ChauffeurNet

Learning to Drive
by Imitating the Best and Synthesizing the Worst

Reading Club (Oct 17): ChauffeurNet (1812.03079.pdf)

Abstract

- Train a policy for autonomous driving via **imitation learning** that is **robust enough to drive a real vehicle**
- Expose the learner to synthesized data in the form of **perturbations to the expert's driving** (including collisions and going off road)
- Augment the imitation loss with additional losses that penalize undesirable events and encourage progress

Additional Problem Framing

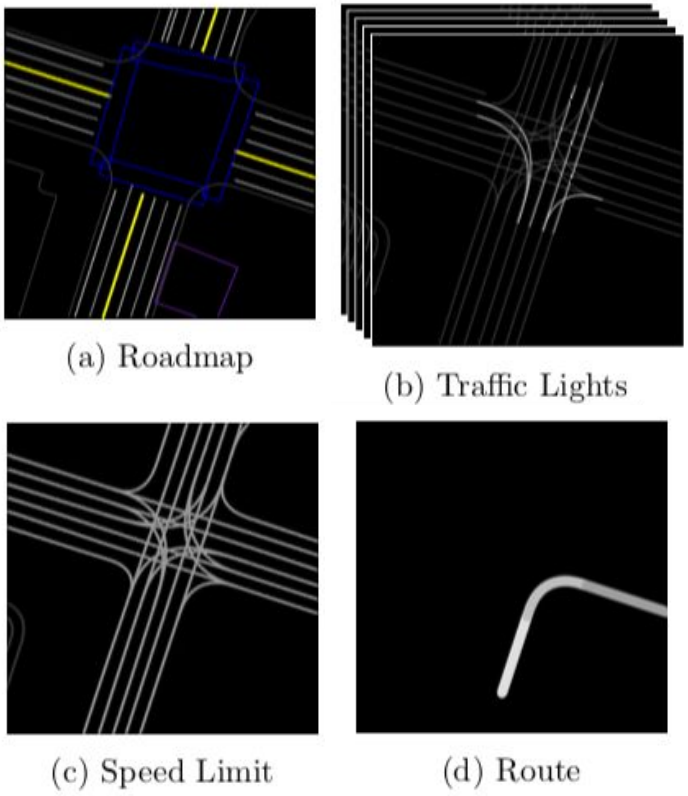
- Even with 30 million examples, and even with mid-level input and output representations that remove the burden of perception and control, pure imitation learning is not sufficient
- The key challenge is that we need to run the system closed- loop, where errors accumulate and induce a shift from the training distribution
  - “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”
- Surmountable if we augment the imitation loss with losses that discourage bad behavior and encourage progress, and, importantly, augment our data with synthesized perturbations in the driving trajectory

From 30,000 ft. (1/2)

Raw Sensor Data
(E.x. Camera & Lidar)

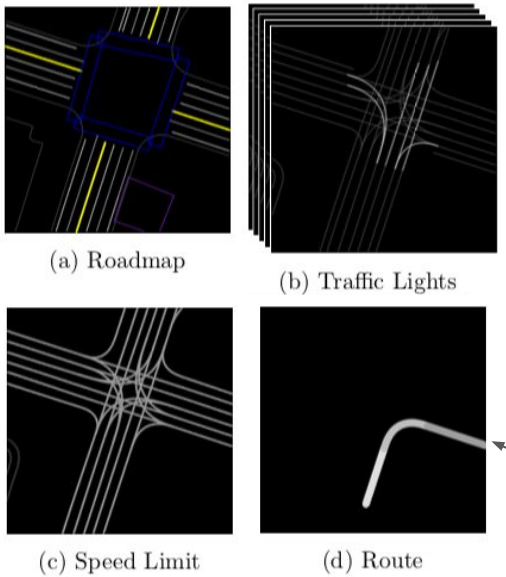
Perception System

Top-Down 2D Environment Representations



From 30,000 ft. (2/2)

Top-Down 2D Environment Representations



ChauffeurNet (RNN) → **Driving Trajectories** → **Steering & Acceleration** (via a controller)

These mid-level representations could come from simulation or real-world data.

Imitation Learning

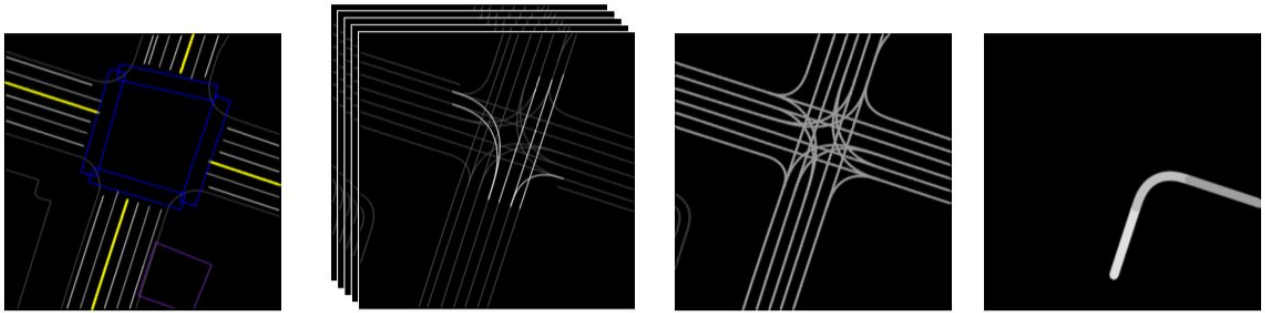
- Learning how to make sequences of decisions in an environment, where the training signal comes from demonstrations
- Counterpart to reinforcement learning, where the goal is to optimize some reward function of the environment
- May be preferred when a when a reward function is difficult to specify (e.g., act “friendly”), or when the reward function is sparse and difficult to optimize directly

Input Output Representation

We begin by describing our top-down input representation that the network will process to output a drivable trajectory. At any time t, our agent (or vehicle) may be represented in a top-down coordinate system by $\mathbf{p}_t, \theta_t, s_t$, where $\mathbf{p}_t = (x_t, y_t)$ denotes the agent's location or pose, θ_t denotes the heading or orientation, and s_t denotes the speed. The top-down coordinate system is picked such that our agent's pose $\mathbf{p}_0$ at the current time $t = 0$ is always at a fixed location (u_0, v_0) within the image. For data augmentation purposes during training, the orientation of the coordinate system is randomly picked for each training example to be within an angular range of $\theta_0 \pm \Delta$, where θ_0 denotes the heading or orientation of our agent at time $t = 0$. The top-down view is represented by a set of images of size $W \times H$ pixels, at a ground sampling resolution of ϕ meters/pixel. Note that as the agent

Driving Model Inputs (a-d)

- (a) **Roadmap**: a color (3-channel) image with a rendering of various map features such as lanes, stop signs, cross-walks, curbs, etc.
- (b) **Traffic lights**: a temporal sequence of grayscale images where each frame of the sequence represents the known state of the traffic lights at each past timestep. Within each frame, we color each lane center by a gray level with the brightest level for red lights, intermediate gray level for yellow lights, and a darker level for green or unknown lights
- (c) **Speed limit**: a single channel image with lane centers colored in proportion to their known speed limit.
- (d) **Route**: the intended route along which we wish to drive, generated by a router (think of a Google Maps-style route).



(a) Roadmap (b) Traffic Lights (c) Speed Limit (d) Route

Driving Model Inputs (e-g)

- (e) **Current agent box:** this shows our agent's full bounding box at the current timestep t = 0.
- (f) **Dynamic objects in the environment:** a temporal sequence of images showing all the potential dynamic objects (vehicles, cyclists, pedestrians) rendered as oriented boxes.
- (g) **Past agent poses:** the past poses of our agent are rendered into a single grayscale image as a trail of points.

(e) Current Agent Box (f) Dynamic Boxes (g) Past Agent Poses (h) Future Agent Poses

Takeaway 1: Intermediate Input Representations

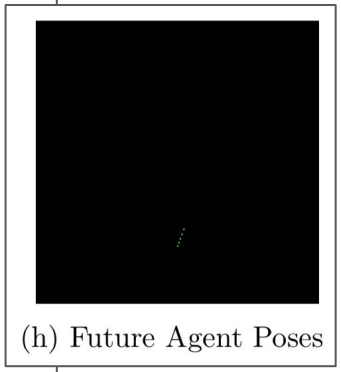
- Easy to generate input data from simulation or create it from real-sensor logs
- Enables testing and validation of models in closed-loop simulations before running them on a real car
- Allows the same model to be improved using simulated data to adequately explore rare situations such as collisions for which real-world data might be difficult to obtain
- Using a top-down 2D view means efficient convolutional inputs, and allows flexibility to represent metadata and spatial relationships in a human-readable format.

Timestep (e.x. 0.2s)

Mid-level Input Representations

$$\mathbf{p}_{t+\delta t} = \text{ChauffeurNet}(I, \mathbf{p}_t)$$

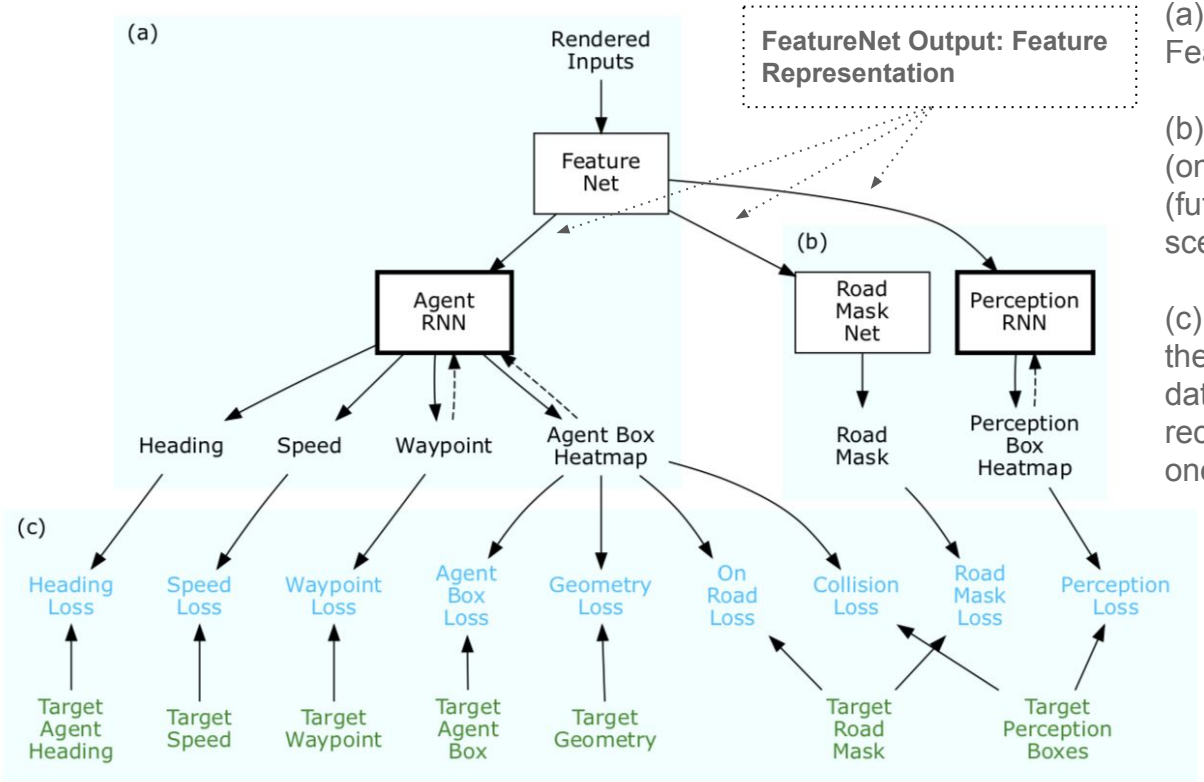
Agent's Pose



(h) Future Agent Poses

- **Takeaway 2:** Performs N iterations and outputs a future trajectory
- Trajectory can be fed to a controls optimizer that computes detailed driving control (such as steering and braking commands)
- Different types of vehicles may possibly utilize different control outputs to achieve the same driving trajectory

Model Design



Training the driving model.

(a) The core ChauffeurNet model with a FeatureNet and an AgentRNN

(b) Co-trained RoadMask prediction net (on-road vs. off-road) and PerceptionRNN (future location of every other agent in the scene)

(c) Training losses are shown in blue, and the green labels depict the ground-truth data. The dashed arrows represent the recurrent feedback of predictions from one iteration to the next.

AgentRNN Imitation Losses

Prob. dist. over spatial coordinates of the predicted waypoint

Cross-entropy loss

Binary mask ground truth with a single pixel activated

$$\mathcal{L}_p = \mathcal{H}(P_k, P_k^{gt})$$

$$\mathcal{L}_B = \frac{1}{WH} \sum_x \sum_y \mathcal{H}(B_k(x,y), B_k^{gt}(x,y))$$

$$\mathcal{L}_\theta = \left\| \theta_k - \theta_k^{gt} \right\|_1$$

Box heading

Past Motion Dropout

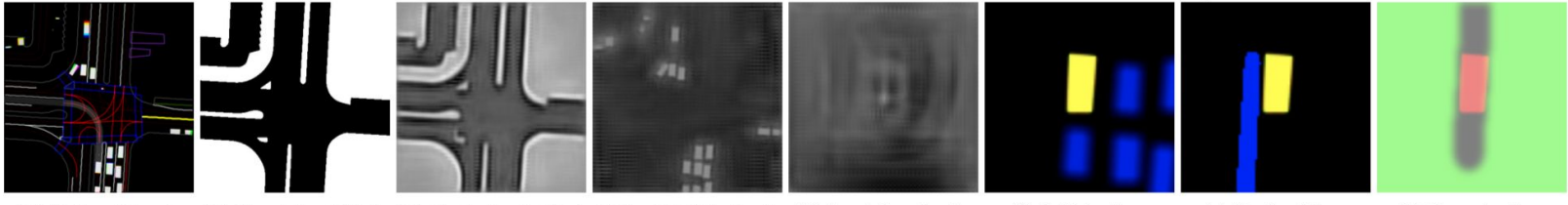
During training, the model is provided the past motion history as one of the inputs (Fig. 1(g)). Since the past motion history during training is from an expert demonstration, the net can learn to "cheat" by just extrapolating from the past rather than finding the underlying causes of the behavior. During closed-loop inference, this breaks down because the past history is from the net's own past predictions. For example, such a trained net may learn to only stop for a stop sign if it sees a deceleration in the past history, and will therefore never stop for a stop sign during closed-loop inference. To address this, we introduce a dropout on the past pose history, where for 50% of the examples, we keep only the current position (u_0, v_0) of the agent in the past agent poses channel of the input data. This forces the net to look at other cues in the environment to explain the future motion profile in the training example.

Takeaway 3: Beyond Pure IL

- Synthesizing **Perturbations**: To prevent the input data from deviating from the training distribution over time when the model is part of a closed-loop system (see image).
- Additional Losses (imitation vs env; refer to paper)
  - Collision Loss (5.2.1)
  - Road Loss (5.2.2)
  - Geometry Loss (5.2.3)
  - Auxiliary (5.2.4), PerceptionRNN

(a) Original

(b) Perturbed



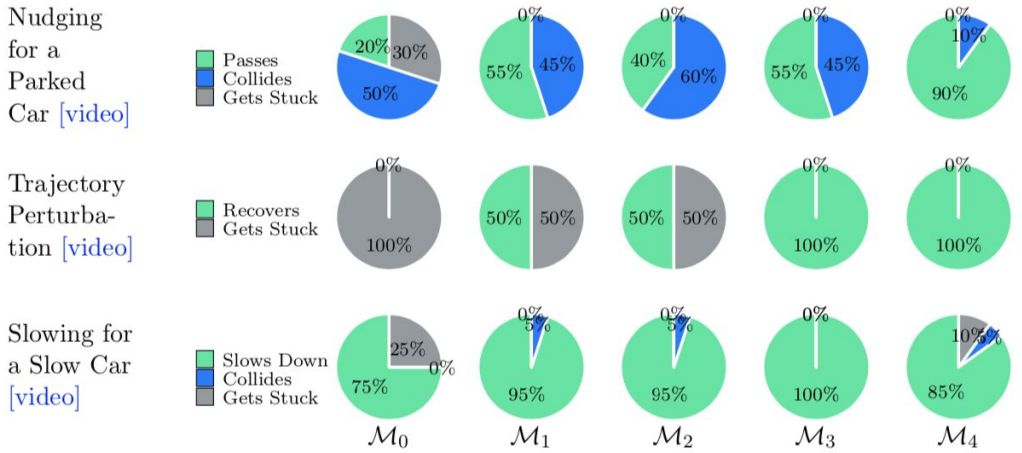
(a) Flattened Inputs (b) Target Road Mask (c) Pred Road Mask (d) Pred Vehicles Logits (e) Agent Pose Logits (f) Collision Loss (g) On Road Loss (h) Geometry Loss

Evaluation (refer to paper)

- Closed Loop
- Open Loop
- Scenarios
  - Nudging around a parked car
  - Recovering from a trajectory perturbation
  - Slowing down for a slow car
- https://sites.google.com/view/waymo-learn-to-drive/
  - Model Ablations
  - Real World Driving

Model	Description	w_{imit}	w_{env}
$\mathcal{M}_0$	Imitation with Past Dropout	1.0	0.0
$\mathcal{M}_1$	$\mathcal{M}_0$ + Traj Perturbation	1.0	0.0
$\mathcal{M}_2$	$\mathcal{M}_1$ + Environment Losses	1.0	1.0
$\mathcal{M}_3$	$\mathcal{M}_2$ with less imitation	0.5	1.0
$\mathcal{M}_4$	$\mathcal{M}_2$ with Imitation Dropout	*Dropout probability = 0.5 (see Section 5.3).*	

Table 3: Model configuration for the model ablation tests.



Conclusion

In this paper, we presented our experience with what it took to get imitation learning to perform well in real-world driving. We found that key to its success is synthesizing interesting situations around the expert's behavior and augmenting appropriate losses that discourage undesirable behavior. This constrained exploration is what allowed us to avoid collisions and off-road driving even though such examples were not explicitly present in the expert's demonstrations. To support it, and to best leverage the expert data, we used middle-level input and output representations which allow easy mixing of real and simulated data and alleviate the burdens of learning perception and control.