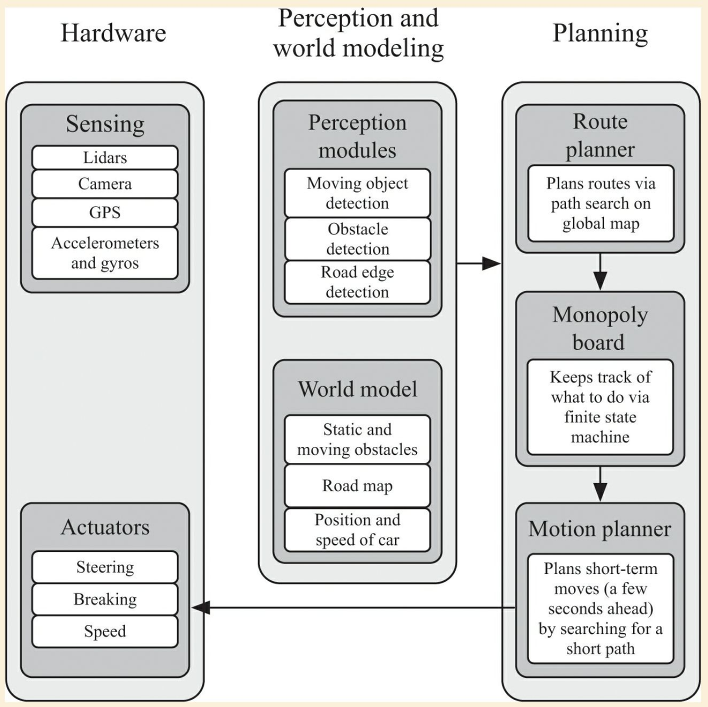
Autonomous Vehicles (Unity + ML Agents)

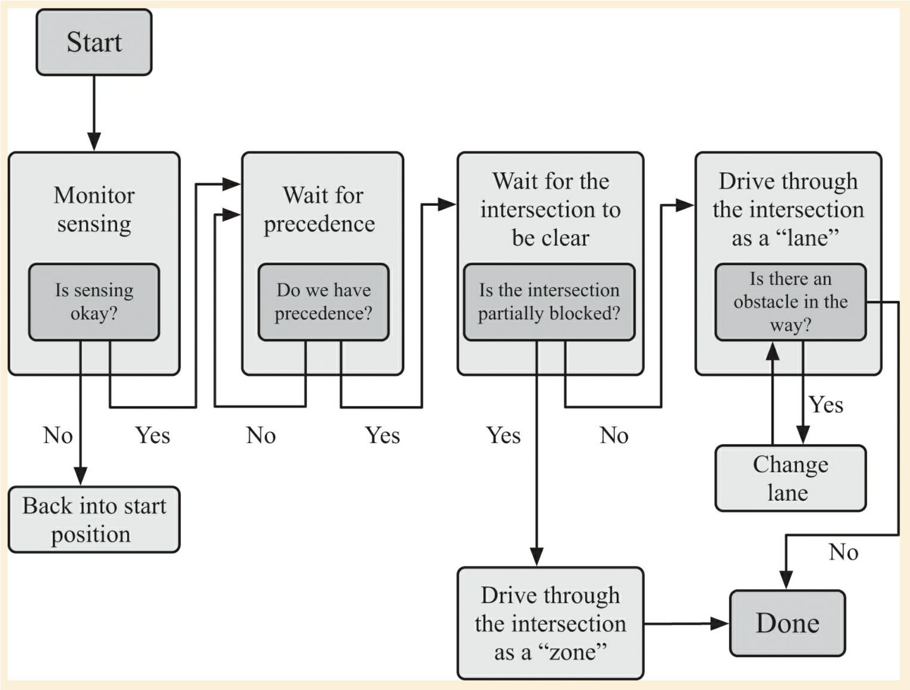
Advanced Deep Learning (EECS 496)

Grant Gasser, Blaine Rothrock

Sources: DeepMind, Tesla, Sean Gerrish, Unity

Autonomous Vehicles Background





Autonomous Vehicles - Sensory Input

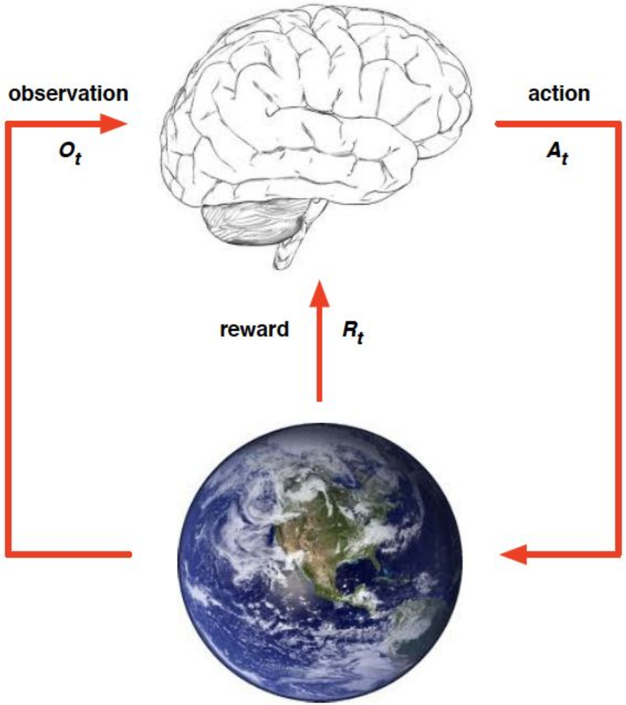
- Sensor Debate
  - To Lidar or to not Lidar? That is the question.
  - Tesla (no Lidar) and Waymo, Lyft, Baidu, Cruise (use Lidar)

Our Approach

- Autonomous Vehicle Simulation
  - Focus on one peice of the planning component: **Lane keeping**
- Two step approach

1. Train a **Reinforcement Learning Model** in fully observable environment
2. Run trained RL model to generate images to train a **CNN model**

Reinforcement Learning Review



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

Reinforcement Learning Review - Value vs. Policy

- Value Based Learning

Definition

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right]$$

- Policy Based Learning

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

RL - Proximal Policy Optimization (PPO)

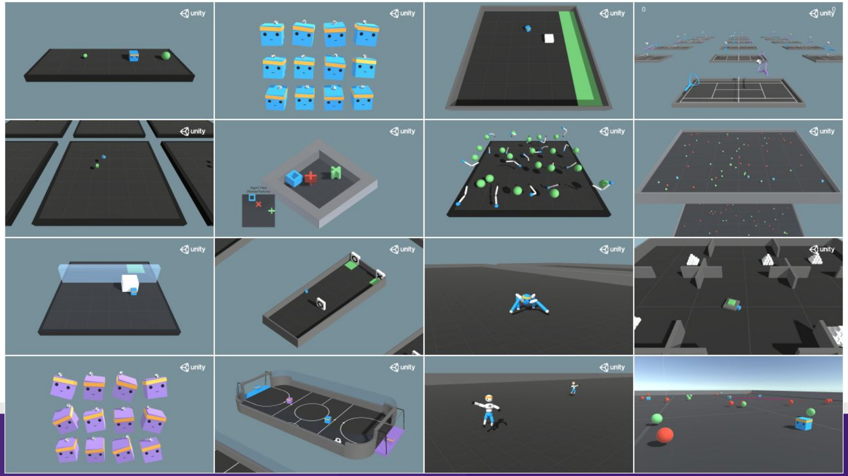
- OpenAI: “[PPO] performs comparably or better than state-of-the-art approaches while being much simpler to implement and tune. PPO has become the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance.”
- Learn neural network to approximate best function that maps agent’s observations to an action given a state
- Objective Function:

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\,]$$

- θ is the policy parameter
- $\hat{E}_t$ denotes the empirical expectation over timesteps
- r_t is the ratio of the probability under the new and old policies, respectively
- $\hat{A}_t$ is the estimated advantage at time t
- ε is a hyperparameter, usually 0.1 or 0.2

Unity & ML Agents

- **Unity:** a game development environment with physics engine
- **ML-Agents:** open source project developed by Unity (beta)
  - Bridging Python and Unity (C#)
  - Agent and Environment development for Reinforcement Learning
    - Supports PPO, SAC
    - Curriculum Training support
    - Customizable/extendable
  - TensorFlow 2

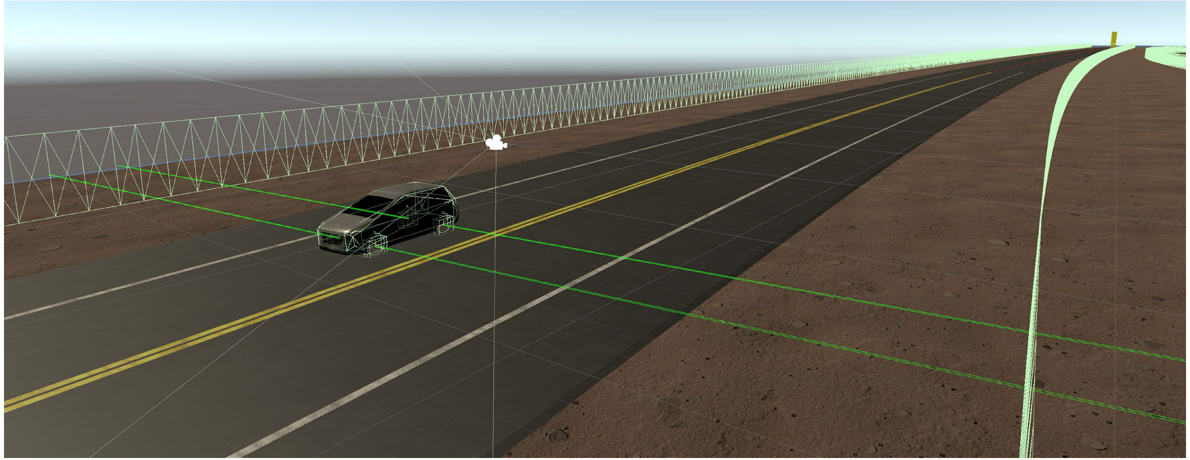


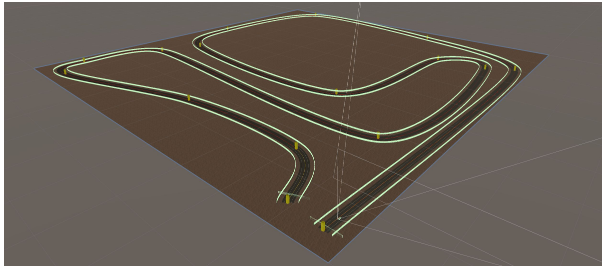
Project Description (Part 1 - RL)

- Create driving environment in **Unity**
    - Roads with lanes including curves, bridges, etc.
    - Realistic car and physics simulation
- Use ML Agents Toolkit to train agent (vehicle) to stay in the middle of lanes
    - Fully observable environment, send precise location data to agent
    - Use built-in reinforcement learning algorithms
    - 2 Deep RL options: **Proximal Policy Optimization (PPO)** or Soft Actor Critic (SAC)

RL: Vehicle Agent & Environment

- **Academy**
  - Manages Agents
- **Brain**
  - Delegate Actions from State
  - Inference from NN file
- **Agent**
  - Report State
  - Apply Actions





```
using UnityEngine;
using System;

using MLAgents;

public class VehicleAgent : Agent
{
    void Start() { ... starting code ... }

    public override void CollectObservations() { ... send state observations to the brain ... }

    public override void AgentAction(float[] vectorAction) { ... act on actions from the inference engine give, reward ... }

    public override float[] Heuristic() { ... manual control ... }

    public override void AgentReset() { ... reset the agents for new episode ... }
}
```

Behavior Parameters (Script)

Behavior Name	VehicleBrain
Vector Observation	
Space Size	5
Stacked Vectors	1
Vector Action	
Space Type	Continuous
Space Size	1
Model	VehicleBrain (NNModel)
Inference Device	CPU
Behavior Type	Default
Team ID	0
Use Child Sensors	✓

Vehicle Agent (Script)

Max Step	0
Reset On Done	✓
On Demand Decisions	
Decision Interval	4
Script	VehicleAgent
Max Angle	30
Max Torque	300
Brake Torque	30000
Wheel Shape	None (Game Object)
Critical Speed	5
Steps Below	5
Steps Above	1
Constant Torque	50
Road Guide Offset	15
Lane Width	5
Agent Reset Position	X 65 Y 0.82 Z 34.2
Agent Reset Rotation	X 0 Y 90 Z 0
End Box	EndBox
Drive Type	Front Wheel Drive
Current Angle	0

Autonomous Vehicles (Unity + ML Agents)

Presentation 2

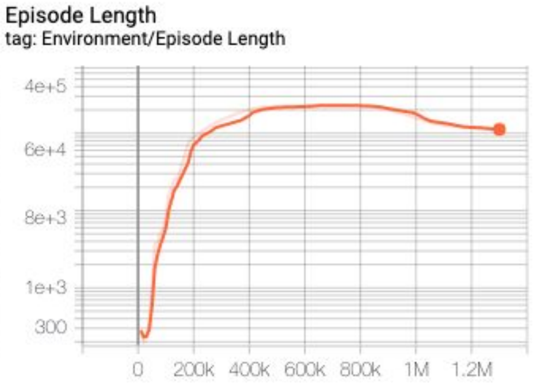
Grant Gasser, Blaine Rothrock

Sources: DeepMind, Tesla, Sean Gerrish, Unity

Project Overview

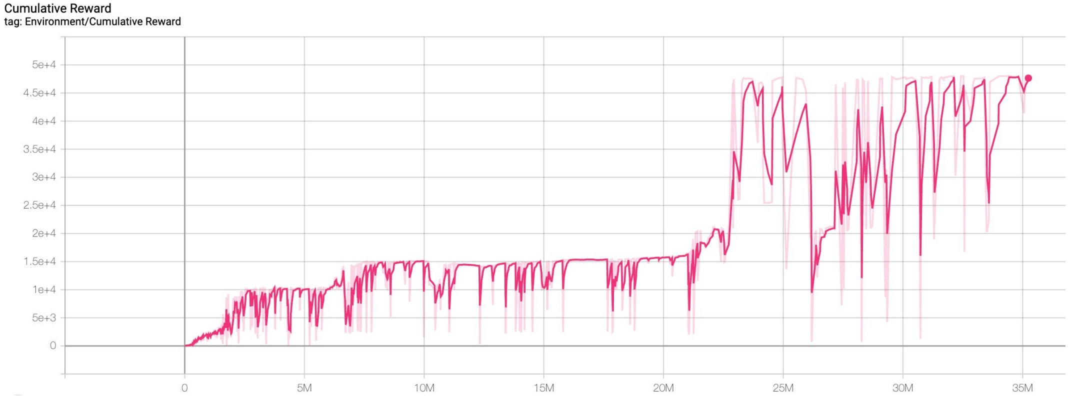
- Autonomous Vehicle Simulation using Unity and ML-Agents Toolkit
  - Focus on one peice of the planning component: **Lane keeping**
  - Train a Reinforcement Learning Model in fully observable environment
  - Run trained RL model to generate images to train a CNN model
- What we've completed:
  - A generalized model RL model for generating data
  - An image data set of ~11,000 256x256x3 images
  - Built the CNN Model
  - **Final steps**: Train CNN model on AWS & Integrate with Unity Environment

RL Improvements: First Model



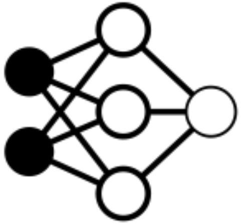
RL Improvements: New Model

- New Discrete Action Model
  - Smooth and generalizes for new tracks with various curvature
  - Small, frequent wheel angle updates
  - Introduced recurrent component
  - LOTS of training
  - Video



Part 2 (CNN)

- Convolutional Neural Network:
  - **Input**: image of what the car sees (pixels)
  - **Target**: wheel angle (+/-)
  - “Given the picture of the road, at what angle should the wheels be set?”

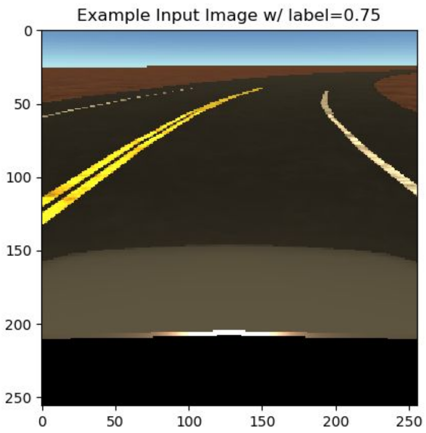


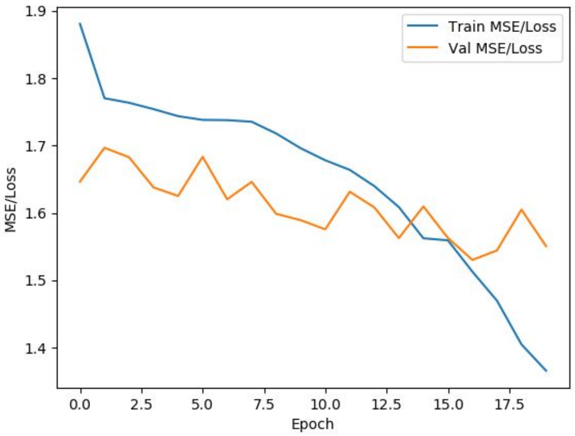
Turn wheel to
-1.5 degrees

CNN Version 1 (Trained locally on 800 images)

Architecture:

```
model = models.Sequential()
model.add(layers.Conv2D(x_train.shape[1], (3, 3), activation='relu', input_shape=(x_train.shape[1:])))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1))
```





Before the end of the quarter

- Train CNN Version 2 on AWS (~11,000 256x256x3 images)
    - Performance did not improve
- Test on unseen data
- Connect the model to Unity MLAgents with Python API