Visual Analytics
Final Presentation

Grant Fennessy

Guided Reinforcement Learning

A significant part of RL is constructing the reward function.

The purpose of these functions are to incentivize certain agent rewards.

“Behavior correctness” is very hard to evaluate, unlike simple annotated data.

We can manually annotate samples with “assumed behavior quality”. Is this action probably a good idea? A bad idea?

The model can then be evaluated based on if its predictions meet assumptions.

The model won’t learn off of this (no loss function integration), so domain experts can change the assumptions at any point, updating the visualization.

Goals

Allow users to see rewards attained by the model as it trains.

Provide easy comparison to past experiment reward values.

Present the prediction (behavior) quality per sample as training progresses.

Display model action taken for a sample at any given step.

Summary: Find out if the model is learning the desired behaviors!

Tasks

Determine early on when model training isn’t progressing quickly enough.

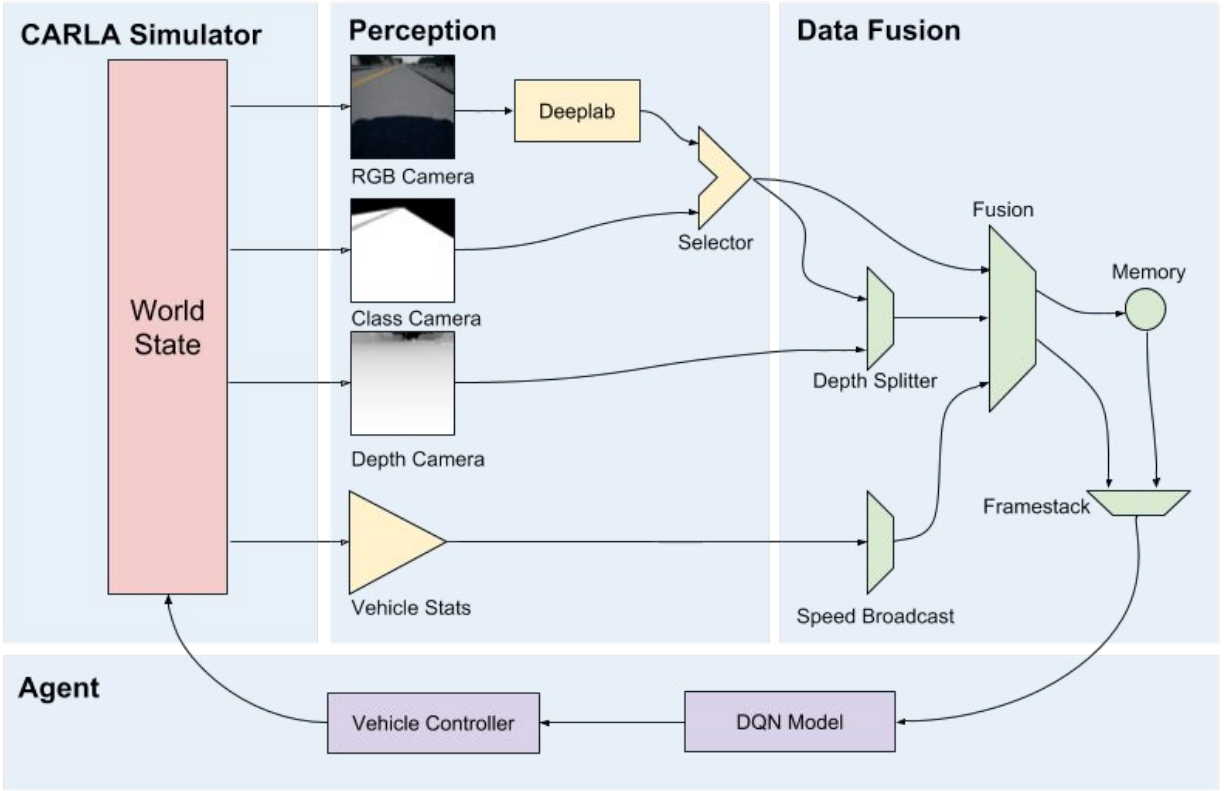
Identify when and where poor action choices are being made.

Make informed hyperparameter changes during model training.

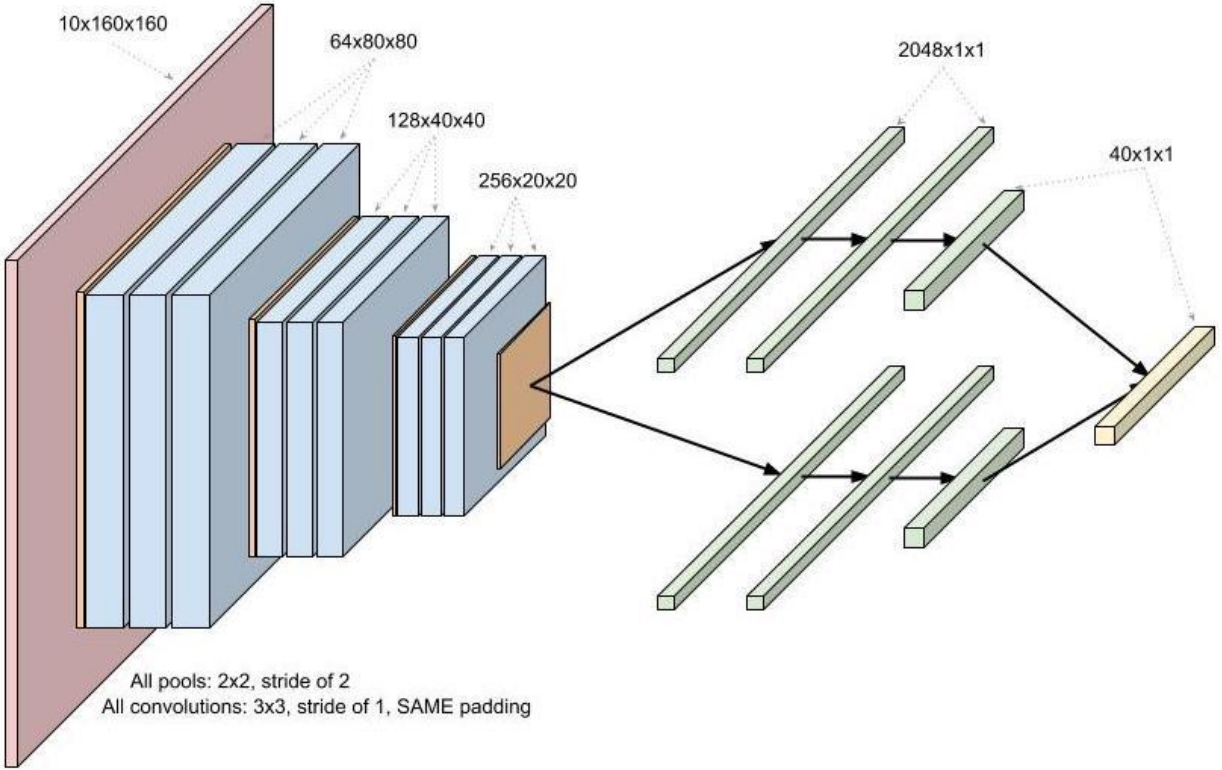
Future versions could even allow changing model exposure rates to certain sample types in an attempt to more resolve problematic decisions.

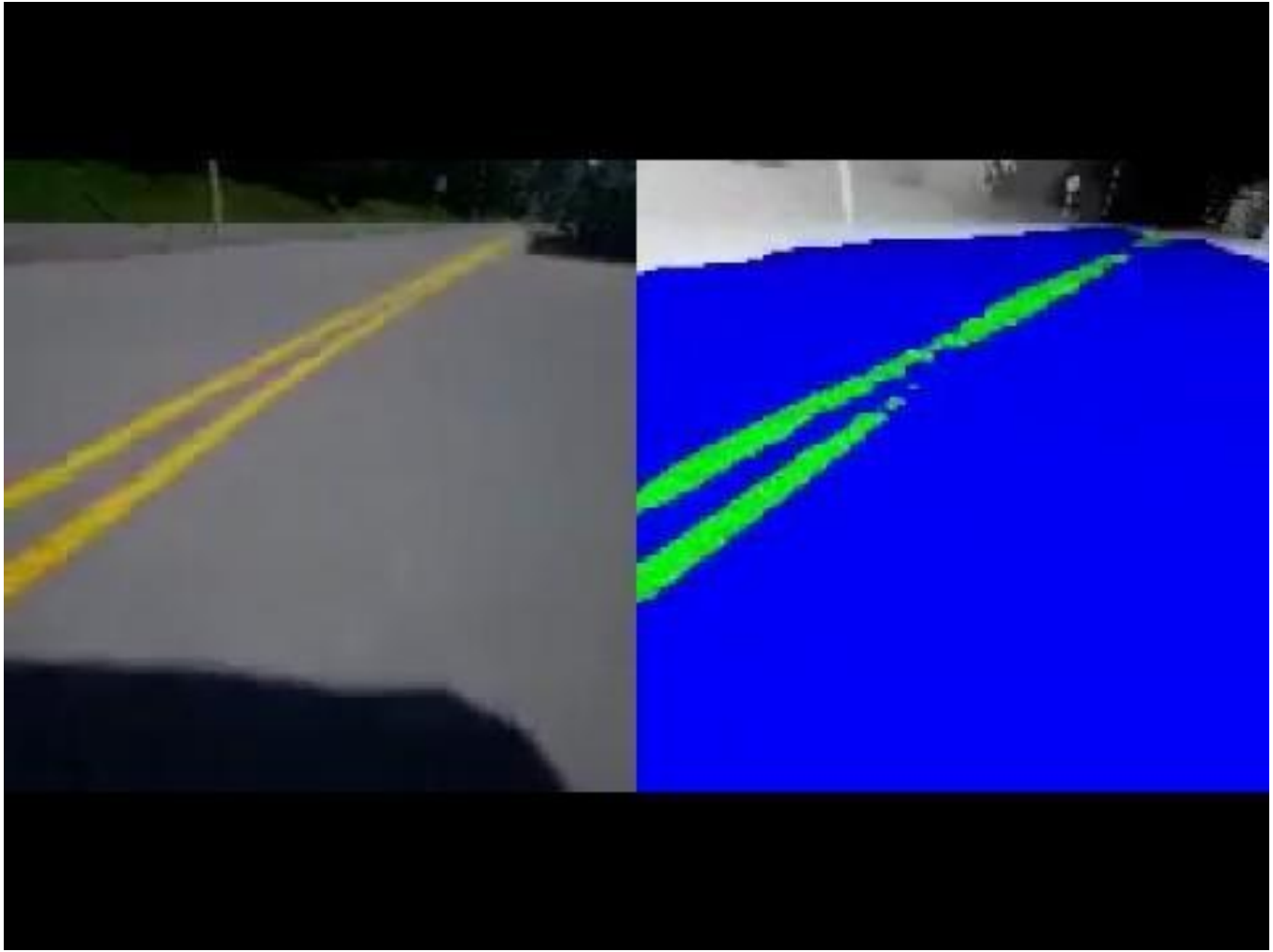
The Model

Autonomous Car Architecture



DDQN Model


10x160x160
64x80x80
128x40x40
256x20x20
2048x1x1
40x1x1
All pools: 2x2, stride of 2
All convolutions: 3x3, stride of 1, SAME padding



Data

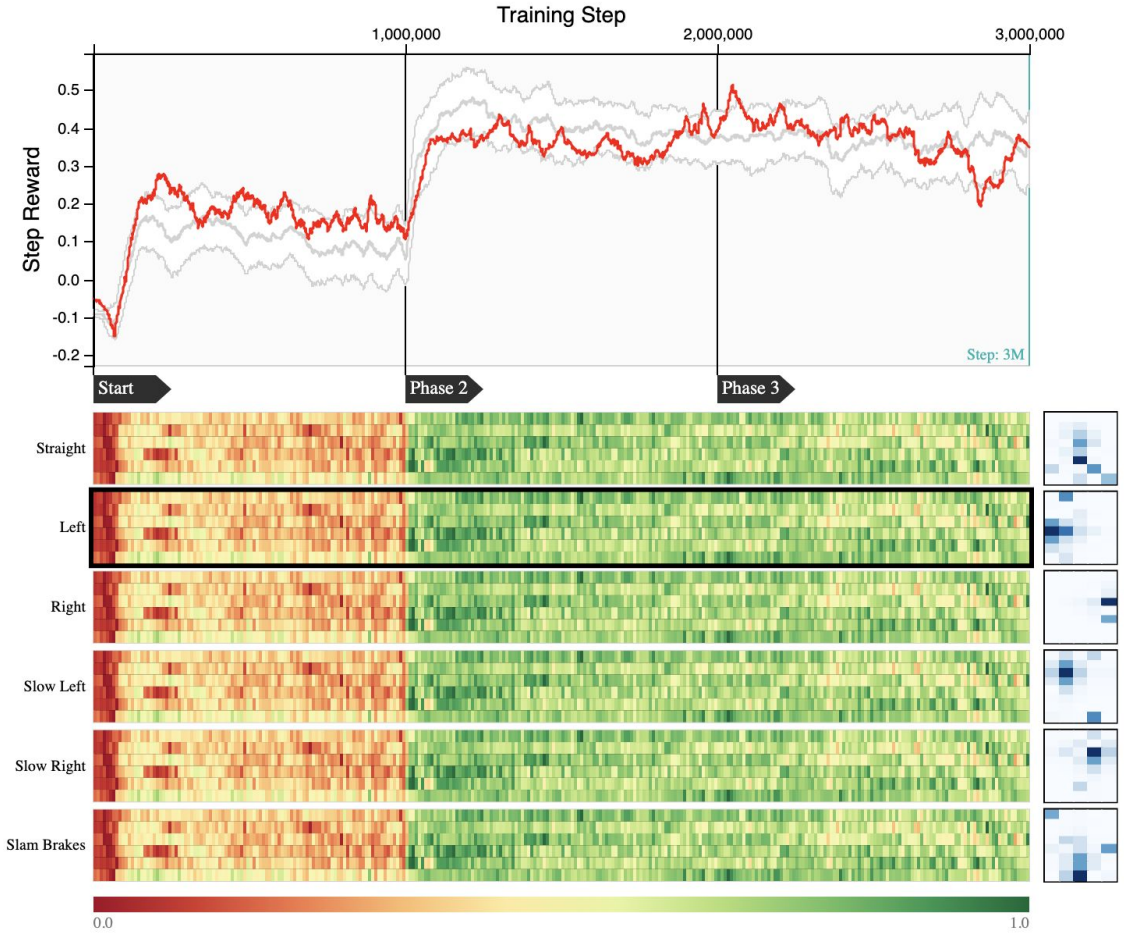
Every step the model outputs an attained reward. With 3mil steps per train, steps are aggregated into 1,000 steps and the mean value is saved.

I have rewards data for all steps across 12 experiments from my thesis - an autonomous vehicle agent driving around a small simulated town.

Every 10,000 steps, prediction is run on all samples, and the prediction results are stored for reference. This part is currently mocked.

Prediction data can then be loaded, along with labels for those samples (also currently mocked) for visualization purposes.

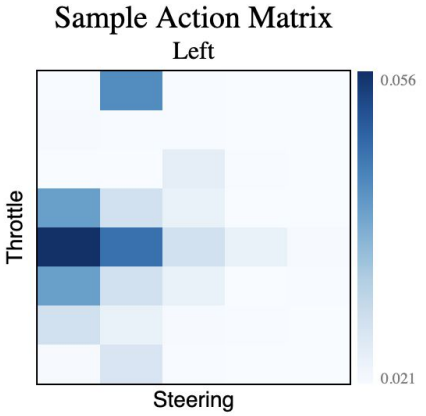
The Visualizations

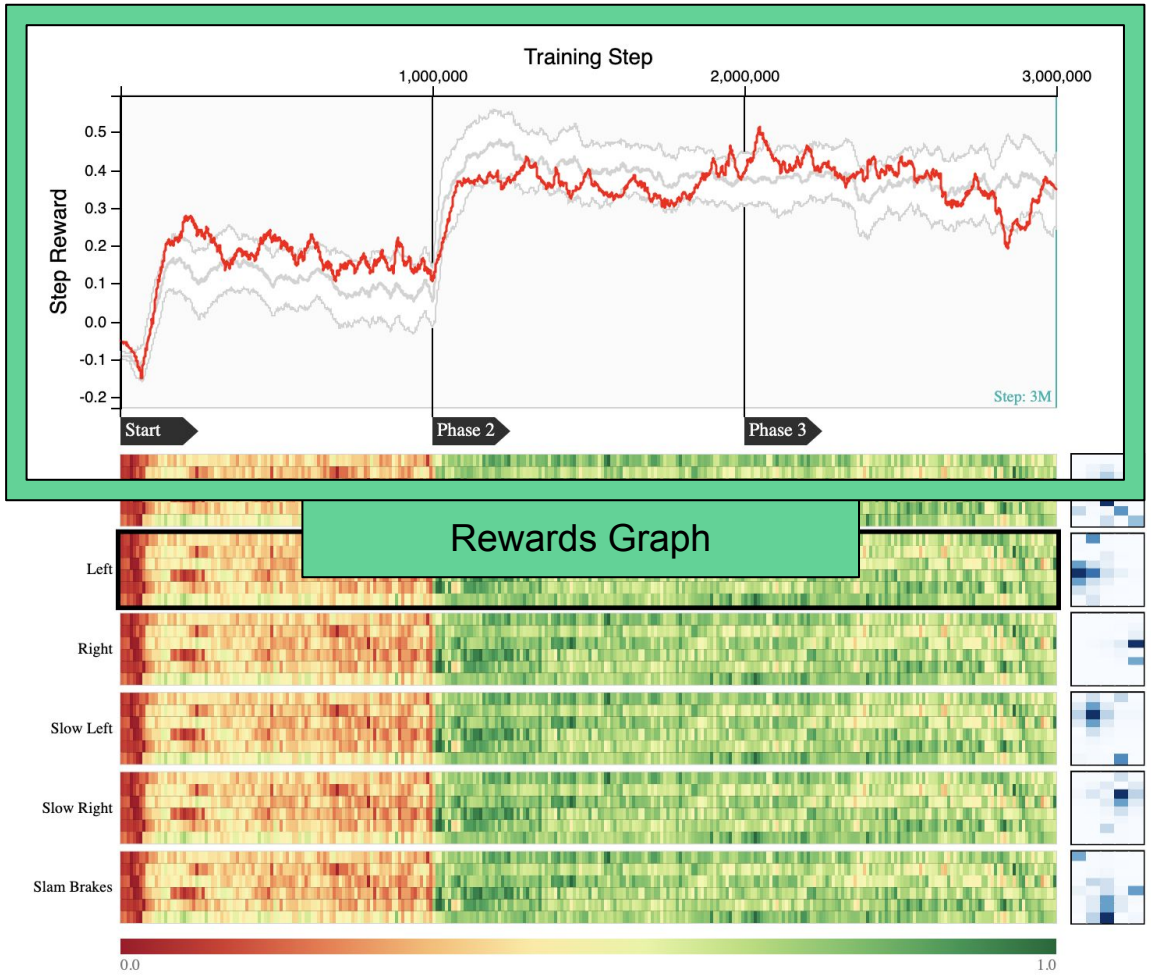


Event Parameters

Learning Rate	0.000001
Exploration Rate	0.05
Reward Shift	0
Step	

New Episode | Create Event





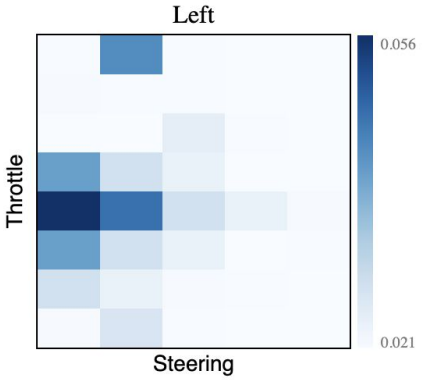
Rewards Graph

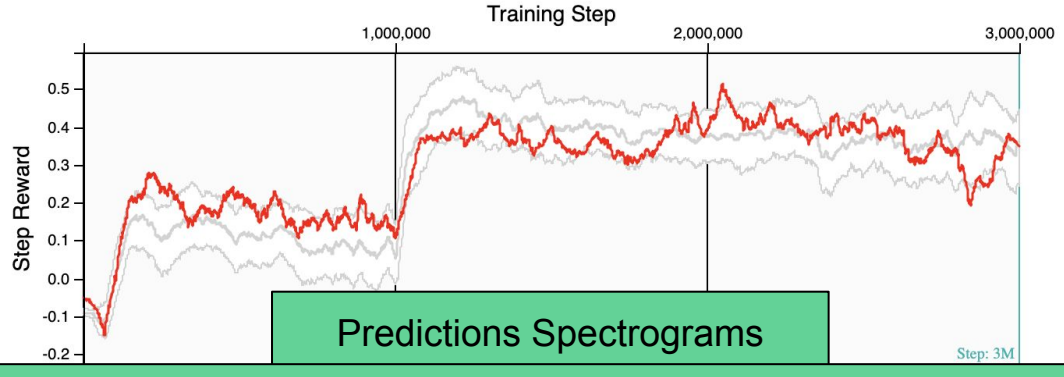
Event Parameters

Learning Rate	0.000001
Exploration Rate	0.05
Reward Shift	0
Step	

New Episode | Create Event

Sample Action Matrix



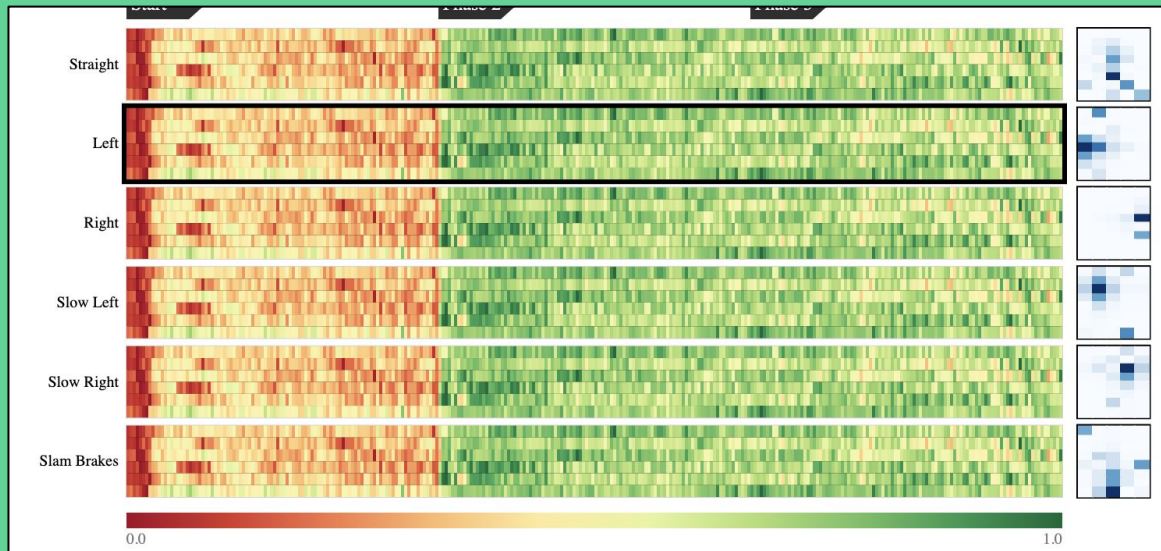


Event Parameters

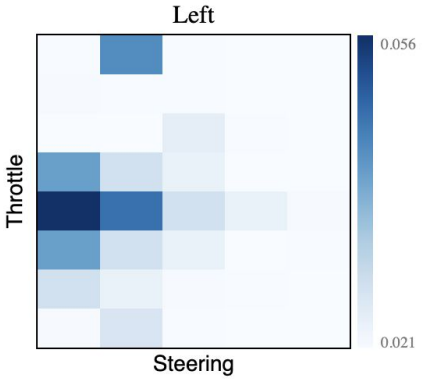
Learning Rate	0.000001
Exploration Rate	0.05
Reward Shift	0
Step	

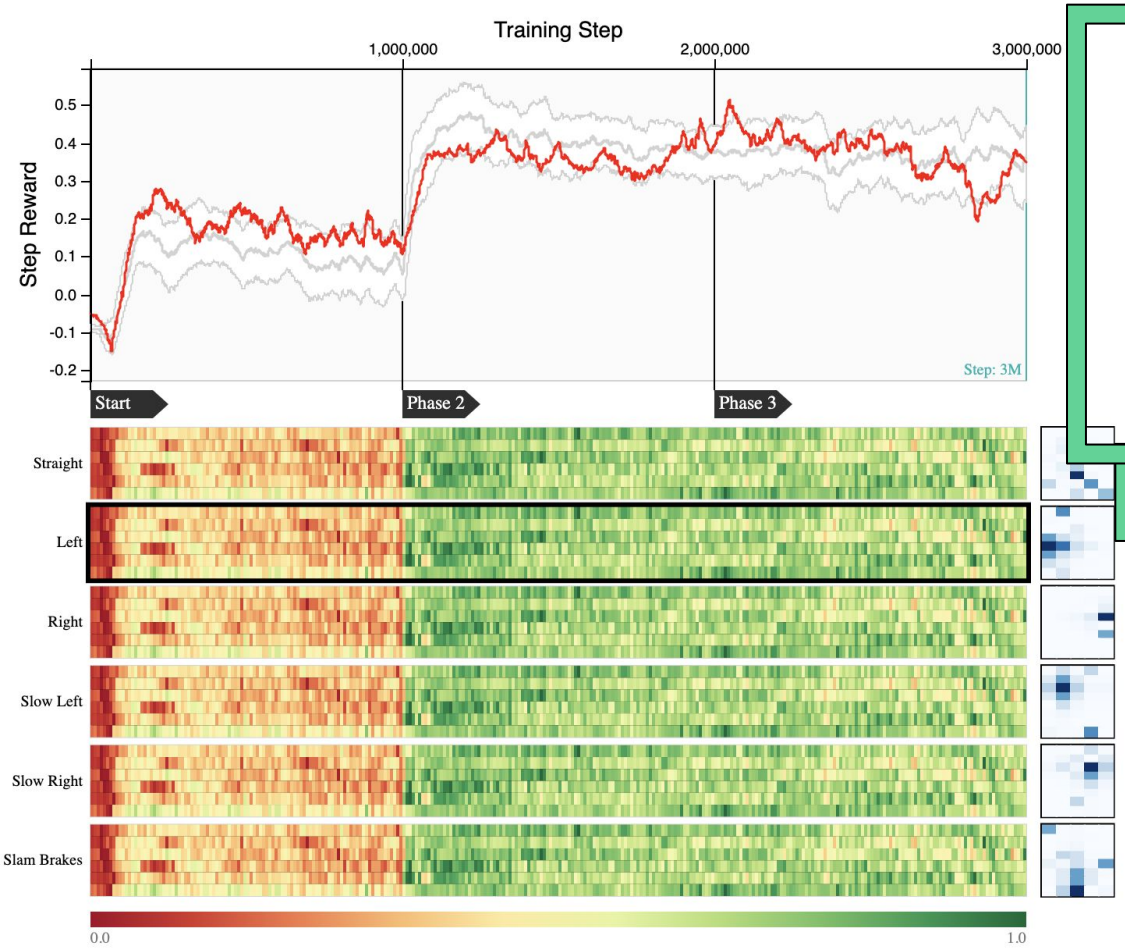
New Episode Create Event

Predictions Spectrograms



Sample Action Matrix



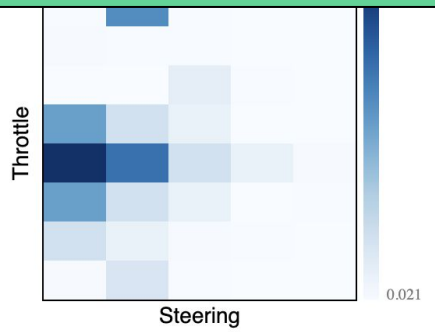


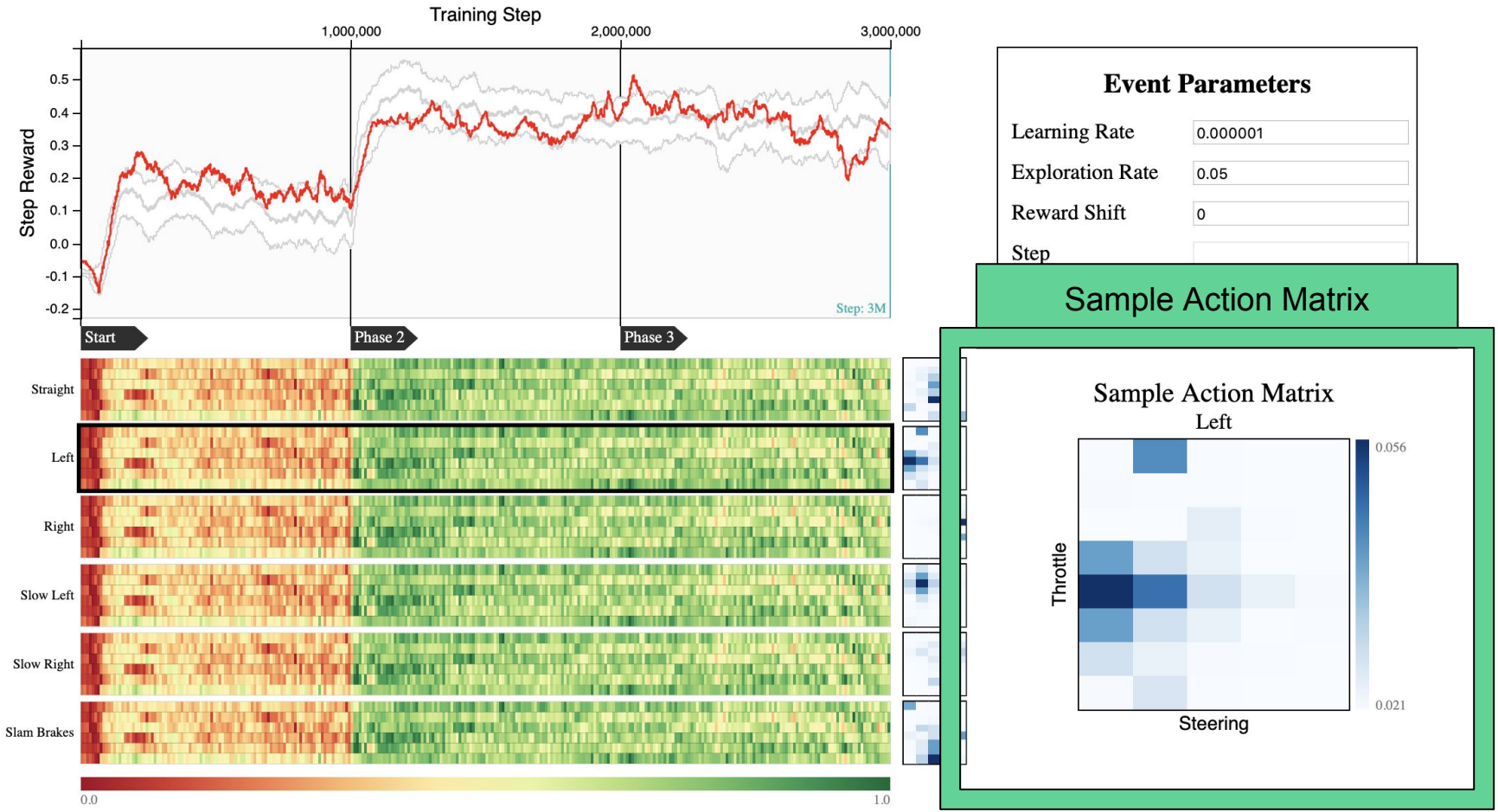
Event Parameters

Learning Rate	0.000001
Exploration Rate	0.05
Reward Shift	0
Step	

New Episode | Create Event

Selected Event Metadata




Training Step
1,000,000
2,000,000
3,000,000
Step Reward
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
Step: 3M
Start
Phase 2
Phase 3
Straight
Left
Right
Slow Left
Slow Right
Slam Brakes
0.0
1.0
Event Parameters
Learning Rate
0.000001
Exploration Rate
0.05
Reward Shift
0
Step
Sample Action Matrix
Sample Action Matrix
Left
Throttle
Steering
0.056
0.021

The Rewards Graph

Rewards Graph
Goals

Presents rewards earned per step, overlaid on statistics about previous run results.

Allows user discovery of the models' ability to attain rewards as training continues, while simultaneously comparing it to previous runs.

Keep track of your model interactions by way of events.

Rewards Graph
Encodings

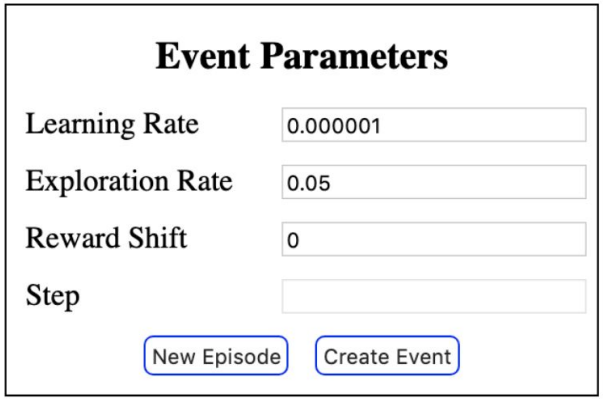
- Y-channel is the step reward.
- X-channel is training step, functioning as time.
- Red line is the current run.
- Background shape is the curve boxplot across all runs, using 25-75 percentile.
  - White is used to provide maximum contrast against the red current line run.
  - Gray stroke is used for the shape to make the edges more clear, without contrasting with the red line.
  - A gray centerline (the 50th percentile) tracks inside the contour, making the median more obvious. All light colors to make the red line clearly visible.
- Event “flag” marks located in x-channel.

Rewards Graph
Interactions

- Click anywhere to ***select*** a step, made clear by the vertical bar.
- Text and a bar displays the step selected.
- Click and drag to ***select*** a range of steps.
  - Or shift+click another point.
- Click an event flag to ***select*** that event, and the event's step.

Event Metadata

Rewards Graph
Goals



Allows tweaking model hyperparameters.

Events will be created, showing up on the rewards graph.

Selecting an event will update the fields with that event’s data.

Creating a new episode will start a fresh run from step 0.

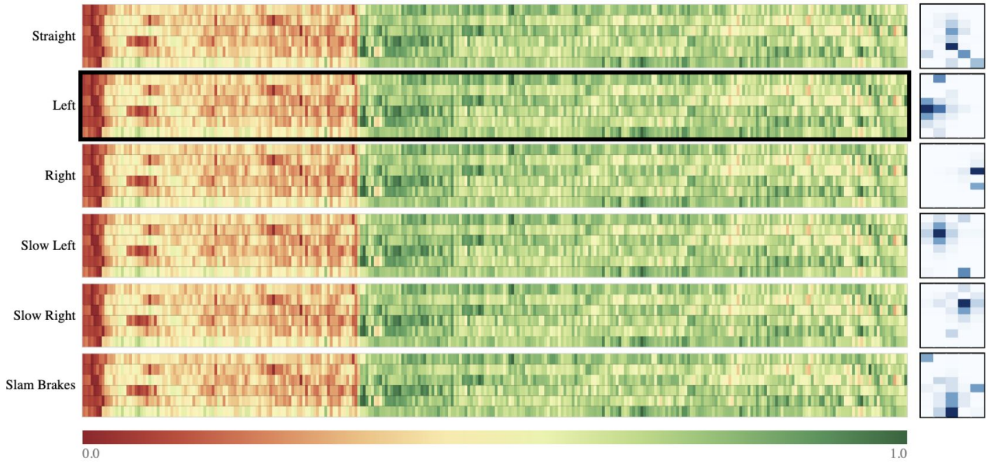
Prediction Spectrograms

Sample Prediction Spectrograms

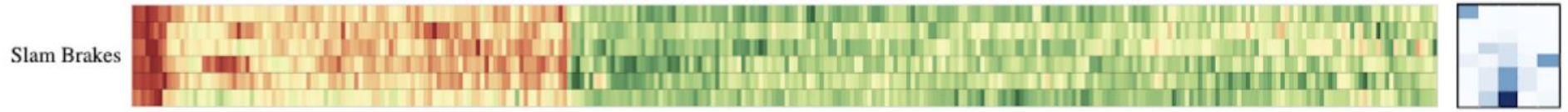
Shows “correctness” of samples across steps (x-channel) and across runs (y-channel). Samples are split into groupings (y-channel).

Correctness (0 = bad, 1 = good) is mapped out onto a red - yellow - green spectrum.

The selected sample has a black border - click to select.



Sample Prediction Spectrograms



Individual sample spectrograms help show behavior prediction quality as training progresses, along with how it functions on a per-run basis.

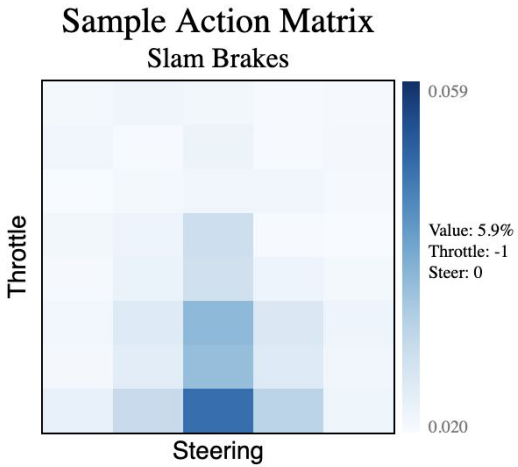
The y-channel encodes the run, with the bottom being the current run.

The x-channel encodes the prediction correctness from red - yellow - green.

d3.**interpolateRdYlGn**(*t*) <>
d3.**schemeRdYlGn**[*k*]

Action Matrix

Action Matrix



Model will produce a softmax output across 40 actions.

Each action index consists of a throttle and steering value.

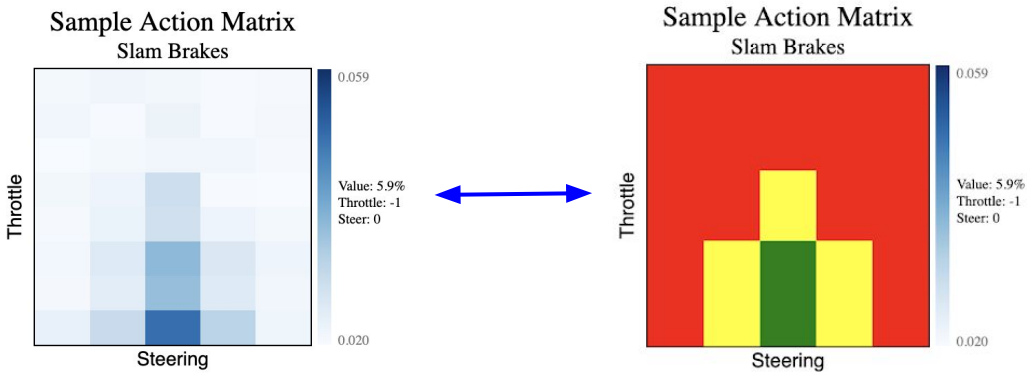
Throttle is on the y-channel, and steering on the x-channel.

Prediction probability encoded by the luminance of each cell.

Sample matrix populated with selected sample and selected step (mock data).

```
# d3.interpolateBlues(t) <>
# d3.schemeBlues[k]
```

Action Matrix



Mouse over a grid node to display the probability value, throttle action (from -1.0 to +1.0), and steering action (from -0.5 to +0.5).

Click to switch colors to encode “assigned label” instead of prediction.

Currently this is all mock data, but for the actual project the labels will be assignable by the users.

Rewards Graph
Action Hover

When hovering over an action within the action matrix, the rewards graph enters the “Action Hover” state.

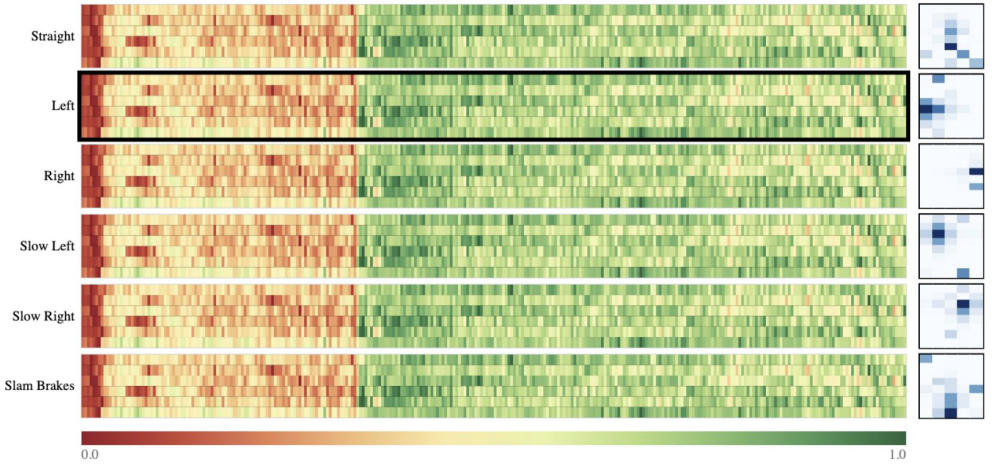
The reward line is colorized based on if each step is attempting to take the selected action.

```
# d3.interpolateBlues(t) <>
# d3.schemeBlues[k]
```

Sample Prediction Spectrograms

Each sample has a small action matrix on the right based on the selected steps.

Allows quick comparisons between samples to check for convergence, and compare actions.



Demo

Questions?