# Baselines Presentation

Grant Fennessy

# Tasks & Data

# Reinforcement Learning

A significant part of RL is constructing the reward functions.

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.

# Data to Work With

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.

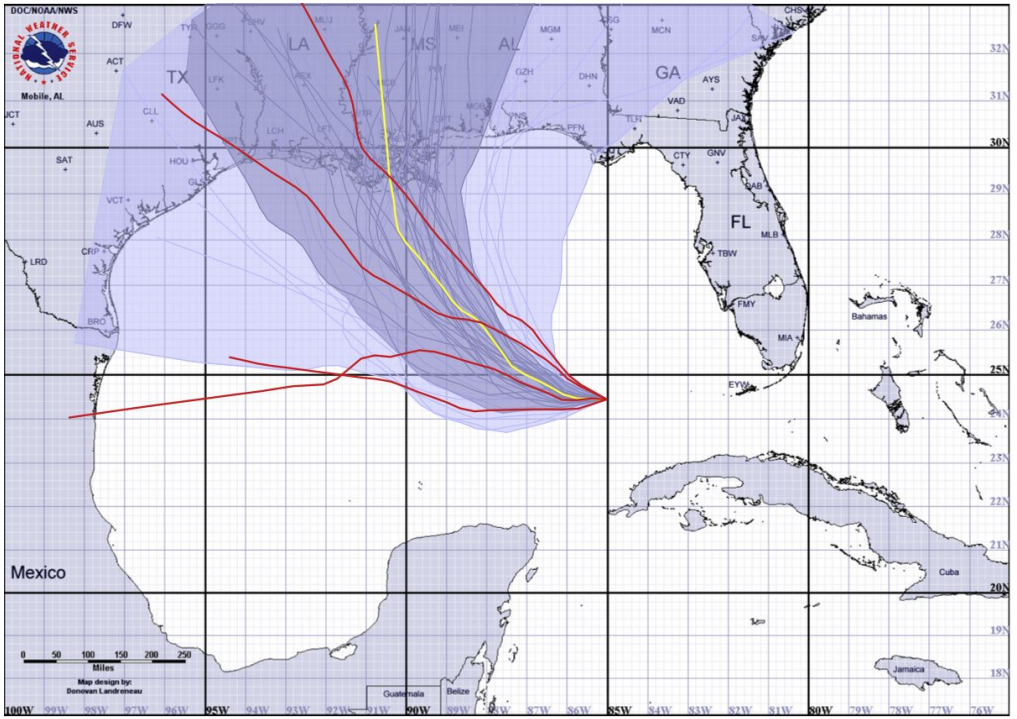
# Related Work

# Curve Boxplots

Too many lines can produce clutter and lower clarity.

Curve Boxplots color an area representing 25%-75% zone.

Can also display outliers, a selected line, or other zones.

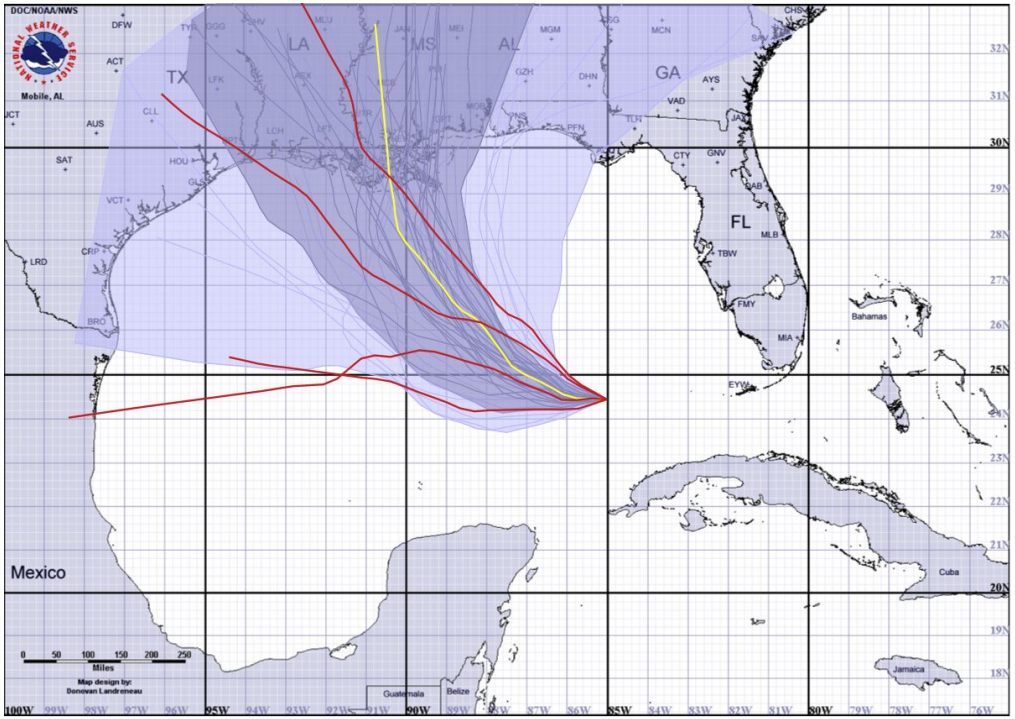



*Curve Boxplot: Generalization of Boxplot for Ensembles of Curves, Mirzargar* et. al.

# Curve Boxplots

The x and y channels represent geographic coordinates.

The lines represent coordinate value changes over time.




*Curve Boxplot: Generalization of Boxplot for Ensembles of Curves, Mirzargar* et. al.

# DeepTracker

The y-channel encodes a validation sample image class.

The x-channel encodes step during training.

Color of grid cell encodes mean validation accuracy across all images from the sample a the step.

Can expand a spectrogram to show all images within the class on y-channel.

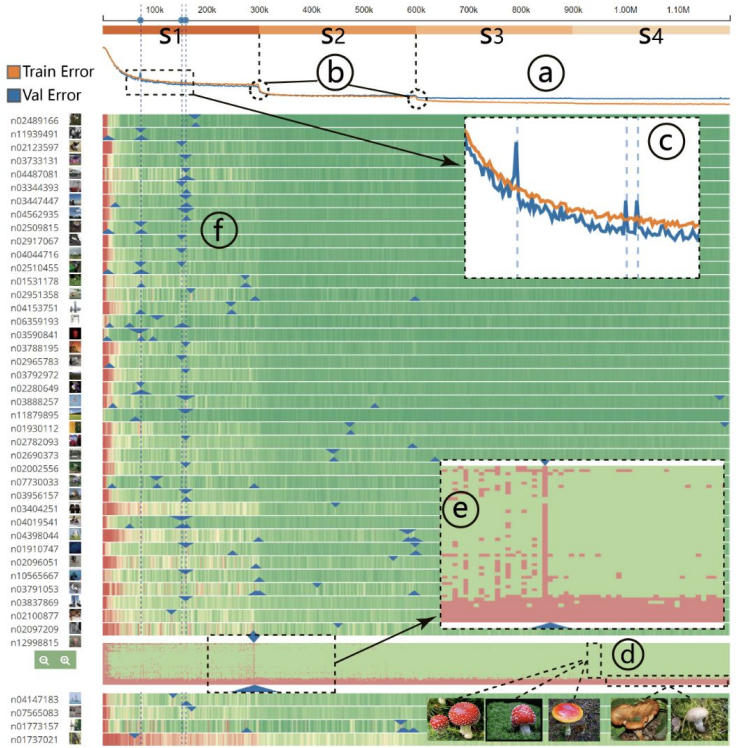



Fig. 6. Overview of validation classes. (a) Two curves show the overall training/validation error rate. (b) Two turning points align well the boundary of stage $s2$. (c) Three peaks appear in the stage $s1$ and align well with (f) the detected anomaly iterations. (d) Two types of mushroom images have different behaviors in the class. (e) Most images in the class flip at the anomaly iteration.

*DeepTracker: Visualizing the Training Process of Convolutional Neural Networks, Dongyu et. al.*

# Correlation View

X-channel encodes a DNN layer.

Y-channel encodes an image category.

Luminance of grid cells encodes the number of anomaly filters for that layer/category pairing.  Darker cells represent larger values, with lighter cells representing smaller values.


*DeepTracker: Visualizing the Training Process of Convolutional Neural Networks, Dongyu et. al.*

# Objectives, Encodings, Interactions

# Objectives

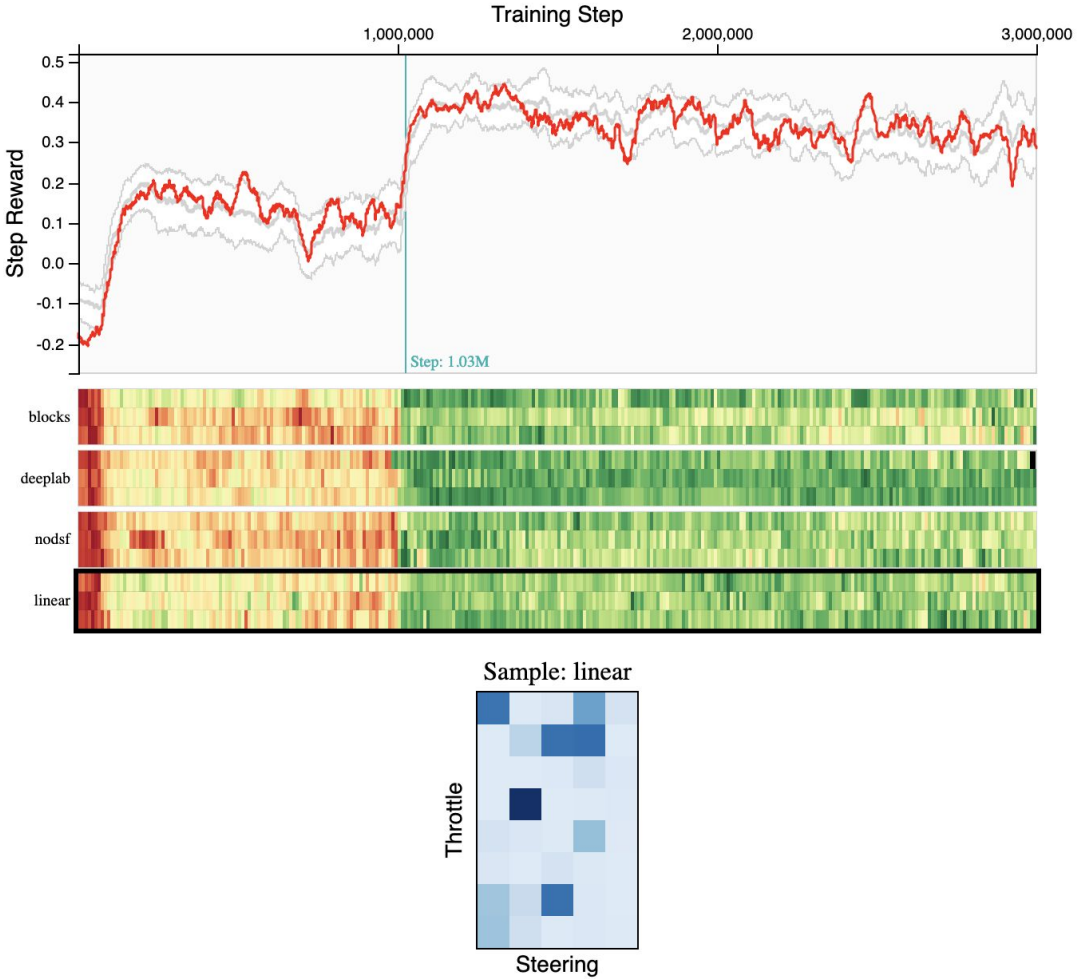
Allow users to see rewards attained by the model as it trains.

Provide easy comparison to past experiment reward values.

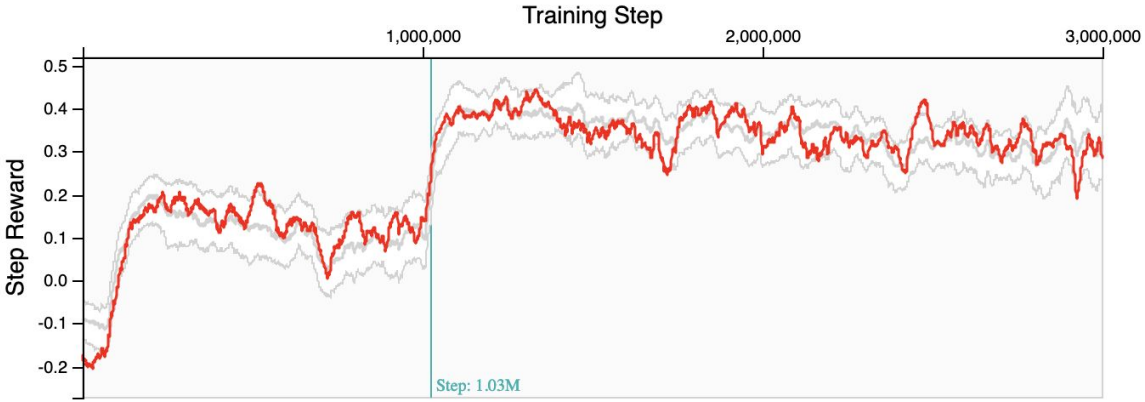
Present the prediction (behavior) quality per sample (an image) over time.

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

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


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: 1.03M
blocks
deeplab
nodsf
linear
Sample: linear
Throttle
Steering

# Rewards Graph




Presents the current run's rewards earned per step, overlaid on a white shape representing statistics about all previous run results.

Y-channel is the step reward (mean over 1,000 steps, and smoothed out with exponentially weighted moving average equation).

X-channel is training step, functioning as time.

Can click anywhere to select a step, made clear by the vertical bar.  Text displays the step selected.

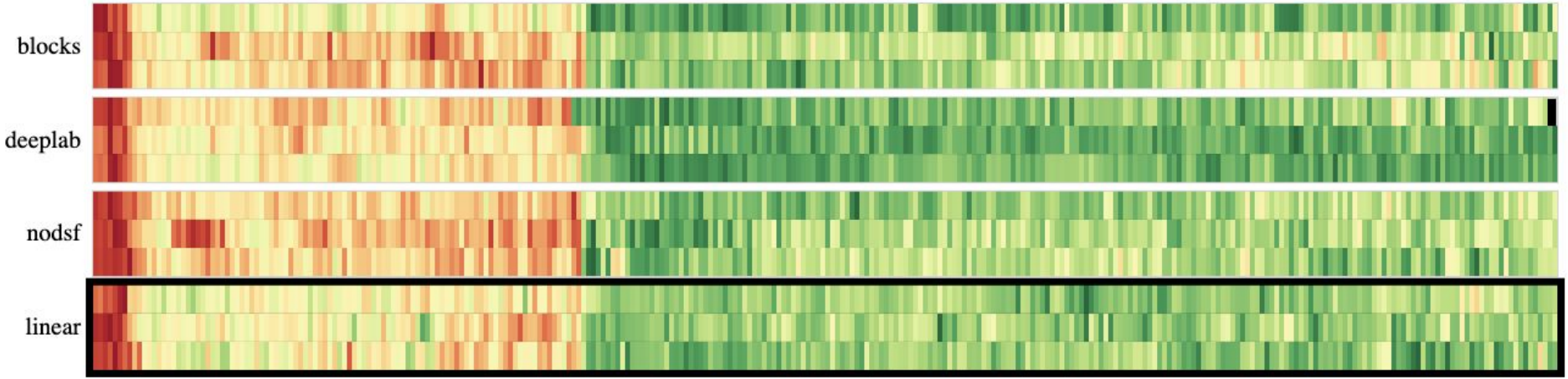
# Rewards Graph


The white 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.

# Sample Prediction Spectrograms

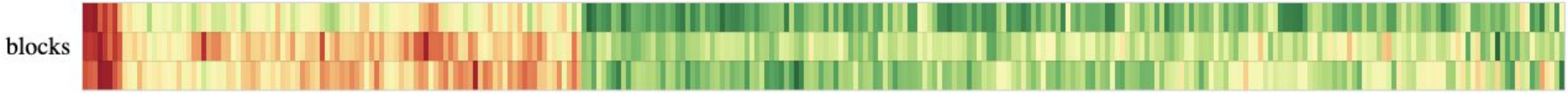



Shows “correctness” of samples across steps (x-channel) and across runs (y-channel). Samples are split into groupings (y-channel).

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



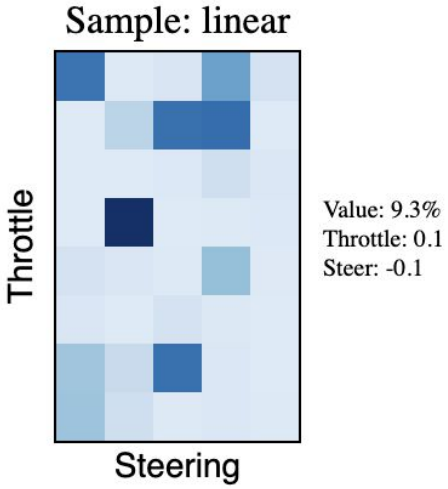

Individual sample spectrograms help show behavior prediction quality as training progresses, along with how it functions on a per-run basis.

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*]

# Predictions Matrix




Model will produce a softmax output across 40 actions.

Each action index consists of a throttle and steering value.

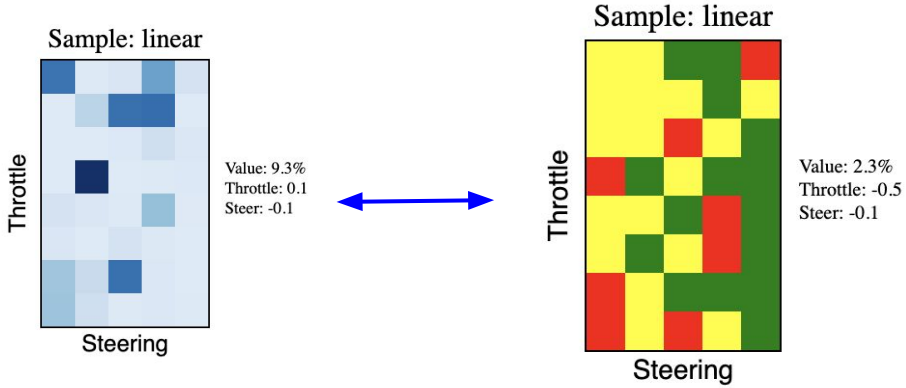
Throttle is on the y-channel, and steering on the x-channel.

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]
```

# Predictions 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.

# Demo

Questions?