Adaptive User Interface Optimisation Using Reinforcement Learning

Team Members:
Apurva Banka (apurvaba)
Kumar Satyam (kumarsat)

Course: CSE 4/546: Reinforcement Learning, Spring 2025
Date: May 1, 2025

Project Description

In the mobile application industry, the design and usability of User Interfaces (UI) are pivotal to app engagement, user retention, and overall success. However, assessing what constitutes a "good" UI and optimizing it remains a complex task, often left to manual A/B testing and designer heuristics.

Our project, **"Adaptive User Interface Optimisation Using Multi-Agent Reinforcement Learning"**, addresses this challenge by using Deep Reinforcement Learning (RL) to predict the quality of mobile app UIs and guide the design of optimized UIs. By learning from 66k+ UIs, their semantic layouts, and metadata from the RICO dataset, our system can automatically infer which UI designs are more successful, based on real-world popularity signals such as Play Store ratings and download counts.

Summary

User interfaces in most apps are static and don't adapt well to individual users' needs, preferences, or interaction habits. This project seeks to change that by using Reinforcement Learning, Multi-Agent Reinforcement Learning to create a dynamic UI that adjusts in real-time based on how users interact with it. The question that we are aiming to answer is:"How can multi-agent reinforcement learning be applied to design a user interface that adapts in real time, improving the user experience based on individual interactions?"

This topic is exciting because it has the potential to revolutionize how we personalize digital experiences. Whether it's improving accessibility, making interfaces more intuitive, or simply optimizing usability, this project has broad applications in making apps smarter and more responsive to user behavior. By using the RICO dataset, which includes a wide variety of real

mobile app UIs, the project aims to bring a data-driven approach to solving the problem of static interfaces and improve how UIs adapt to users in a meaningful way.

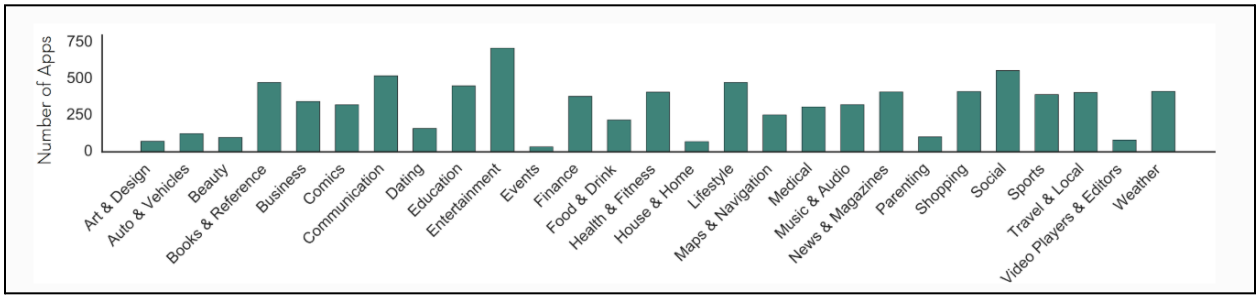
Objective

The goal is to develop an adaptive UI that adjusts based on real-time user behavior. To achieve this, the first step will be to analyze the RICO dataset, identifying key UI elements that can be adapted based on user interactions. This will provide the foundational understanding of what elements need to change and how to implement those changes dynamically.

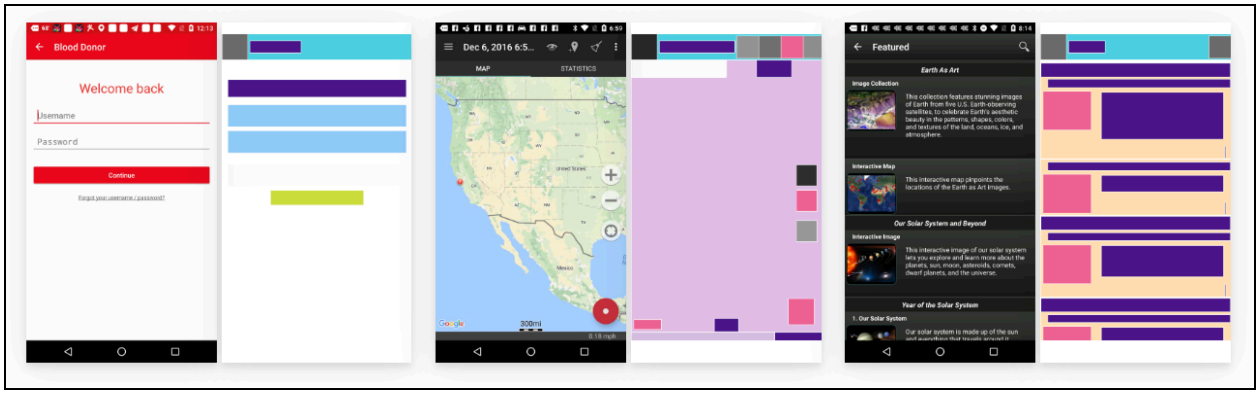
Background

The RICO dataset is the most comprehensive public dataset for mobile UI analysis, offering:

- UI screenshots and view hierarchies.
- Semantic annotations of components, text buttons, and icons.
- App metadata including ratings and downloads.
- Gesture traces and interaction flows.



A few examples of UI screens and their semantic annotations are shown below.



Traditional approaches to UI evaluation involve static heuristics or human studies. Our approach leverages **Dueling Deep Q-Networks (Dueling DQNs)** to learn from data directly and generalize across unseen UIs. Prior works like Deka et al. (RICO) and Google's App UI similarity learning via autoencoders inspired our use of deep models for structural and semantic understanding.

Environment

We built a custom environment inspired by OpenAI Gym for training our RL agents:

- **Observation space**: Combines 128×128 CNN-encoded screenshots + 64D layout features + 318D semantic vectors from RGB-mapped masks.
- **Action space**: Single dummy action (since we only predict rewards).
- **Reward**: Ground truth app success score, computed as a weighted sum:

$$\text{app\_score} = 0.7 \cdot \text{normalized rating} + 0.3 \cdot \log(\text{normalized downloads})$$

The environment loads cached features from `.npz` files during training to reduce I/O overhead by 10x.

Dataset Summary

We used the **RICO dataset**, covering:

➔ **66k+ UI Screens**

➔ **9.3k Unique Apps**

➔ **27 App Categories**, including:
`Books & Reference, Comics, Health & Fitness, Social, Entertainment, ...`

➔ **3 Semantic Channels**:

- 24 UI components (from `component_legend.json`)
- 197 text button classes (from `textButton_legend.json`)
- 97 icon classes (from `icon_legend.json`)

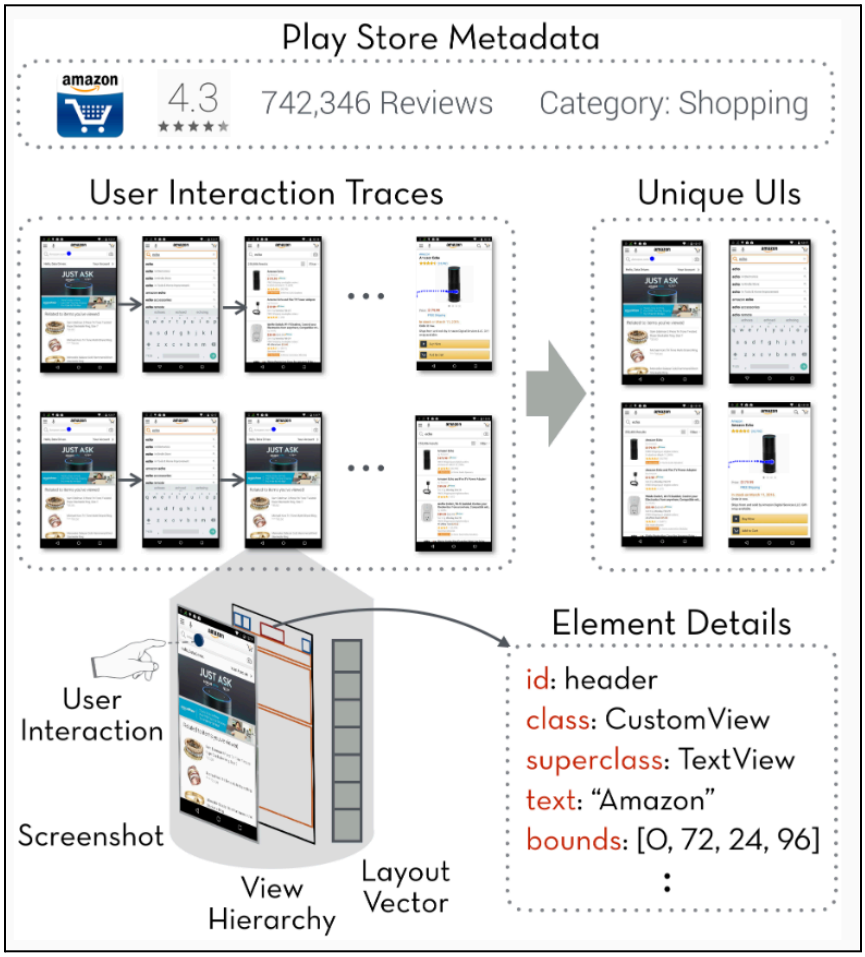
➔ **App metadata (app_details.csv)**:

- App name, Play Store category, average rating, number of downloads.

➔ **UI metadata (ui_details.csv)**:

- UI number, app package, interaction trace, and position in trace.

Semantic masks were processed to extract RGB counts for each class and normalized into frequency vectors.



Methodology

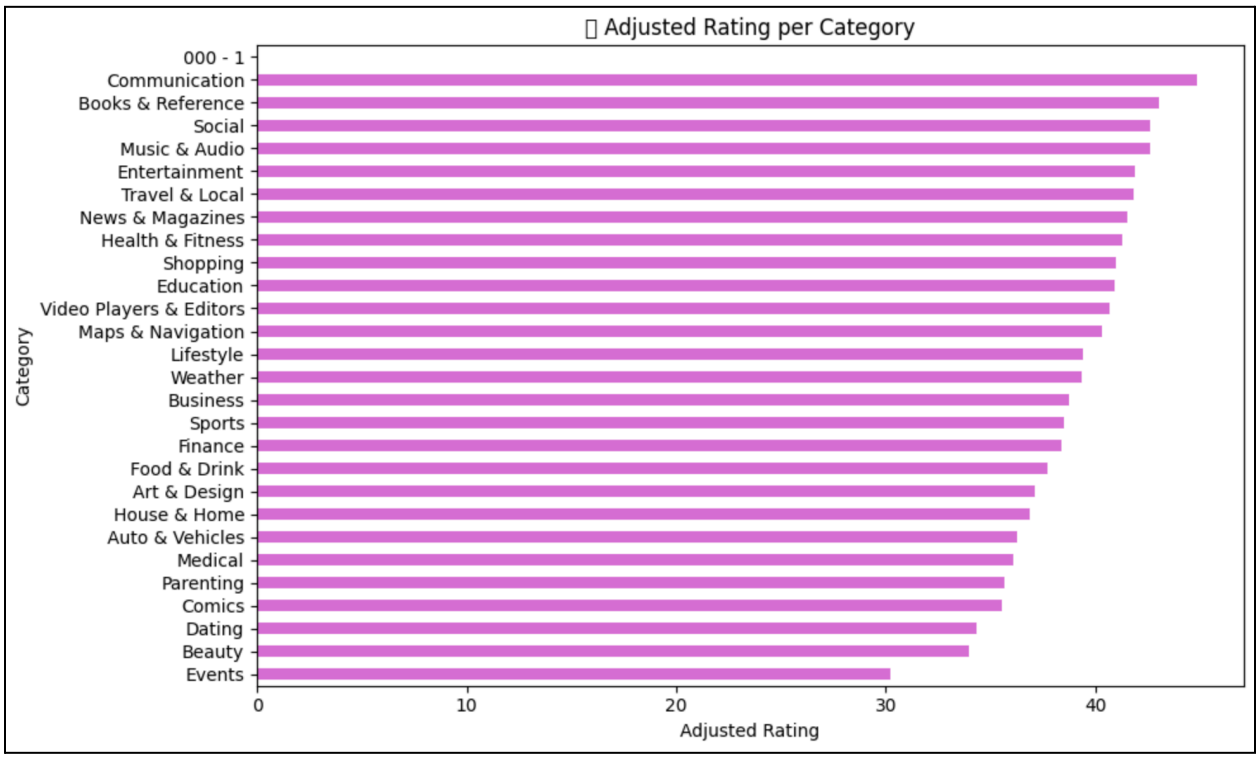
a. Preprocessing Pipeline

1. Parse view hierarchy JSON to extract layout vector: `[total_nodes, visible_nodes, clickable_nodes]`

2. Load semantic mask image, map pixel RGB to semantic class using all three legends.

3. Concatenate screenshot, layout, and semantic vectors.

4. Save as `.npz` file for efficient access during training.



We predicted a normalised score for all the categories of apps so that we can use this in our reward structure.

As we can see above, communication has a higher score. So, we are giving slightly higher rewards based on the above visualised graph.

b. Model Architecture

We adopted the **Dueling Deep Q-Network (DQN)** architecture with two branches:

```
class DuelDQN(nn.Module):
     def __init__(self, n_actions, meta_dim):
        super().__init__()
        self.cnn = resnet34(weights="DEFAULT")
        self.cnn.fc = Linear(512, 128)         # Screenshot features
        self.meta = Sequential(Linear(meta_dim, 64), ReLU())  # Layout + Semantic
        self.adv = Sequential(Linear(192, 128), ReLU(), Linear(128, n_actions))
        self.val = Sequential(Linear(192, 128), ReLU(), Linear(128, 1))

    def forward(self, img, meta):
        f_img = self.cnn(img)
        f_meta = self.meta(meta)
        x = torch.cat([f_img, f_meta], dim=1)
        return self.val(x) + self.adv(x) - self.adv(x).mean(dim=1, keepdim=True)
```

c. Training Strategy

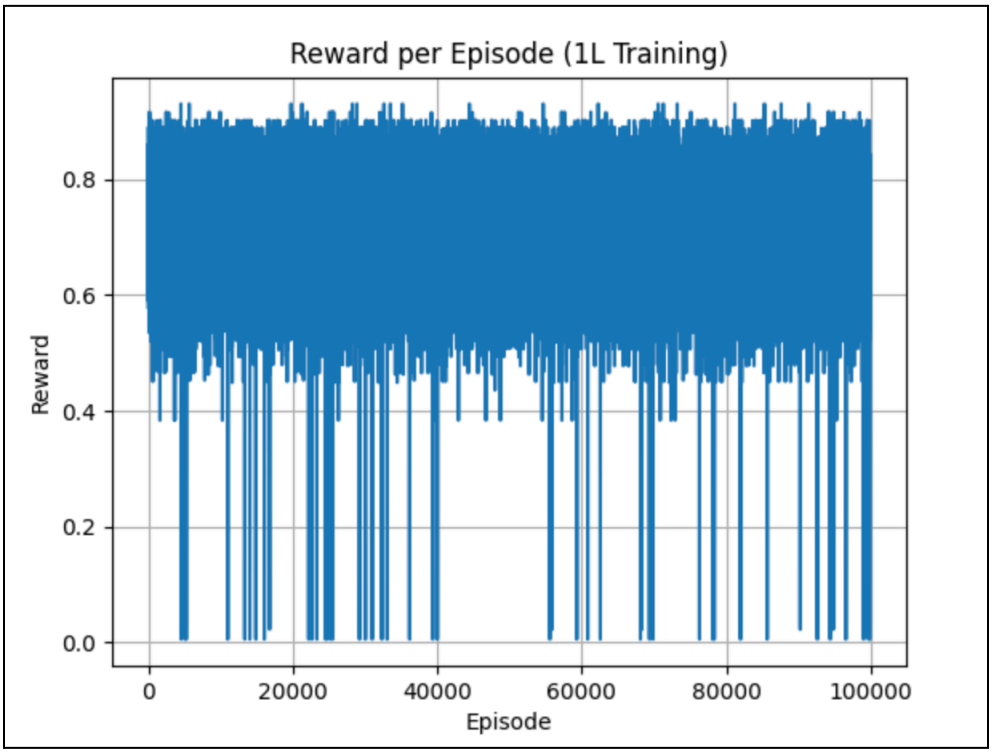
- **Buffer**: Uniform experience replay (`deque` with size 10,000)
- **Loss**: Mean Squared Error (MSE) between predicted and ground truth app scores
- **Batch size**: 32
- **Episodes**: 100,000+
- **Optimizer**: Adam (`lr=1e-4`)
- **Target network update**: Every 1000 episodes

We considered prioritized experience replay, but opted for uniform sampling due to hardware constraints (NVIDIA 940MX GPU with 8GB RAM). Due to the size of the dataset (~50GB) we could not leverage CCR for this project.

Results

a. Training Curve

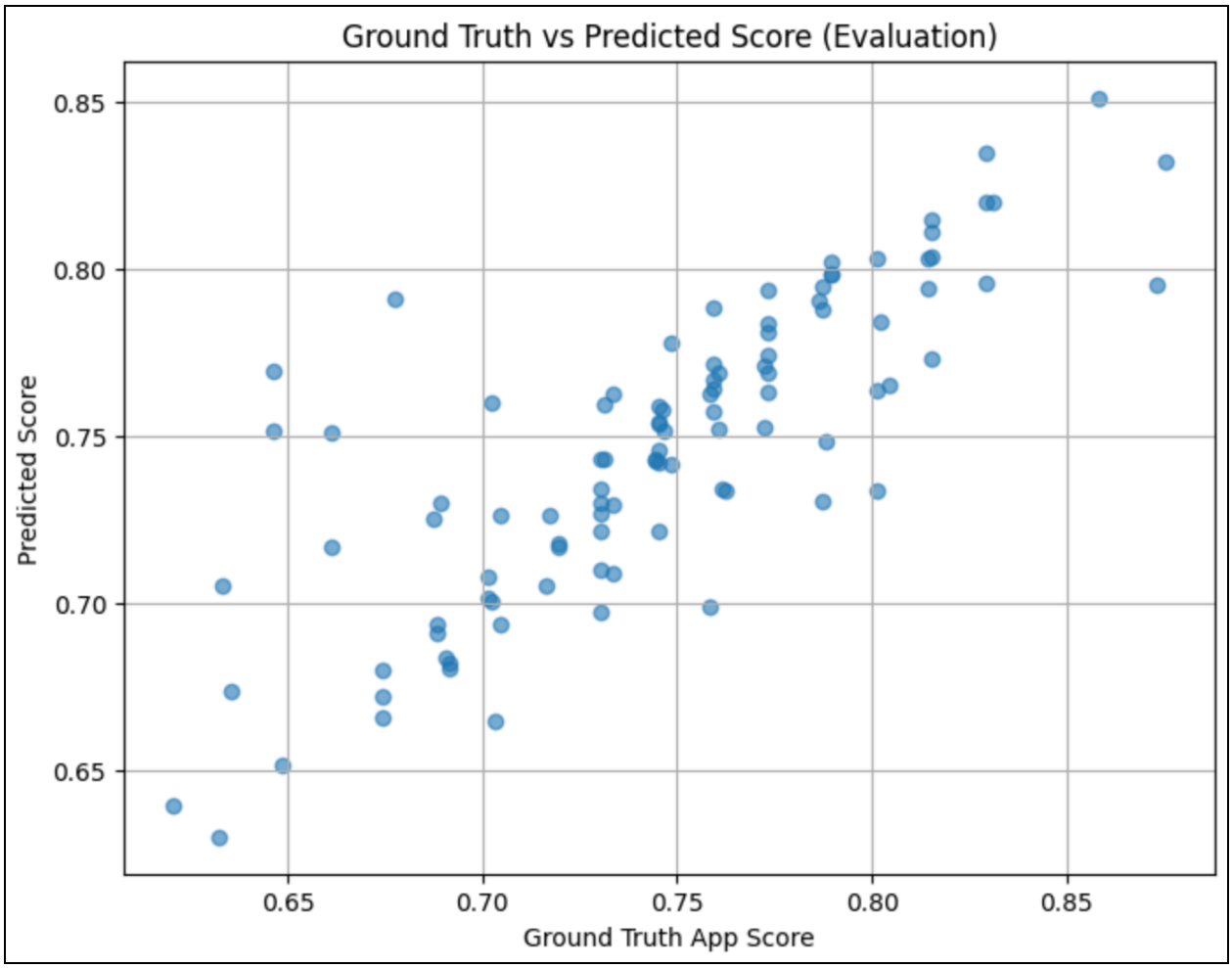
Reward improved significantly after caching optimization and training over 100K episodes:



b. Evaluation: Ground Truth vs Prediction

Using a held-out set of 100 UIs, we plotted the predicted vs ground truth scores:

- **Pearson correlation**: `0.8063`
- **Observation**: Model generalizes well across different app categories and semantic compositions.



Demo

We showcased a notebook-based demo that:

- Loads a given UI screenshot, layout JSON, and semantic PNG.
- Predicts the app score using the trained Dueling DQN model.
- Displays attention on UI components, highlighting high-scoring vs low-scoring screens.

```
⭐ Predicted UI 2661 Score: 0.7341
```

```
9355                    com.madieo.imovie          2.1
```

The predicted UI score is for “**com.madieo.imovie**”. As we can see the UI score for the above is less. This is verified by the rating of the app on PlayStore.

Key Observations & Takeaways

★ Semantic annotations significantly improved prediction quality.

★ Caching `.npz` features made training 10x faster and more memory efficient.

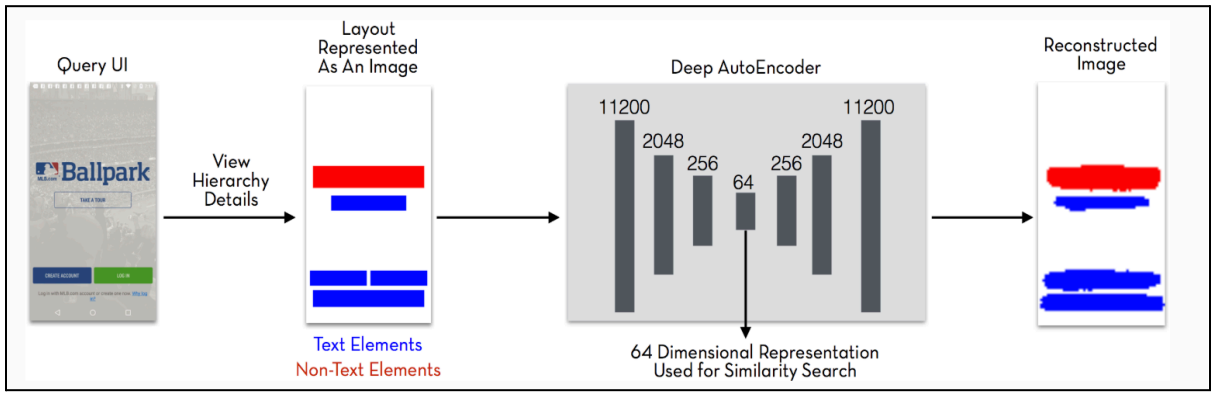
★ The model correctly predicted higher scores for apps in top categories (e.g., Travel, Finance).

★ Lightweight experience replay was sufficient; prioritized replay can be explored further.

Future Work

- Integrate **multi-agent reinforcement learning** to simulate multiple UI flows and user strategies.
- Incorporate **gesture trace** data to model navigation behavior.
- Train a **semantic segmentation U-Net** to predict annotation masks from screenshots directly.
- Convert reward regression to **UI ranking formulation** via pairwise loss.
- Leverage MCTS for training the model. Implement the model using the below research work.

  https://ranjithakumar.net/resources/rico.pdf



Real-world RL Application

Our project addresses a real-world challenge: automatically optimizing mobile user interfaces (UIs) based on real-world usability metrics such as app ratings and download counts. These metrics reflect user satisfaction and app success, making them ideal reward signals for reinforcement learning.

We use the RICO dataset, which includes:

- 66k+ UI screenshots with corresponding view hierarchies and semantic annotations
- 24 component types, 197 text button classes, and 97 icon classes
- Real-world app metadata (app_details.csv) from the Google Play Store: ratings, downloads, categories
- UI mappings (ui_details.csv) linking each screen to its app and interaction trace

This allows us to compute a reward function based on app performance and train a Dueling DQN agent that learns to predict UI quality from:

- Visual features (via ResNet-34)
- Semantic frequency vectors (from annotation masks)
- Structural layout features (from JSON)

By learning from real user behavior, our model supports data-driven UI assessment and optimization, offering practical value to app developers and designers.

Contribution Summary

Member	Contributions
Apurva Banka	Data preprocessing, semantic vector mapping, ResNet integration, report writing
Kumar Satyam	RL environment design, Dueling DQN implementation, evaluation, visualization

References

Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., & Kumar, R. (2017).
Rico: A mobile app dataset for building data-driven design applications.
In Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (pp. 845–854).
https://doi.org/10.1145/3126594.3126651

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015).
Human-level control through deep reinforcement learning.
Nature, 518(7540), 529–533.
https://doi.org/10.1038/nature14236

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & de Freitas, N. (2016).
Dueling network architectures for deep reinforcement learning.
In Proceedings of the 33rd International Conference on Machine Learning (pp. 1995–2003).
https://proceedings.mlr.press/v48/wangf16.html

Lin, L. J. (1992).
Self-improving reactive agents based on reinforcement learning, planning and teaching.
Machine Learning, 8(3–4), 293–321.
https://doi.org/10.1007/BF00992699

Thank You

We sincerely thank the CSE 4/546 course staff for the guidance and the RICO dataset authors for enabling impactful mobile UI research.

For questions or demo, please contact:

- apurvaba@buffalo.edu
- kumarsat@buffalo.edu