Learning Robust Driving Policies

Andreas Geiger

Autonomous Vision Group
University of Tübingen / MPI for Intelligent Systems Tübingen

August 23, 2020
Collaborators

Eshed Ohn-Bar      Aditya Prakash

Kashyap Chitta    Aseem Behl    Andreas Geiger
Motivation: Hard coding policies is difficult ⇒ follow data-driven approach!

- **Given:** demonstrations or demonstrator
- **Goal:** train a policy to mimic decision
Conditional Imitation Learning

**Advantages:**
- End-to-End Trainable
- Cheap Annotations

**Limitations:**
- Generalization
- High Sample Complexity
- Covariate Shift
- Interpretability
How can we learn to drive under the vast diversity of all visual, planning and control scenarios?
Situational Driving
Inspiration: World Models

- Step 1: Learn **generative model** of game environments (VAE)
- Step 2: Learn dynamics model and control model in **latent space** (CMA-ES)
- Not sufficient ⇒ we combine this idea with **imitation learning**

Learning Situational Driving

- **Step 1:** Learn a **mixture of expert policies** $\{\alpha^k, \pi^k_\theta\}$ via imitation (LSD)
- **Step 2:** Learn a **general purpose context embedding** $q_\phi$ as a $\beta$-VAE
- **Step 3:** Perform **task-driven policy refinement** by interacting with the simulation and maximizing a driving task reward (LSD+)

Learning Situational Driving

\[ \pi_{\Theta}(a|o, c) = \sum_{k=1}^{K} \alpha^k_{\Theta}(o, c) \pi^k_{\Theta}(a|o, c) + \Psi \]

where:

- \( \pi_k(a|o, c) = \mathcal{N}\left(a | \mu_k(o, c), \text{diag}(\sigma_k(o, c))^2 \right) \)

Observations: \( o = [I, v] \in \mathcal{O} \)
Command: \( c \in C = \{\text{left, right, straight, follow}\} \)
Actions: \( a \in A = [-1, 1]^2 \)
Learning Situational Driving

\[ \pi_\Theta(a|o, c) = \sum_{k=1}^{K} \alpha_k^\Theta(o, c) \pi_k^\Theta(a|o, c) + \Psi \]

\[ \pi_k^\Theta(a|o, c) = \mathcal{N}(a | \mu_k^\Theta(o, c), \text{diag}(\sigma_k^\Theta(o, c))^2) \]

Training:

- Step 1: Learn Mixture of Experts: \( \mathcal{L}_{\text{MoE}} = -\log \left[ \sum_{k=1}^{K} \alpha_k^\Theta \pi_k^\Theta \right] + \mathcal{L}_V + \mathcal{L}_R \)
- Step 2: Learn Context Embedding: \( \mathcal{L}_{\text{VAE}} = \beta \text{KL} (q_\Phi(z|I) \parallel p_0(z)) + ||d_\Phi(z) - I||_2^2 \)
- Step 3: Task-driven optimization: \( J_{\text{TASK}}(\theta_{\text{readout}}, \Psi) = \mathbb{E}_{\pi_\Theta} \left[ \sum_{t=0}^{T} r_t \right] \)
Experiments
CARLA Benchmark

- Random start and end location, 4 known weathers, 2 unseen weathers
- Metric: Percentage of successfully completed episodes (success rate)
- Collision does not necessarily terminate episode

NoCrash Benchmark

- Difficulty varies with number of dynamic agents in the scene
- Empty: 0 Agents  Regular: 65 Agents  Dense: 220 Agents
- All collisions terminate episode
AnyWeather Benchmark

Evaluation on 10 unseen weathers, quantifies generalization performance

Importance of Mixture Model

<table>
<thead>
<tr>
<th>Evaluation Task</th>
<th>Training Data and Mixture Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Navigation (Static, K=1)</td>
</tr>
<tr>
<td>Straight (Static)</td>
<td>99</td>
</tr>
<tr>
<td>One Turn (Static)</td>
<td>98</td>
</tr>
<tr>
<td>Navigation (Static)</td>
<td>96</td>
</tr>
<tr>
<td>Navigation (Dynamic)</td>
<td>40</td>
</tr>
</tbody>
</table>

Results of Mixture Model on CARLA Benchmark:

- Static model solves static scenes well but cannot handle dynamic objects
- Dynamic model handles dynamic scenes better but degrades on static scenes
- Dynamic mixture model generalizes to all scenarios (without on-policy data)
Importance of Mixture Model and Task-based Refinement

<table>
<thead>
<tr>
<th>Model</th>
<th>Success Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic (K=1)</td>
<td>75</td>
</tr>
<tr>
<td>MoE Shared Backbone (K=3)</td>
<td>89</td>
</tr>
<tr>
<td>MoE Shared Backbone (K=5)</td>
<td>90</td>
</tr>
<tr>
<td>MoE Shared Backbone (K=8)</td>
<td>87</td>
</tr>
<tr>
<td>MoE Separate Backbone (K=3)</td>
<td>94</td>
</tr>
<tr>
<td>MoE Separate Backbone (K=5)</td>
<td>93</td>
</tr>
<tr>
<td>MoE Separate Backbone (K=8)</td>
<td>93</td>
</tr>
<tr>
<td>MoE Separate Backbone + Refinement (K=3)</td>
<td><strong>98</strong></td>
</tr>
</tbody>
</table>

Results of Full Model on CARLA Benchmark:

- Performance improves up to 3 or 5 mixture components
- Separate backbones increase diversity and generalization
- Tasked-based refinement improves performance further
Emergent Driving Modes

Emergent Driving Modes:

- Acceleration distribution of three different experts during testing

Results on CARLA Benchmark

<table>
<thead>
<tr>
<th>Driving Task</th>
<th>CIRL</th>
<th>CILRS</th>
<th>CILRS (ours)</th>
<th>LSD (ours)</th>
<th>LSD+R (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight</td>
<td>100</td>
<td>96</td>
<td>96</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>One Turn</td>
<td>71</td>
<td>84</td>
<td>86</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>Navigation</td>
<td>53</td>
<td>69</td>
<td>67</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>Navigation Dynamic</td>
<td>41</td>
<td>66</td>
<td>64</td>
<td>94</td>
<td>98</td>
</tr>
</tbody>
</table>

- Using reward-based optimization alone (CIRL) is not sufficient
- LSD enables better driving behavior across all driving tasks
- Large improvements in the presence of dynamic objects
### Results on CARLA NoCrash Benchmark

<table>
<thead>
<tr>
<th>Driving Task</th>
<th>CILRS</th>
<th>CILRS+R</th>
<th>LSD (ours)</th>
<th>LSD+R (ours)</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>66 ± 2</td>
<td>65 ± 2</td>
<td>93 ± 2</td>
<td><strong>94 ± 1</strong></td>
<td>96 ± 0</td>
</tr>
<tr>
<td>Regular</td>
<td>49 ± 5</td>
<td>46 ± 2</td>
<td>66 ± 2</td>
<td><strong>68 ± 2</strong></td>
<td>91 ± 1</td>
</tr>
<tr>
<td>Dense</td>
<td>23 ± 1</td>
<td>20 ± 1</td>
<td>27 ± 2</td>
<td><strong>30 ± 4</strong></td>
<td>41 ± 2</td>
</tr>
</tbody>
</table>

- All methods perform worse due to challenges (density, collision terminations)
- Expert provided by CARLA often fails in dense environments (e.g., clogging)
- LSD enables better driving behavior across all driving tasks
Results on AnyWeather Benchmark

<table>
<thead>
<tr>
<th>Task</th>
<th>CILRS</th>
<th>LSD (ours)</th>
<th>LSD+R (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight</td>
<td>83.2</td>
<td>85.2</td>
<td>85.6</td>
</tr>
<tr>
<td>One Turn</td>
<td>78.4</td>
<td>80.4</td>
<td>81.6</td>
</tr>
<tr>
<td>Navigation</td>
<td>76.4</td>
<td>78.8</td>
<td>79.6</td>
</tr>
<tr>
<td>Nav. Dynamic</td>
<td>75.6</td>
<td>77.2</td>
<td>78.4</td>
</tr>
</tbody>
</table>

- AnyWeather benchmark test generalization to challenging unseen weathers
- All methods can fail even on simple straight driving tasks
- Some challenging weathers lead to zero success rate for all methods
- More research is required to address these challenges
Qualitative Results
How useful is **data aggregation** for self-driving?
Imitation Learning

Hard coding policies is often difficult ⇒ Rather use a data-driven approach!

▶ **Given:** demonstrations or demonstrator
▶ **Goal:** train a policy to mimic decision
Formal Definition of Imitation Learning

**General Imitation Learning:**

\[
\begin{align*}
\arg\min_{\theta} \mathbb{E}_{s \sim P(s|\pi_\theta)} [\mathcal{L} (\pi^*(s), \pi_\theta(s))] \\
\text{▶ State distribution } P(s|\pi_\theta) \text{ depends on rollout determined by current policy } \pi_\theta
\end{align*}
\]

**Behavior Cloning:**

\[
\begin{align*}
\arg\min_{\theta} \mathbb{E}_{(s^*,a^*) \sim P^*} [\mathcal{L} (a^*, \pi_\theta(s^*))] \\
= \sum_{i=1}^{N} \mathcal{L}(a^*_i, \pi_\theta(s^*_i)) \\
\text{▶ State distribution } P^* \text{ provided by expert} \\
\text{▶ Reduces to supervised learning problem}
\end{align*}
\]
Challenges of Behavior Cloning

- Behavior cloning makes IID assumption
  - Next state is sampled from states observed during expert demonstration
  - Thus, next state is sampled independently from action predicted by current policy

- What if $\pi_\theta$ makes a mistake?
  - Enters new states that haven’t been observed before
  - New states not sampled from same (expert) distribution anymore
  - Cannot recover, can lead to catastrophic failure
Experiment by Held and Hein

**Data Aggregation (DAgger):**

- Iteratively build a set of inputs that the final policy is likely to encounter based on previous experience. Query expert for aggregate dataset.
- But can easily overfit to main mode of demonstrations
- High training variance (random initialization, order of data)

---

Ross, Gordon and Bagnell: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS, 2011.
Distribution over Driving Actions

Imitation Learning

DAgger Iter 1

DAgger Iter 2

DAgger Iter 3

DAgger with Critical States and Replay Buffer

Key Ideas:

1. Sample **critical states** from the collected on-policy data based on the utility they provide to the learned policy in terms of driving behavior

2. Incorporate a **replay buffer** which progressively focuses on the high uncertainty regions of the policy's state distribution
Sampling Strategies:

- Task-based: Sample uniformly from “left”, “right”, “straight”
- Policy-based: Use test-time dropout to estimate epistemic uncertainty
- Expert-based: Highest loss or deviation in brake signal wrt. expert
Distribution over Driving Actions

Imitation Learning

Our Approach Iter 1

Our Approach Iter 2

Our Approach Iter 3

Experiments
Evaluation

- CARLA **NoCrash benchmark**
- **Dense setting** with 220 agents
- Comparison to various baselines with (+) and without data augmentation
Data augmentation increases the performance of all methods

DAgger overfits quickly (!), not better than data augmentation

Our model consistently improves upon the baselines in all conditions

Data augmentation increases the performance of all methods

DAgger overfits quickly (!), not better than data augmentation

Our model consistently improves upon the baselines in all conditions
Data augmentation increases the performance of all methods
DAgger overfits quickly (!), not better than data augmentation
Our model consistently improves upon the baselines in all conditions
Evaluation

<table>
<thead>
<tr>
<th>Task</th>
<th>CILRS(^+)</th>
<th>DART</th>
<th>DA-RB(^+) (Ours)</th>
<th>DA-RB(^+)(E) (Ours)</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>45±6</td>
<td>50±1</td>
<td>62±1</td>
<td>66±5</td>
<td>71±4</td>
</tr>
<tr>
<td>New Weather</td>
<td>39±4</td>
<td>37±2</td>
<td>60±1</td>
<td>56±1</td>
<td>72±3</td>
</tr>
<tr>
<td>New Town</td>
<td>23±1</td>
<td>26±2</td>
<td>34±2</td>
<td>36±3</td>
<td>41±2</td>
</tr>
<tr>
<td>New Town &amp; Weather</td>
<td>26±2</td>
<td>21±1</td>
<td>25±1</td>
<td>35±2</td>
<td>43±2</td>
</tr>
</tbody>
</table>

Mean and standard deviation over 3 evaluation runs.

- Ensemble (E) improves performance further (particularly in new environment)
- Expert provided by CARLA often fails in dense environments (e.g., clogging)
Significant reduction in collisions with dynamic objects
More time-outs due to less infractions (e.g., clogged scenes, red lights)
## Training Variance

<table>
<thead>
<tr>
<th>Iter</th>
<th>CILRS $^+$</th>
<th>DAgger $^+$</th>
<th>DA-RB $^+$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14.6 ± 3.4</td>
<td>14.6 ± 3.4</td>
<td>14.6 ± 3.4</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>15.2 ± 5.1</td>
<td>24.8 ± 1.9</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>13.2 ± 1.9</td>
<td>25.4 ± 1.5</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>17.8 ± 3.6</td>
<td>27.0 ± 0.9</td>
</tr>
</tbody>
</table>

Standard deviation wrt. 5 random training seeds (New Town & Weather)

- Significant reduction in variance compared to CILRS and DAgger
- Sampling the dataset based on critical states is crucial for variance reduction
Interpretability: GradCAM Attention Maps

CILRS [Codevilla et al. 2019]  Our Approach
Interpretability: GradCAM Attention Maps

CILRS [Codevilla et al. 2019]  
Our Approach

Qualitative Results

CILRS+ (Codevilla et al. 2019)  DA-RB+ (Our Approach)
Summary
Summary

- A single imitation learner cannot capture the complexities of driving
- Mixture of experts can significantly improve generalization
- Task-driven optimization is difficult but important
- Data augmentation is important but can easily overfit in self-driving
- Sampling critical states in a replay buffer improves aggregation performance
- Training variance can be reduced using this strategy
- Generalization to all CARLA weathers remains unsolved
- Better experts and improvements in the CARLA simulation are needed
  (and on their way – current version is CARLA 0.9.9!)
Thank you!

http://autonomousvision.github.io