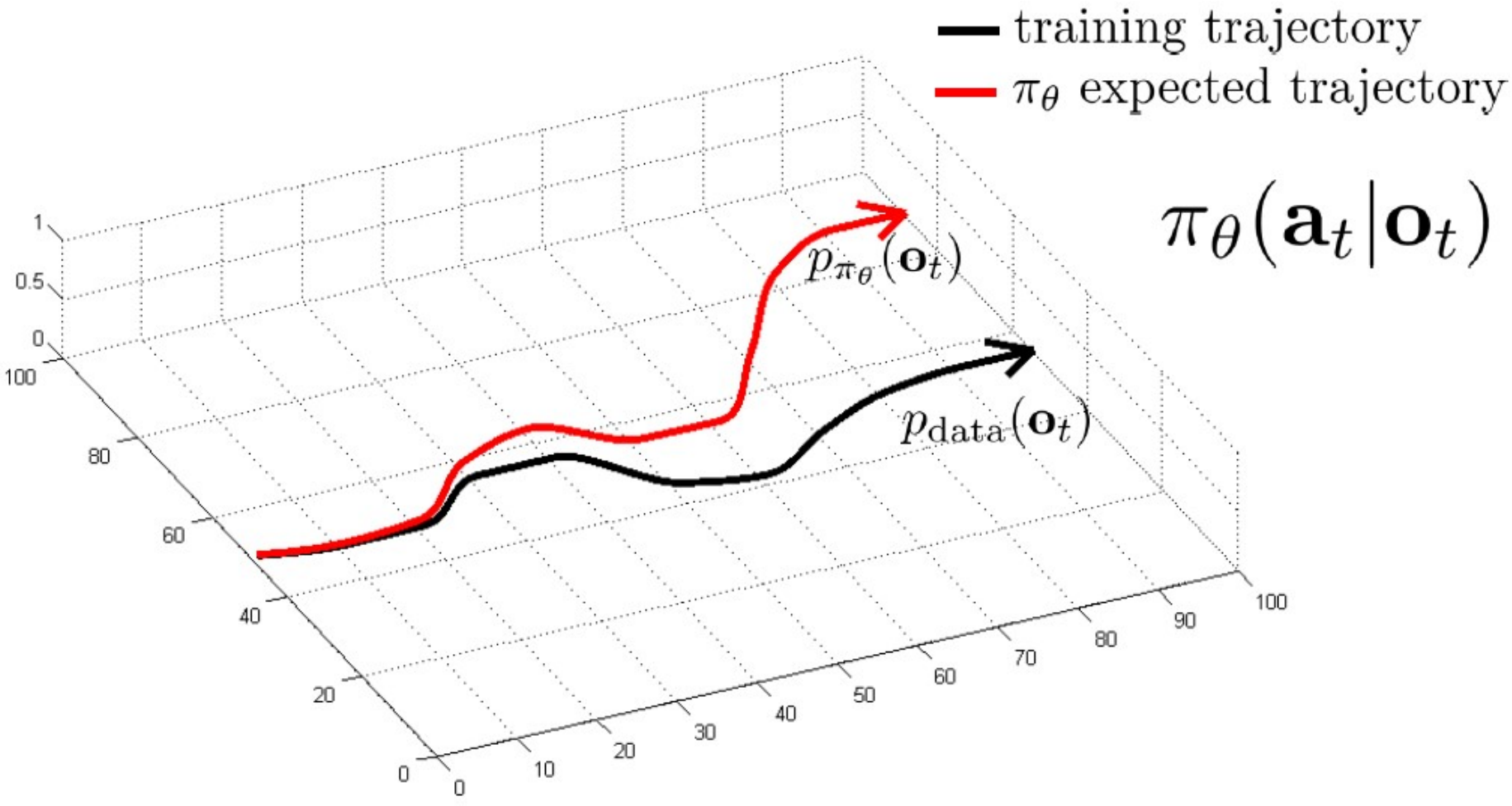
Warum funktioniert es in der Praxis?





„Hack“: Distributional Shift wird durch geeignete Anordnung der 3 Kameras (links, geradeaus und rechts) reduziert.
Bojarski et al. ’16, NVIDIA

Können wir die Stabilität erhöhen?



can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

Dataset Aggregation

DAgger

Wie können wir erreichen, dass $p_{data}(o_t) = p_{\pi_\theta}(o_t)$?
Idee: Anstelle $p_{\pi_\theta}(o_t)$ zu verbessern, verbessern wir $p_{data}(o_t)$!

DAgger: Dataset Aggregation

- **Ziel:** Einholen von Trainingsdaten von $p_{\pi_\theta}(o_t)$ anstelle von $p_{data}(o_t)$
- **Vorgehen:** Ausführen von $\pi_\theta(a_t|o_t)$ zur Trainingszeit
- **Voraussetzung:** Labels *(=vom Demonstrator erstellte Ground Truth)* für die ausgeführten Aktionen a_t zur Trainingszeit!

Algorithmus *(Ross et al. '11)*

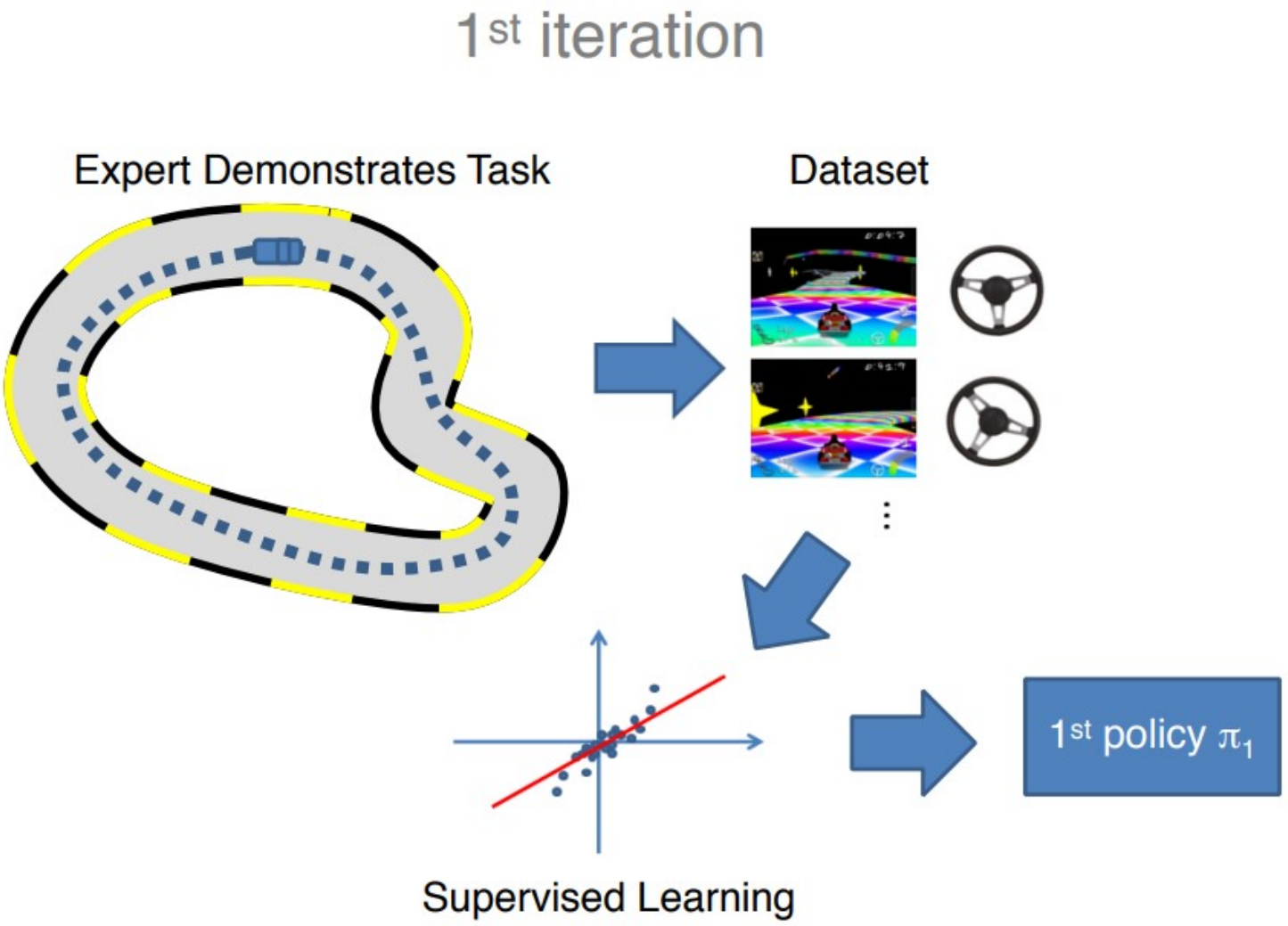
1. Trainiere $\pi_\theta(a_t|o_t)$ auf den durch menschlichen Demonstrator gelabelten Daten $\mathcal{D} = \{o_1, a_1, ..., o_N, a_N\}$ *($\Rightarrow$ Distributional Shift)*
2. Führe $\pi_\theta(a_t|o_t)$ aus, um den Datensatz $\mathcal{D}_\pi = \{o_1, ..., o_M\}$ zu erhalten
3. Demonstrator labelt $\mathcal{D}_\pi$ mit entsprechenden Aktionen a_t
4. Aggregiere $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$
5. Gehe zu 1.

$\Rightarrow$ $p_{data}(o_t)$ nähert sich mit zunehmenden Iterationen $p_{\pi_\theta}(o_t)$ an.
$\Rightarrow$ Distributional Shift wird reduziert!

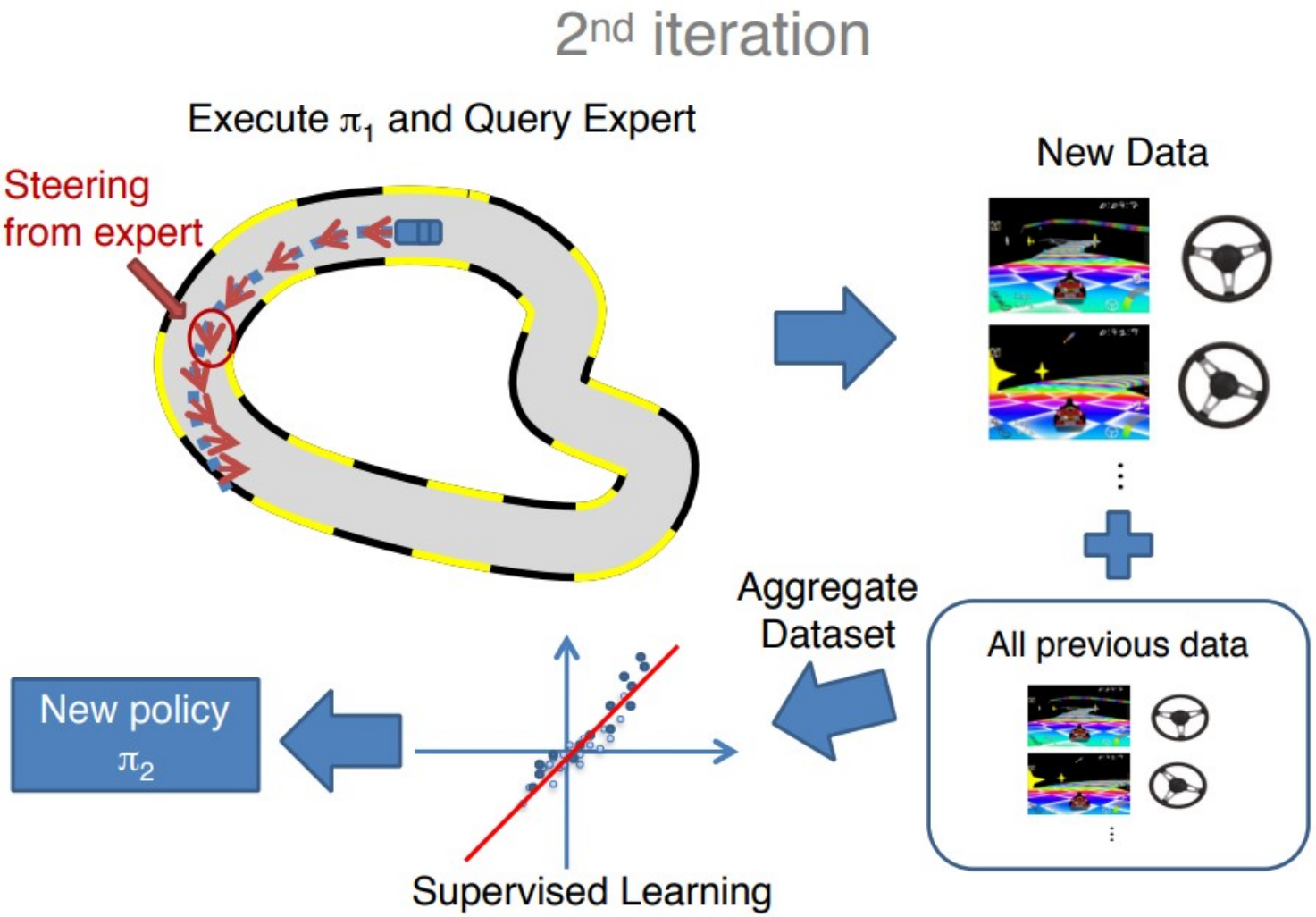
Herausforderungen

- Ausführung einer unsicheren bzw. nur teilweise trainierten Policy
- wiederholtes Einholen des Expertenfeedbacks notwendig

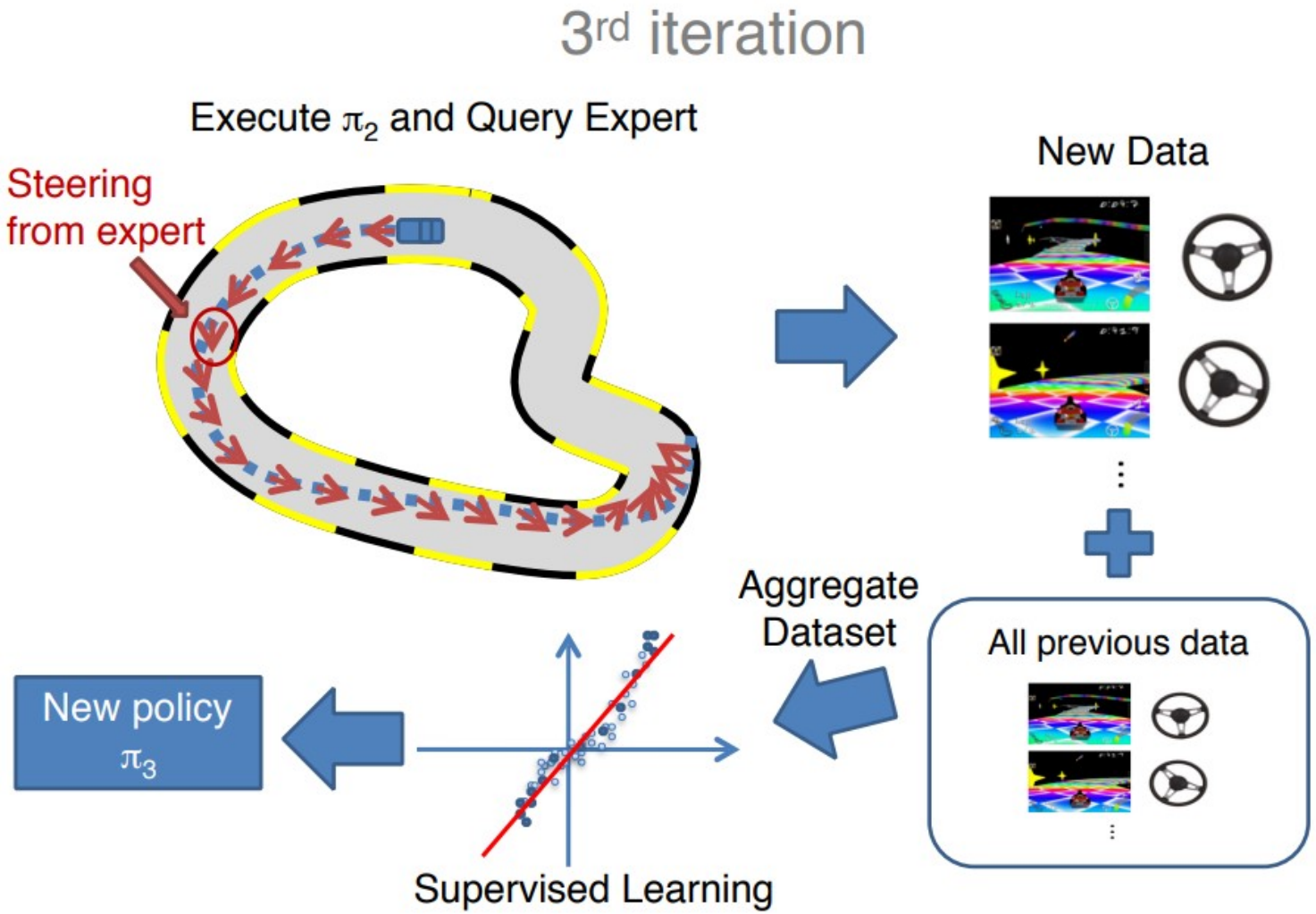
DAgger 1. Iteration



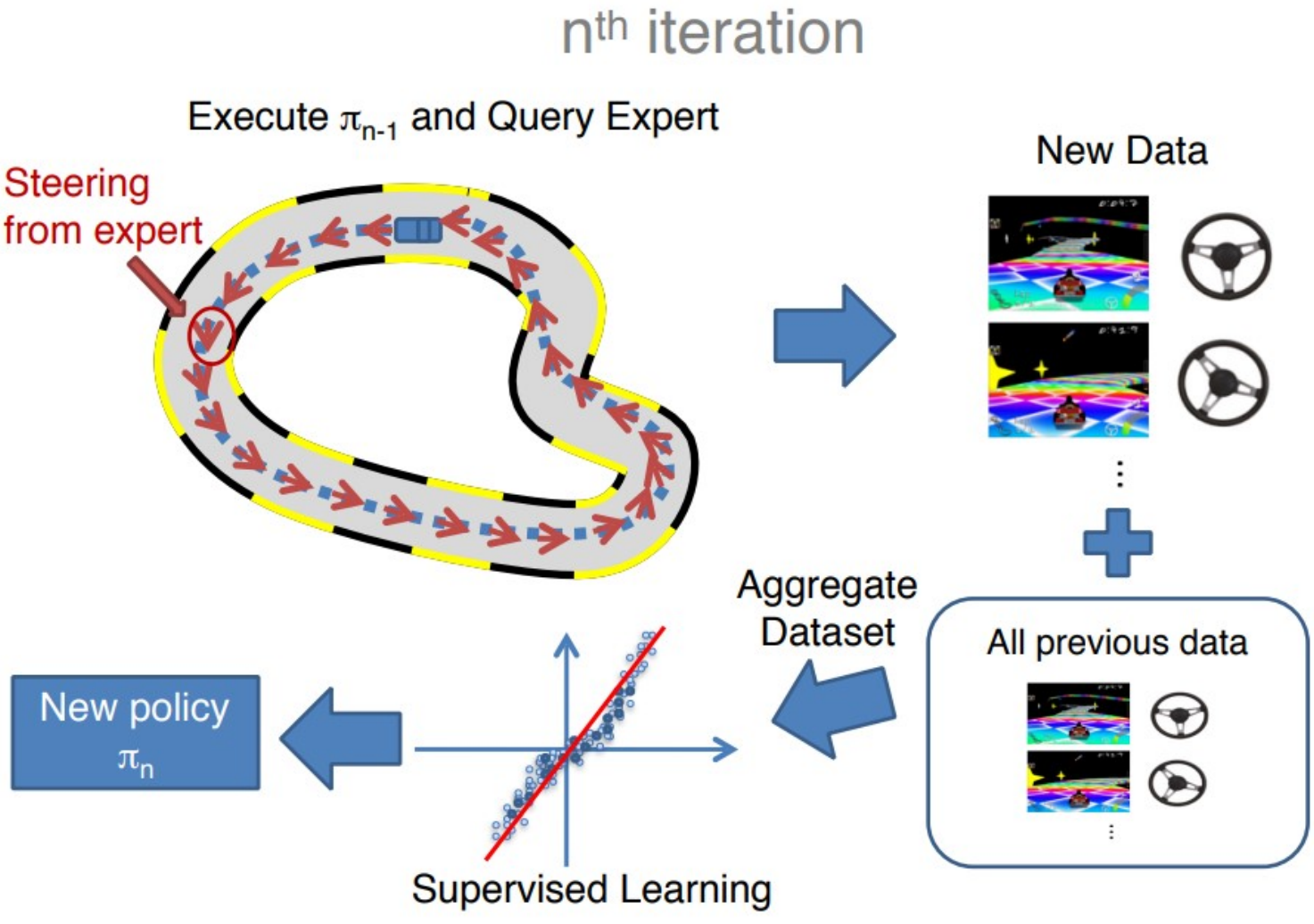
DAgger 2. Iteration



DAgger 2. Iteration



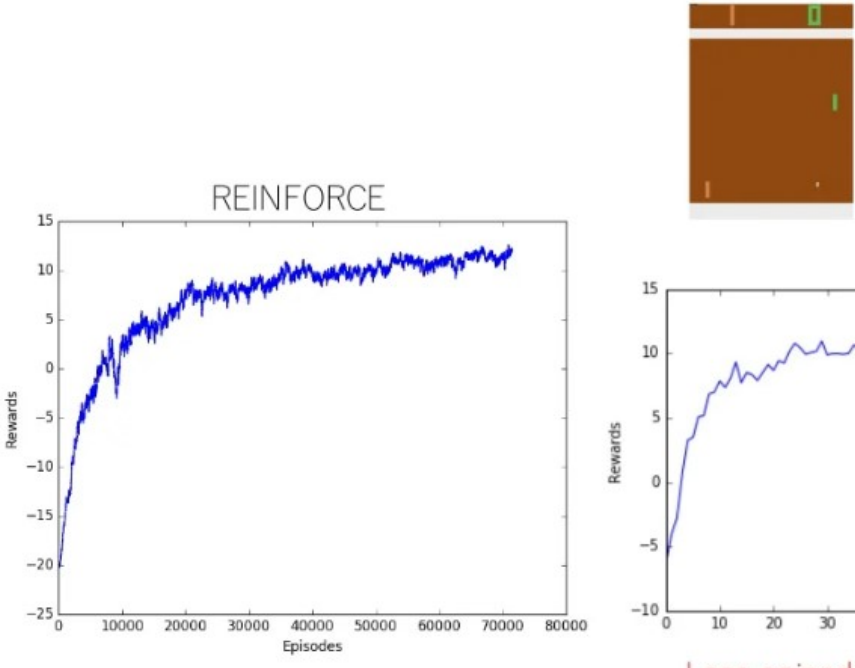
DAgger n-te Iteration

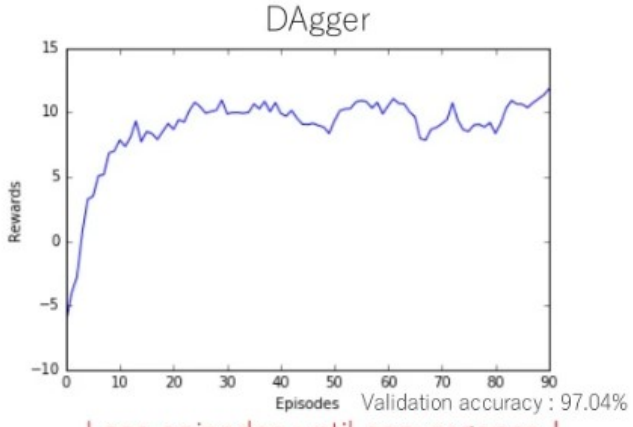


DAgger: Erwartungswert des (konvexen) Fehlers steigt linear mit den Zeitschritten.

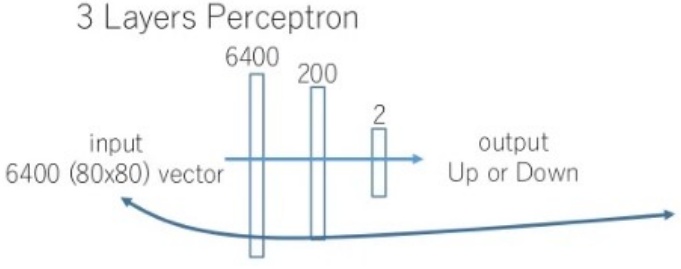
DAgger Beispiel: Pong in OpenAI Gym

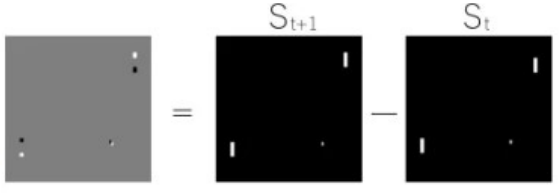
Vergleich Reinforcement Learning vs. DAgger





Less episodes until convergence !



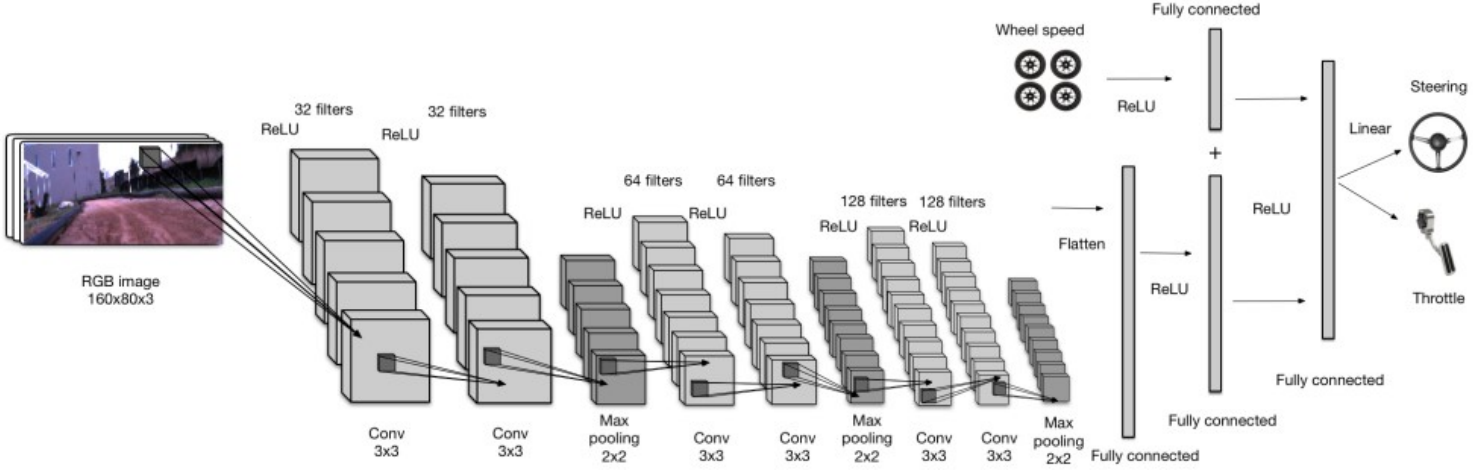


Zustand: 80x80, Belohnung: Gewinnen +1, Verlieren -1

Aus: Imitation Learning for Autonomous Driving in TORCS
Preferred Networks, Y. Kudo

DAgger Beispiel: High-Speed Autorennen

Terrestrial Agility: Drive Faster Than Human Pilots, Don't Crash!

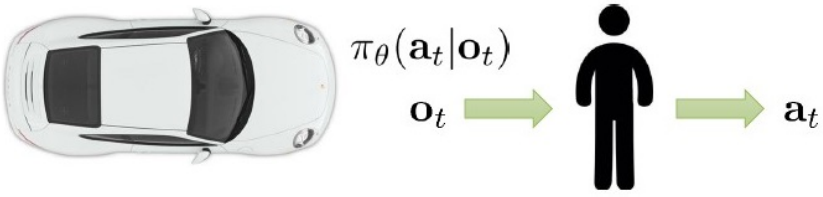


Aus: Adaptive Control and Reinforcement Learning
Georgia Institute of Technology, B. Boots

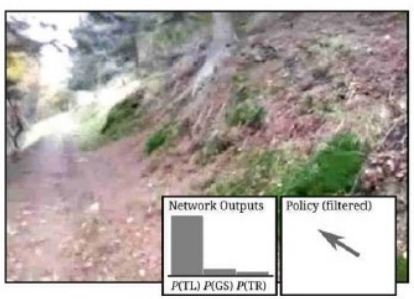
Probleme mit der Modellierung des Experten

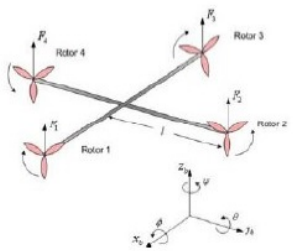
Imitation Learning: Was sind die Probleme?

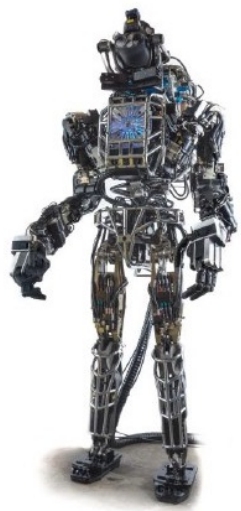
1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$



- Humans need to provide data, which is typically finite
  - Deep learning works best when data is plentiful
- Humans are not good at providing some kinds of actions





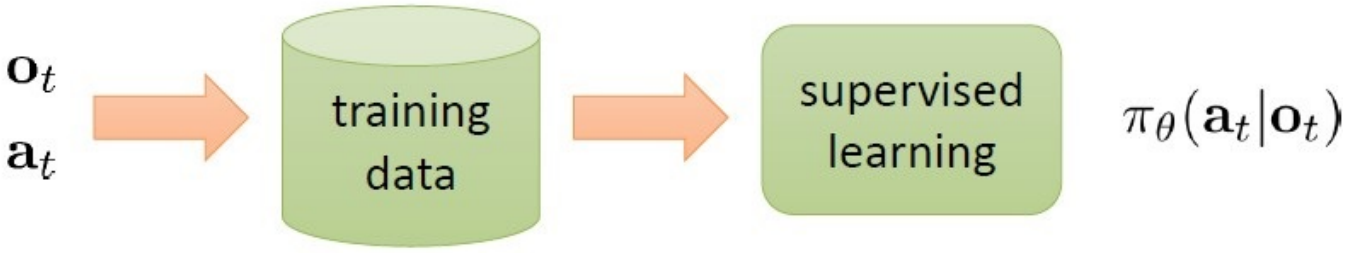


- Humans can learn autonomously; can our machines do the same?
  - Unlimited data from own experience
  - Continuous self-improvement

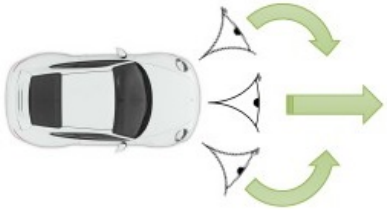
Zusammenfassung

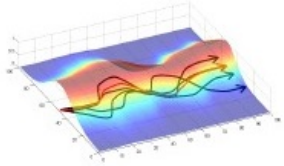
Zusammenfassung

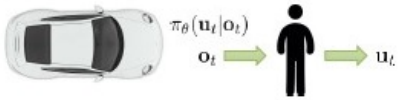
Imitation Learning



- Als alleinstehender Ansatz oftmals unzureichend
  - Distributional Shift Problem
- Funktioniert besser durch z.B.
  - Hacks (z.B. links/rechts Bilder beim autonomen Fahren)
  - Sampling einer stabilen Verteilung einer Trajektorie
  - Hinzufügen von On-Policy Daten z.B. DAgger
  - Wahl eines geeigneten Modells



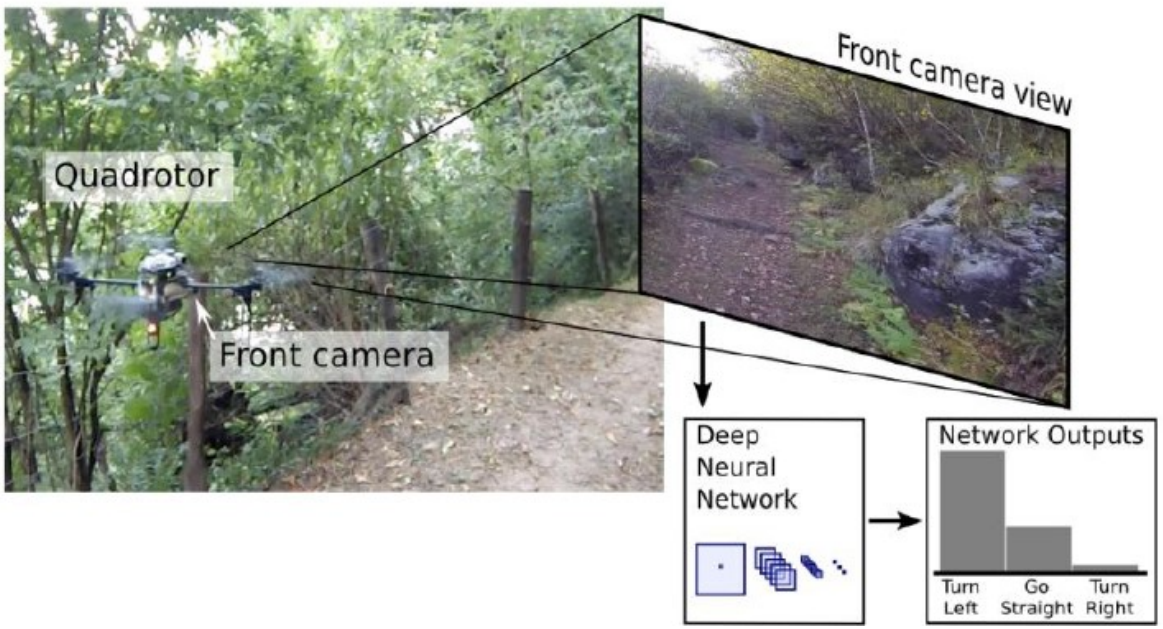




Anwendungsbeispiel: Pfadverfolgung als Klassifikationsproblem

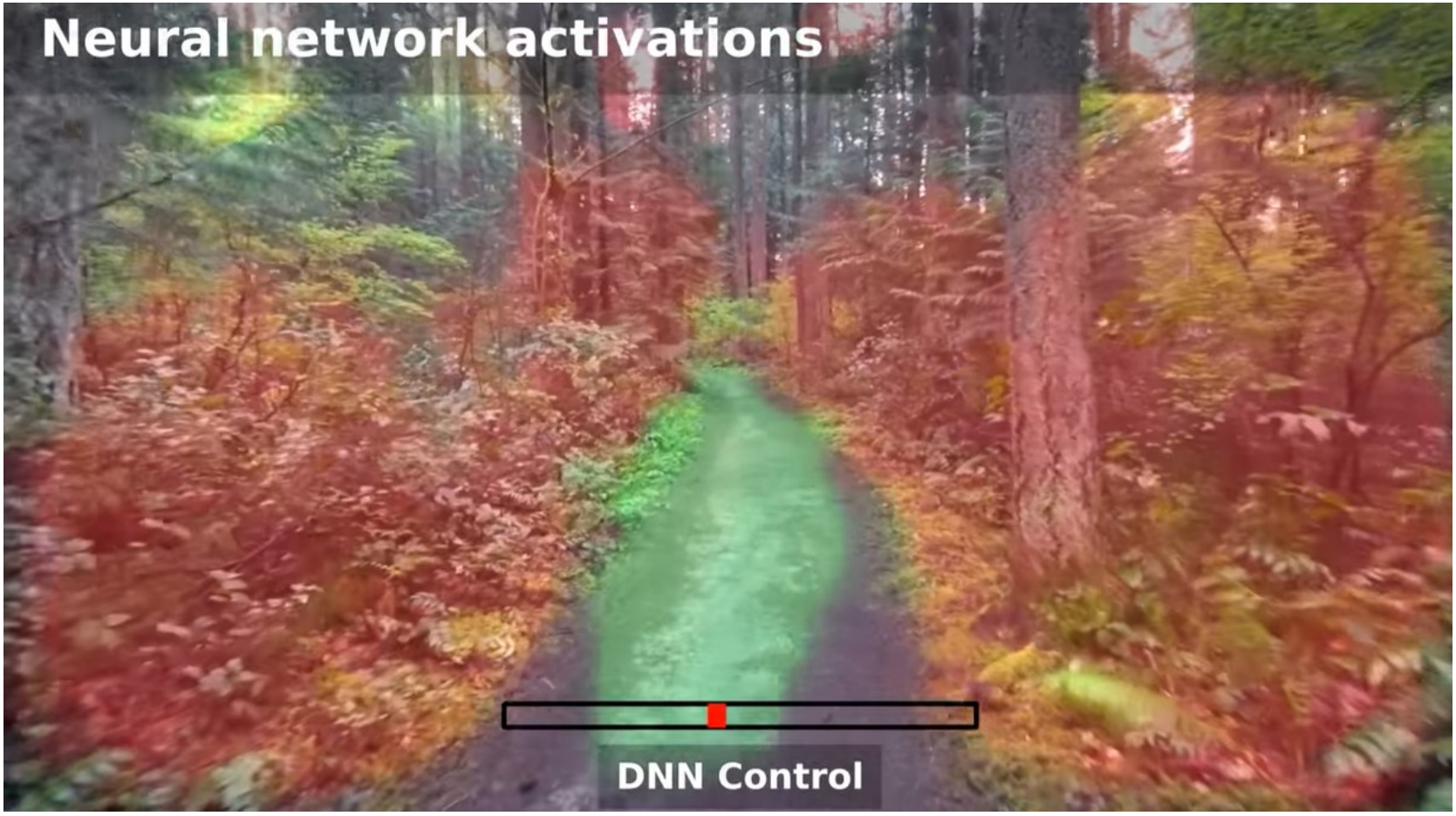
A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots

Alessandro Giusti[1], Jérôme Guzzi[1], Dan C. Cireşan[1], Fang-Lin He[1], Juan P. Rodríguez[1]
Flavio Fontana[2], Matthias Faessler[2], Christian Forster[2]
Jürgen Schmidhuber[1], Gianni Di Caro[1], Davide Scaramuzza[2], Luca M. Gambardella[1]



Basis: Daten eines menschlichen Demonstrators

Drohnenflug entlang 250 Meter Waldpfad



Autonomous Drone Navigation with Deep Learning. Flight over 250 meter Forest Trail
Link: https://www.youtube.com/watch?v=H7Ym3DMSGms

Erinnerung: 2-minütige Präsentationsübung am 23.2.

Präsentieren Sie während 2 Minuten Ihr KI/ML-Lieblingsthema!