Example: SuperTuxKart

- **Locations of karts, pickups, etc. in 3D space**
- **Screen Image**
- **State**
- **Reward**
- **Observation**
- **Agent**
- **SuperTuxKart**
- **Action**
- **Finished?**
- **Human player or network**
- **Acceleration, steering**
Formally: (Partially Observable) Markov Decision Process

- Partially Observable State space: $s \in S$
- Action space: $a \in A$
- Transition probabilities: $P(s_{i+1} | s_i, a_i)$
- Observation space: $o \in \Omega$
- Observation probabilities: $O(o_{i+1} | s_{i+1}, a_i)$
- Reward: $r(s_i, a_i)$
- Environment
- Network
- Action
- Observation
- Reward

Thoughts:
- State

Diagram:
- Markov Decision Process
- Observation
- Reward
- Environment
- Network
- Action
- Observation
- Reward
Optimize rewards *in the long term*

Rollout

**Trajectory** \( \tau = s_0, a_0, s_1, a_1, \ldots, s_n, a_n \)

**Return** \( R(\tau) = \sum_{i=0}^{n} \gamma^i r(s_i, a_i) \) “Discounted sum”

**Discount factor**
Challenges

Network Environment

Network Environment

Network Environment

Discounted Sum

Reward

Reward

Reward

Return

Long-term dependencies

Sparse rewards

Not differentiable!
Imitation Learning

- Gather example trajectories from an expert

\[ \tau^E = s_0, a_0^E, s_1, a_1^E, \ldots, s_n, a_n^E \]

- Use supervised learning

\[ \begin{align*}
    s_0 &\rightarrow a_0^E \\
    s_1 &\rightarrow a_1^E \\
    \vdots \\
    s_n &\rightarrow a_n^E
\end{align*} \]
Imitation Learning: Problems

- Expert trajectories can be hard to gather
- Expert limits performance
- Distribution shift
Imitation Learning Tips and Tricks

- Pre-training (for image inputs)
- Data Augmentation
"On-policy"