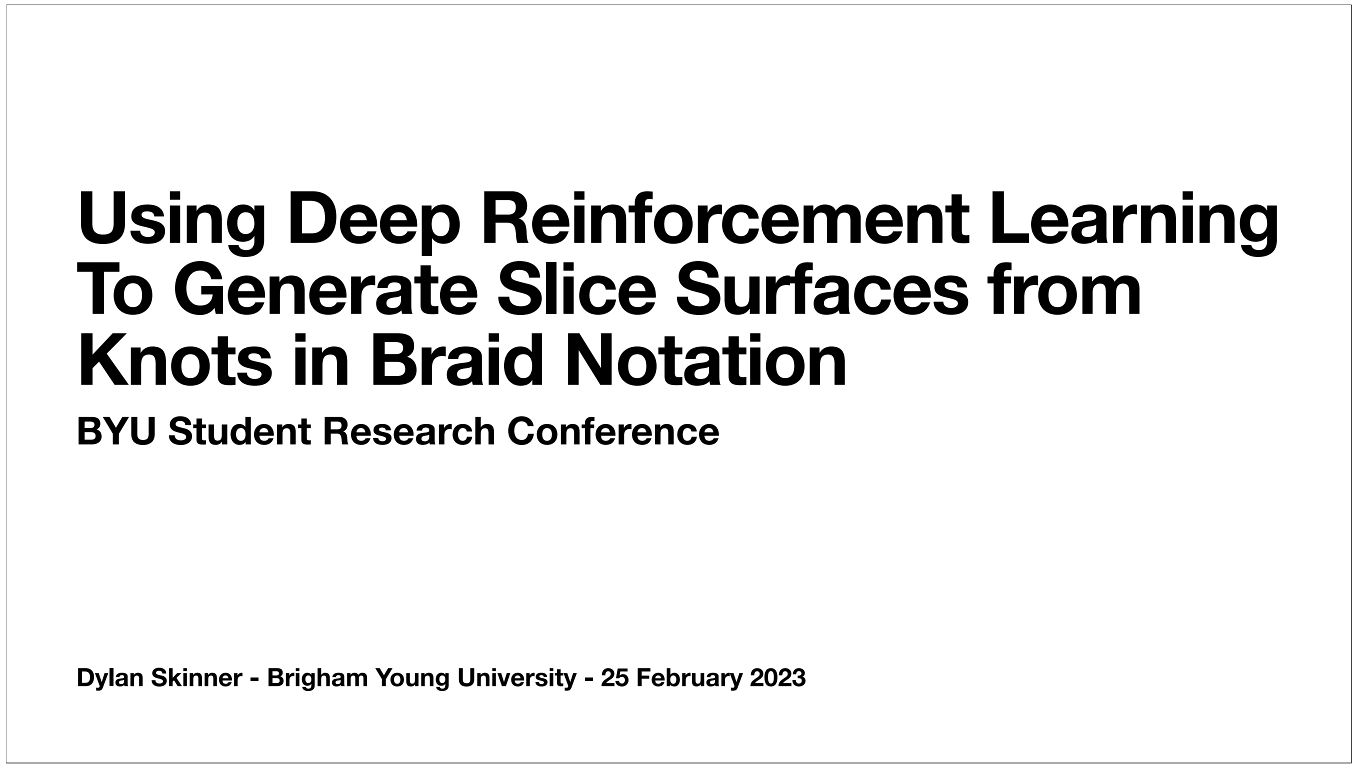


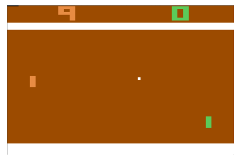
Using Deep Reinforcement Learning To Generate Slice Surfaces from Knots in Braid Notation

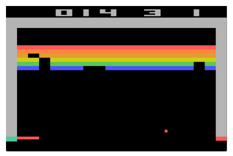
BYU Student Research Conference

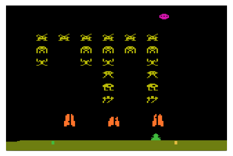
Dylan Skinner - Brigham Young University - 25 February 2023

Fancy way of saying I used deep learning to try and find a specific invariant of a knot.

What is Reinforcement Learning?
Introduction



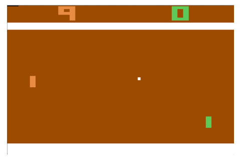


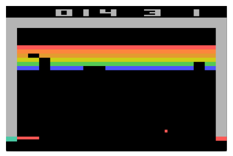


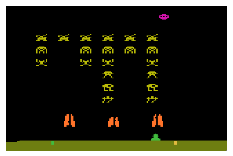
What is Reinforcement Learning?

Introduction

- Agent is placed in an environment



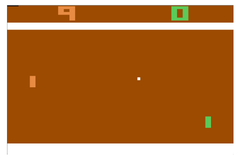


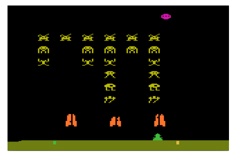


What is Reinforcement Learning?

Introduction

- Agent is placed in an environment
- Agent interacts with the environment through a set of actions

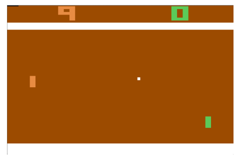


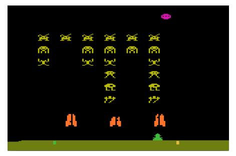


What is Reinforcement Learning?

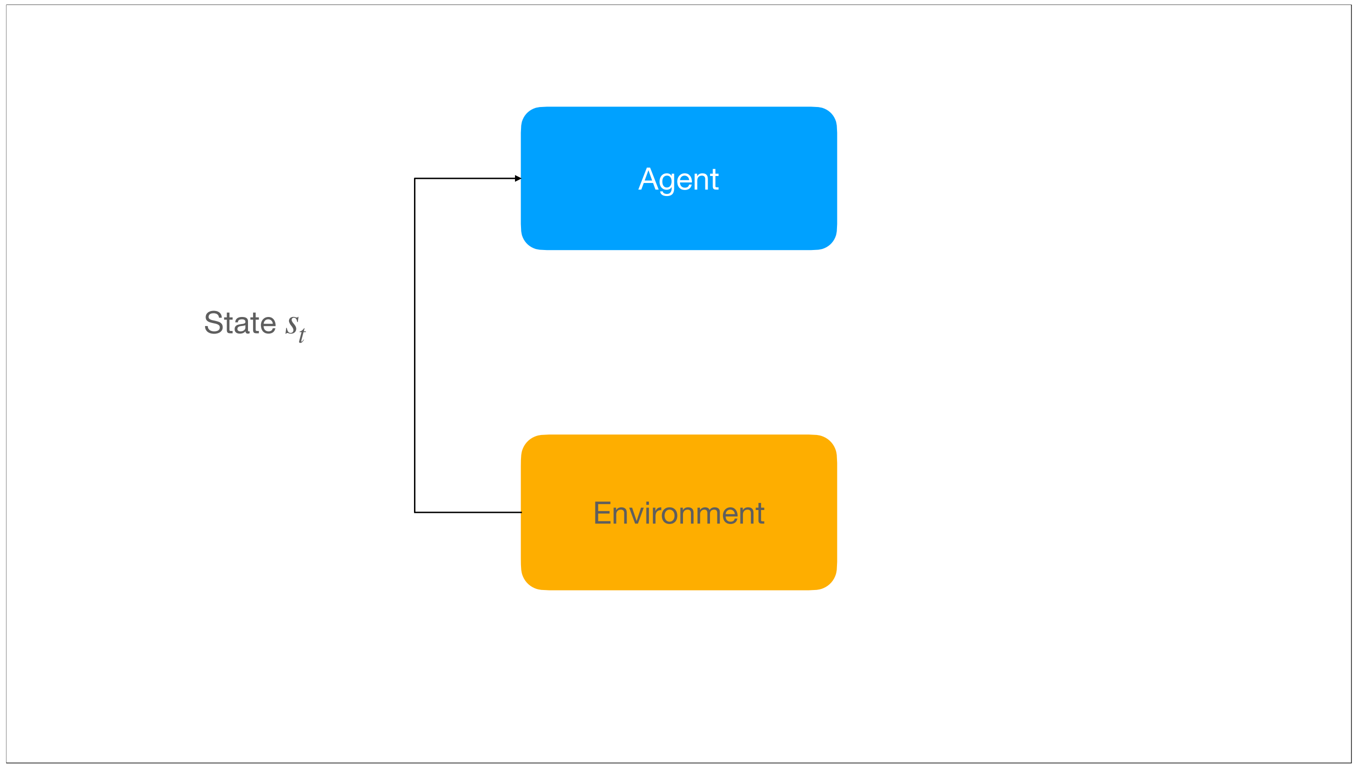
Introduction

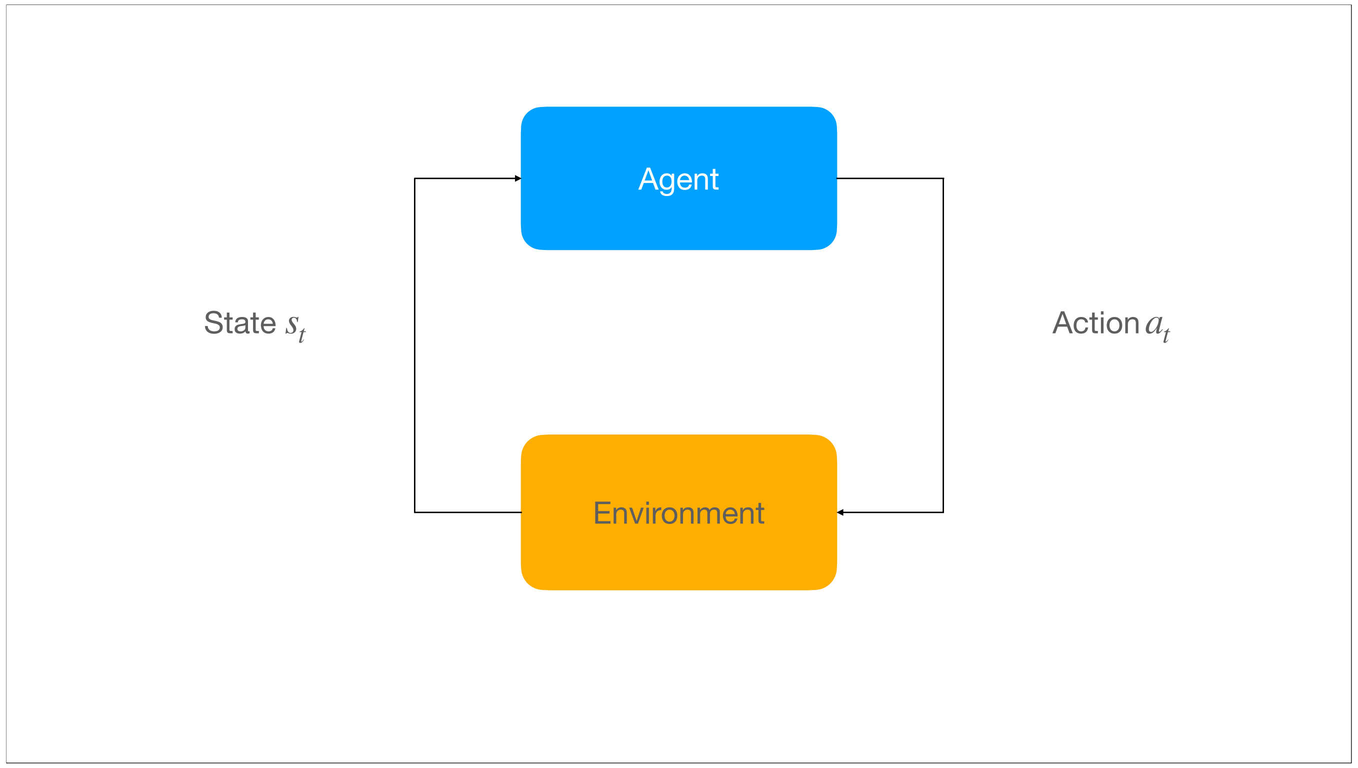
- Agent is placed in an environment
- Agent interacts with the environment through a set of actions
- Agent chooses its actions to maximize a reward (goal)



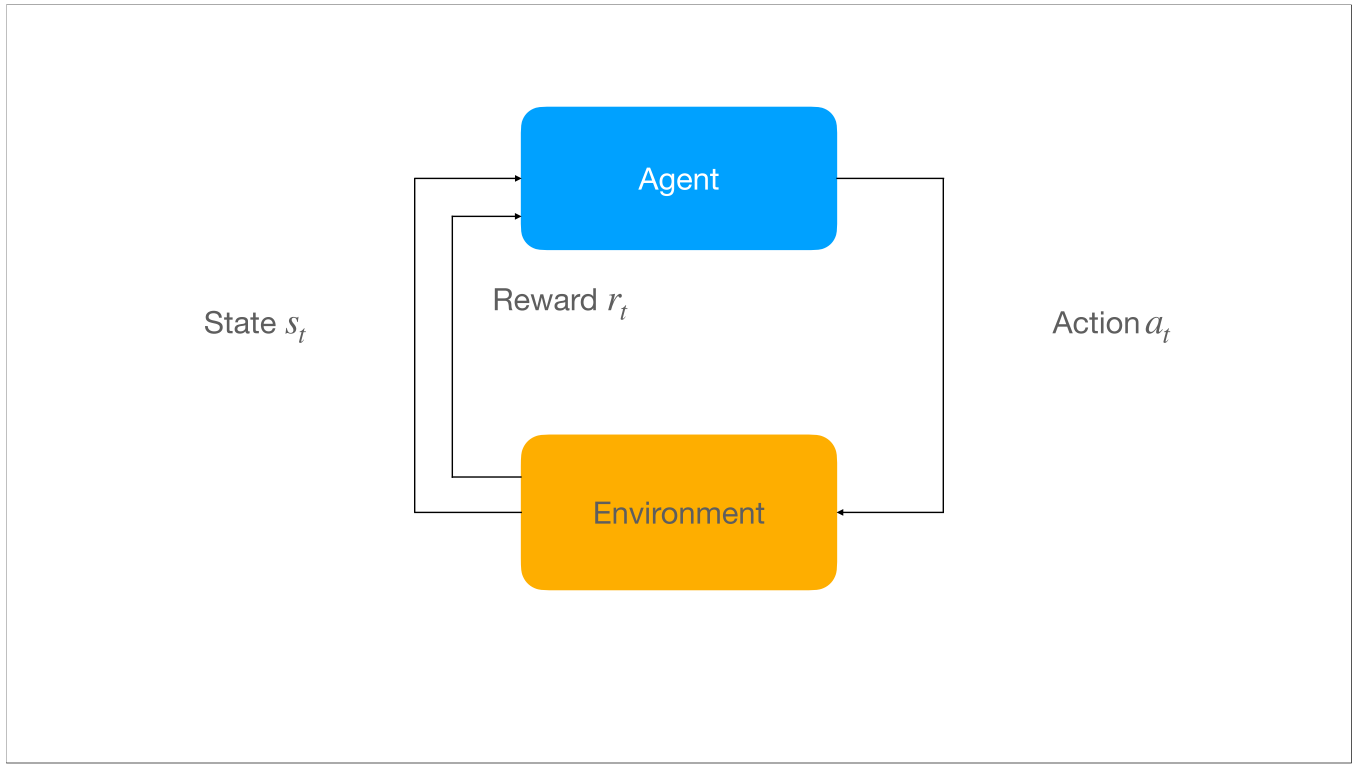


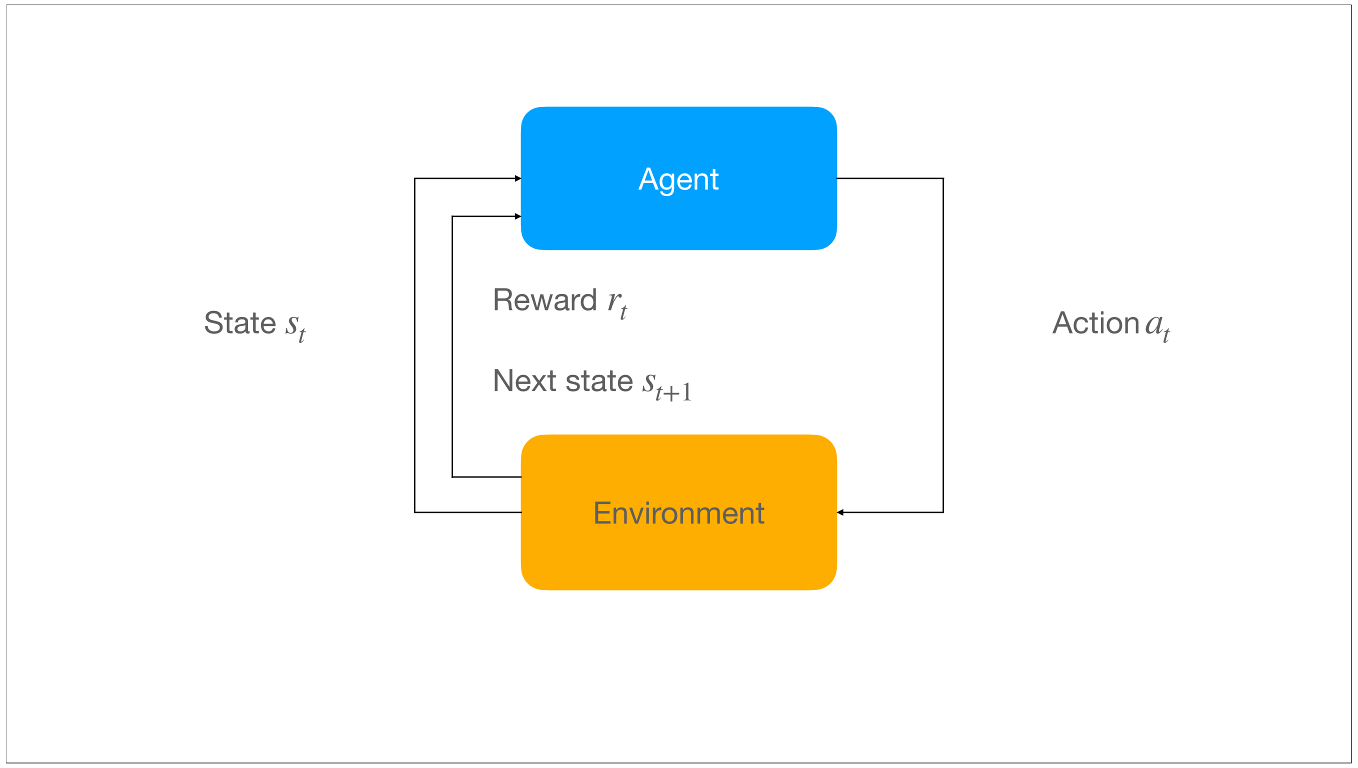
Agent
Environment


Agent
State s_t
Environment



- The action changes the state.


Agent
Reward r_t
State s_t
Action a_t
Environment



The way the agent decides which action to take is influenced by the algorithm the agent is using to evaluate the world it lives in. The algorithm that we used is Proximal Policy Optimization (PPO).

Proximal Policy Optimization (PPO)

Introduction

Proximal Policy Optimization (PPO)

Introduction

- Algorithm developed by OpenAI in 2017

Proximal Policy Optimization (PPO)

Introduction

- Algorithm developed by OpenAI in 2017
- Seeks a balance between ease of implementation, sample complexity, and ease of tuning

Proximal Policy Optimization (PPO)
Introduction

- Algorithm developed by OpenAI in 2017
- Seeks a balance between ease of implementation, sample complexity, and ease of tuning
- Accomplished by computing update at each step to minimize cost function and deviate only slightly from current policy

Proximal Policy Optimization (PPO)
Introduction

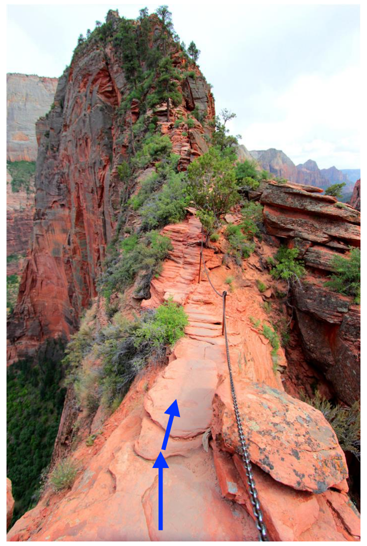
- Algorithm developed by OpenAI in 2017
- Seeks a balance between ease of implementation, sample complexity, and ease of tuning
- Accomplished by computing update at each step to minimize cost function and deviate only slightly from current policy
- In order for this to work, the algorithm uses two separate policy networks

Proximal Policy Optimization (PPO)

Visualized

Proximal Policy Optimization (PPO)

Visualized



Proximal Policy Optimization (PPO)

Visualized

Proximal Policy Optimization (PPO)

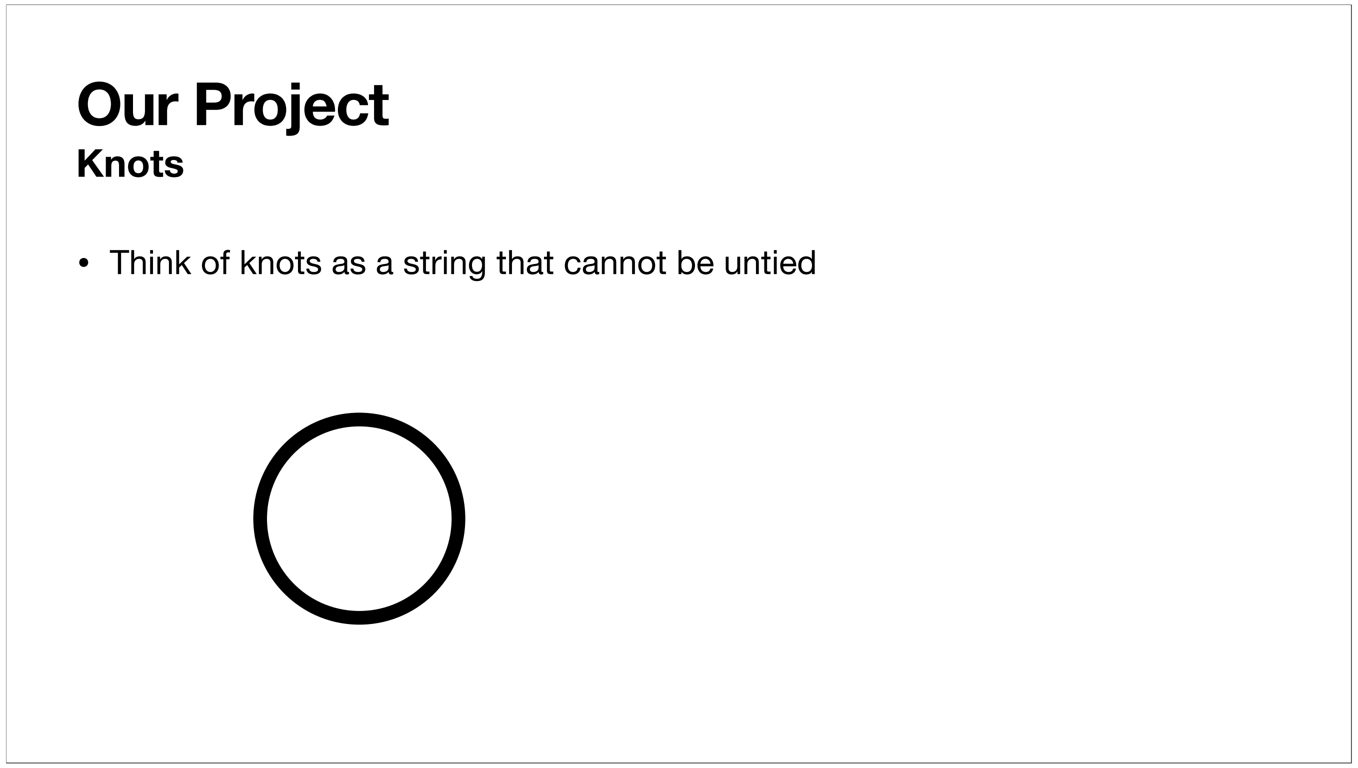
Visualized

Proximal Policy Optimization (PPO)

Visualized

Proximal Policy Optimization (PPO)

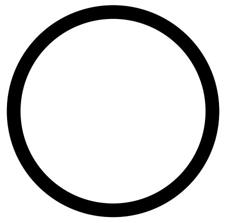
Visualized



Our Project

Knots

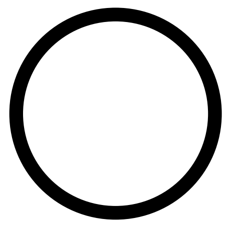
- Think of knots as a string that cannot be untied

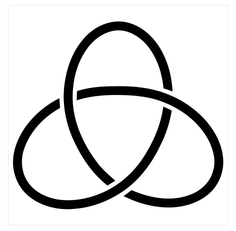


Our Project

Knots

- Think of knots as a string that cannot be untied

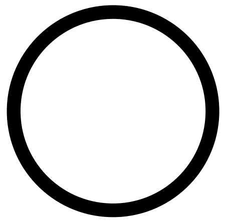
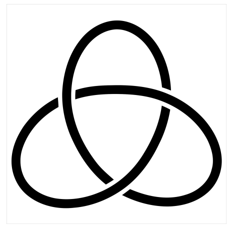
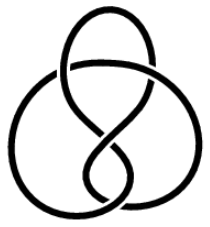


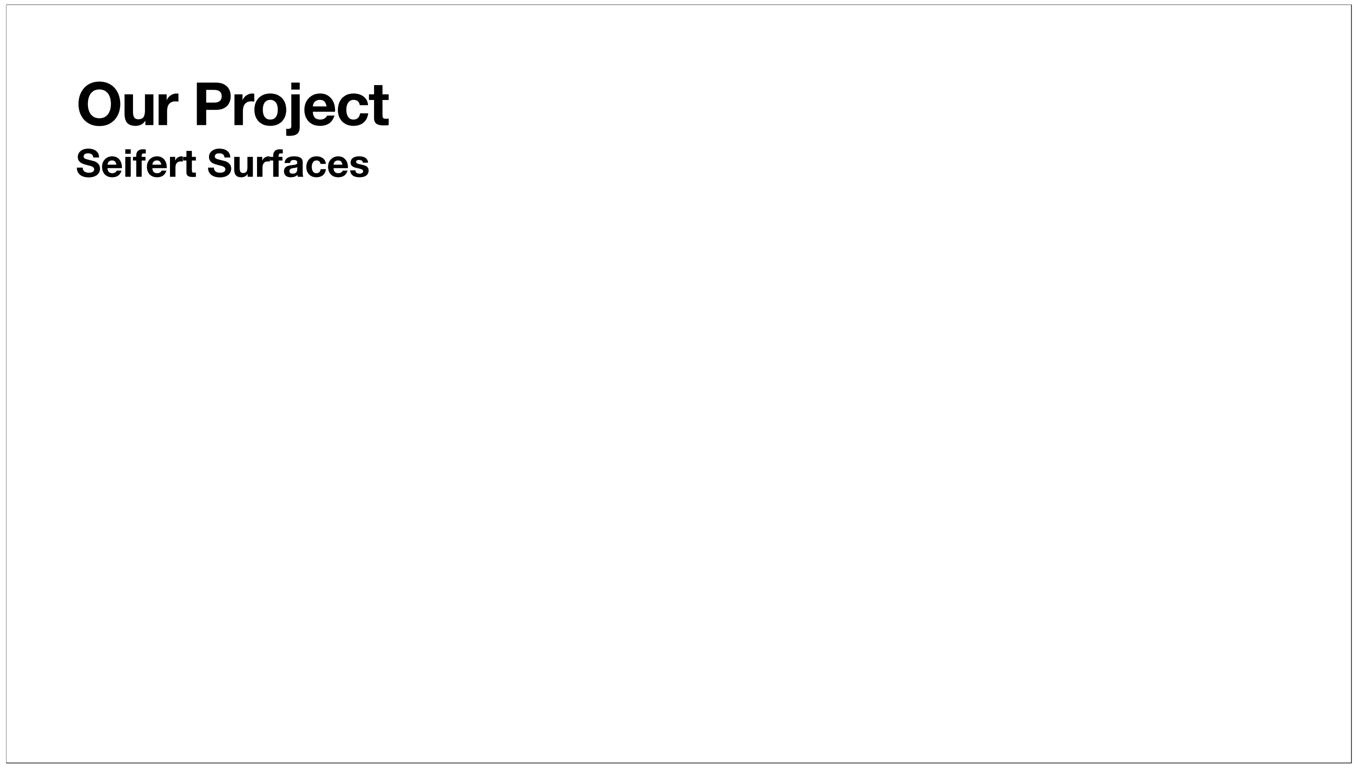


Our Project

Knots

- Think of knots as a string that cannot be untied



Our Project

Seifert Surfaces

- Trying to find a surface that has the least number of holes in it (because it is easy to find one with a large number)
- To find a Seifert surface, simply take the knot, draw a surface connecting the lines, then count the number of holes you see (pretty over simplified).

Our Project

Seifert Surfaces

- Trying to find minimal genus slice surfaces

Our Project

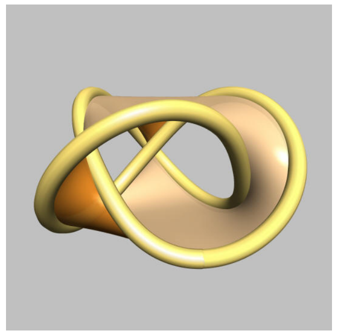
Seifert Surfaces

- Trying to find minimal genus slice surfaces
- Seifert surfaces are orientable surfaces bounded by the knot

Our Project

Seifert Surfaces

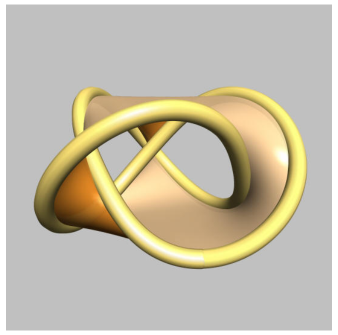
- Trying to find minimal genus slice surfaces
- Seifert surfaces are orientable surfaces bounded by the knot

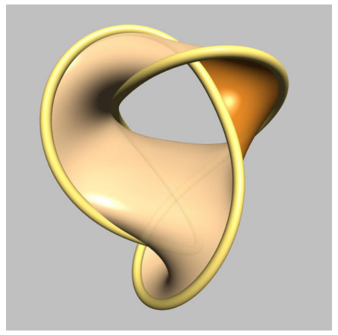


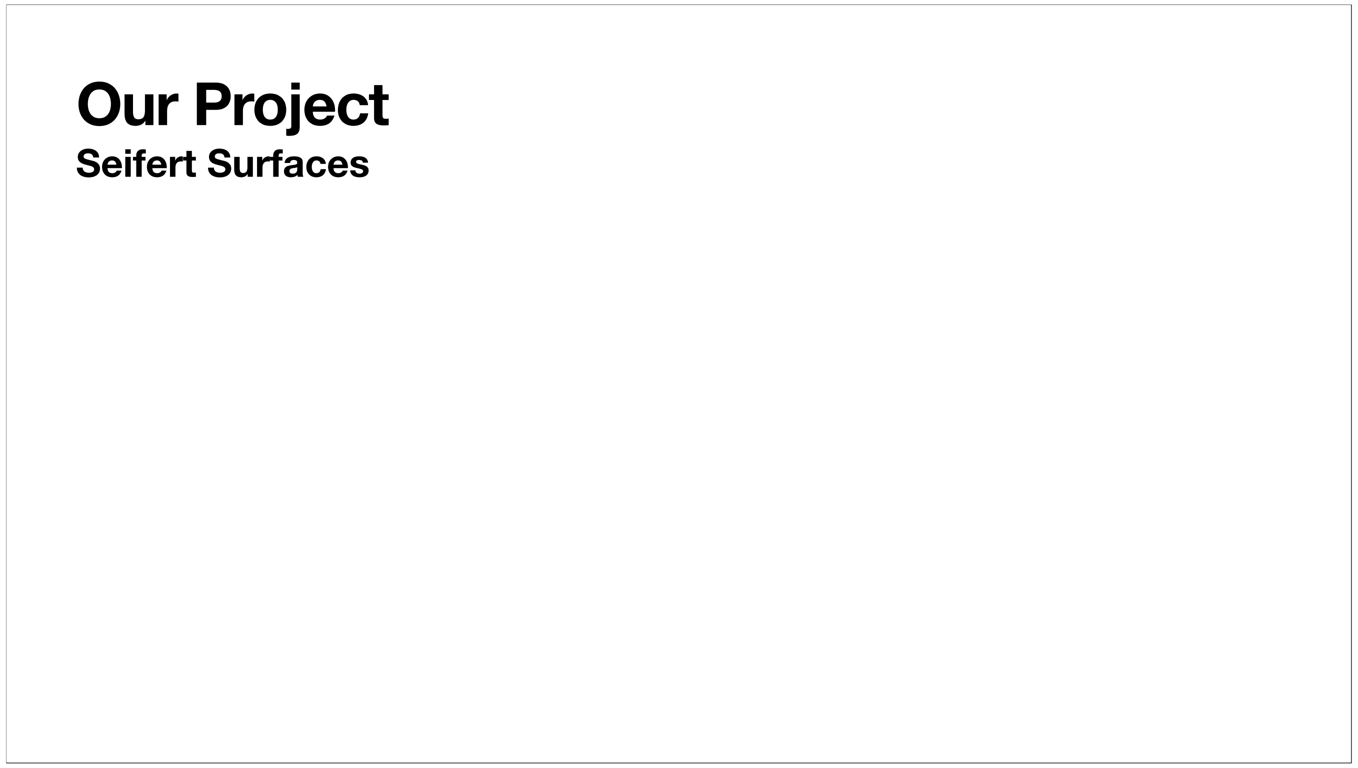
Our Project

Seifert Surfaces

- Trying to find minimal genus slice surfaces
- Seifert surfaces are orientable surfaces bounded by the knot







Our Project

Seifert Surfaces

- Some important things to know: In S^3, the unknown is the only thing that bounds a disk (has genus 0).

Our Project

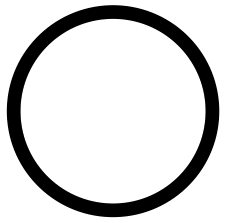
Seifert Surfaces

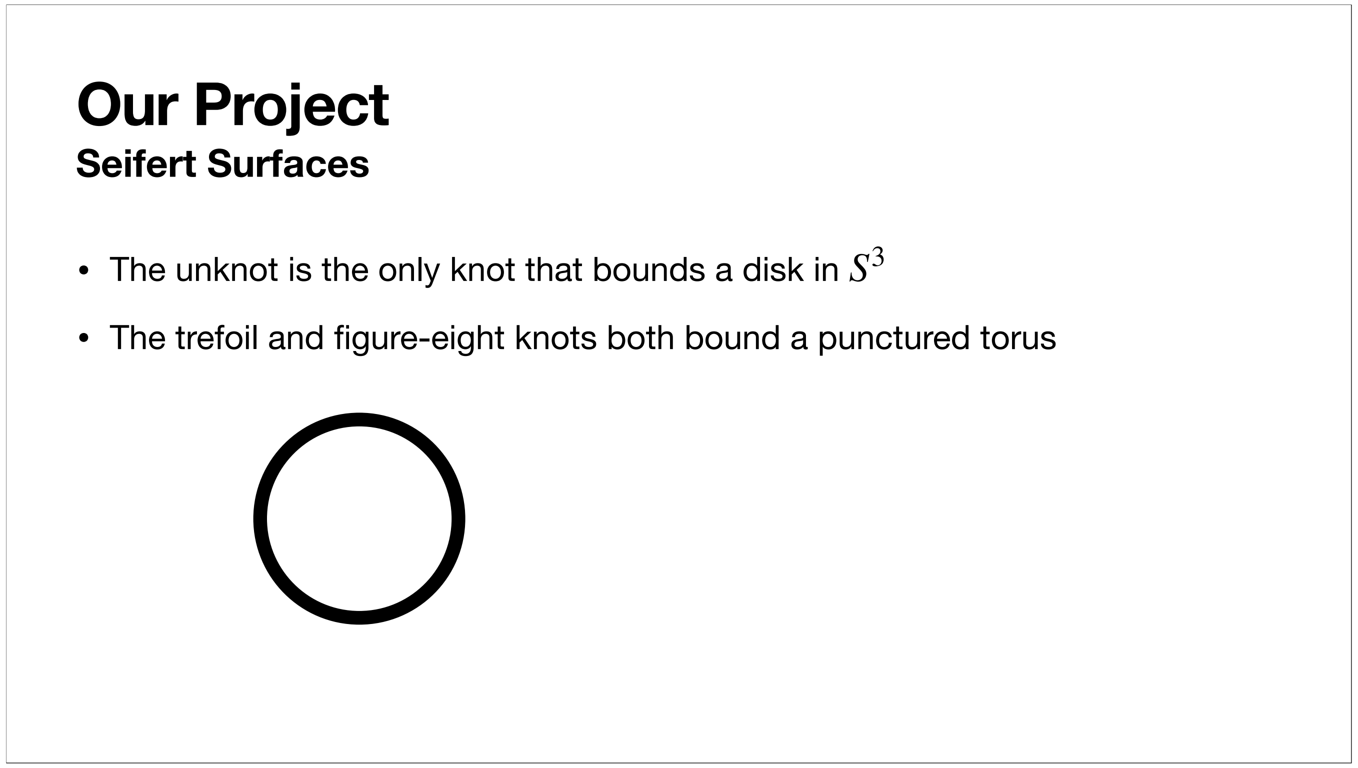
- The unknot is the only knot that bounds a disk in S^3

Our Project

Seifert Surfaces

- The unknot is the only knot that bounds a disk in S^3

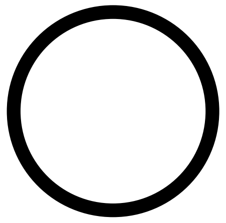




Our Project

Seifert Surfaces

- The unknot is the only knot that bounds a disk in S^3
- The trefoil and figure-eight knots both bound a punctured torus

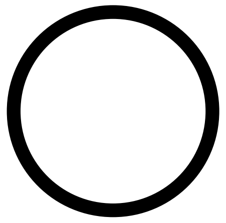


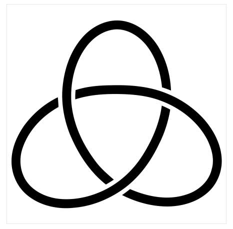
- Two knots that have genus 1 are the trefoil and the figure-eight.
- So if you were to build a surface for each of these, the unknot would just look like a flat surface with no holes, and the trefoil and figure-eight knot would have one hole each.

Our Project

Seifert Surfaces

- The unknot is the only knot that bounds a disk in S^3
- The trefoil and figure-eight knots both bound a punctured torus

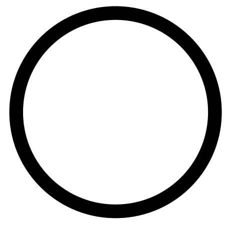
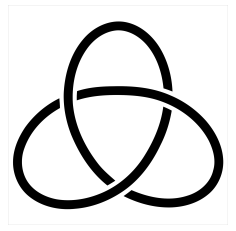
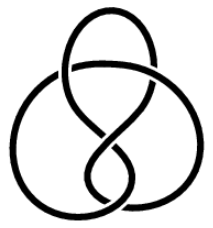




Our Project

Seifert Surfaces

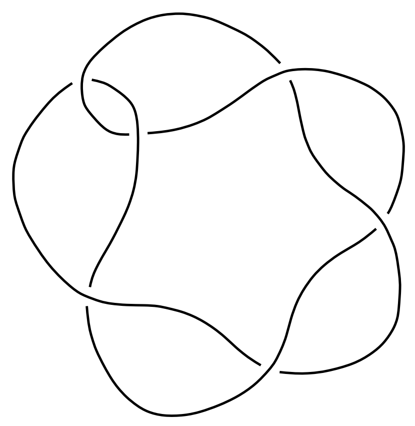
- The unknot is the only knot that bounds a disk in S^3
- The trefoil and figure-eight knots both bound a punctured torus

Our Project

Seifert Surfaces

- 6^1 knot also bounds a punctured torus in S^3

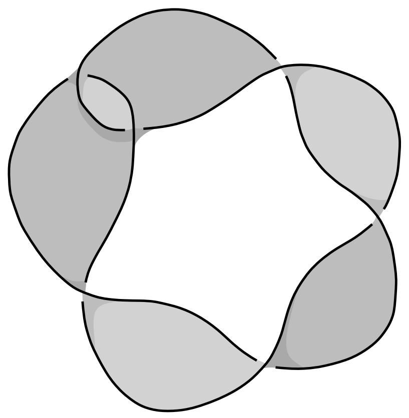


6^1 knot has genus 1 in 3D. So when you build it, the least number of holes you can get it to have is 1.

Our Project

Seifert Surfaces

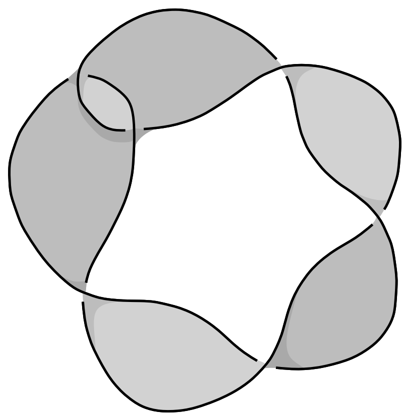
- 6^1 knot also bounds a punctured torus in S^3



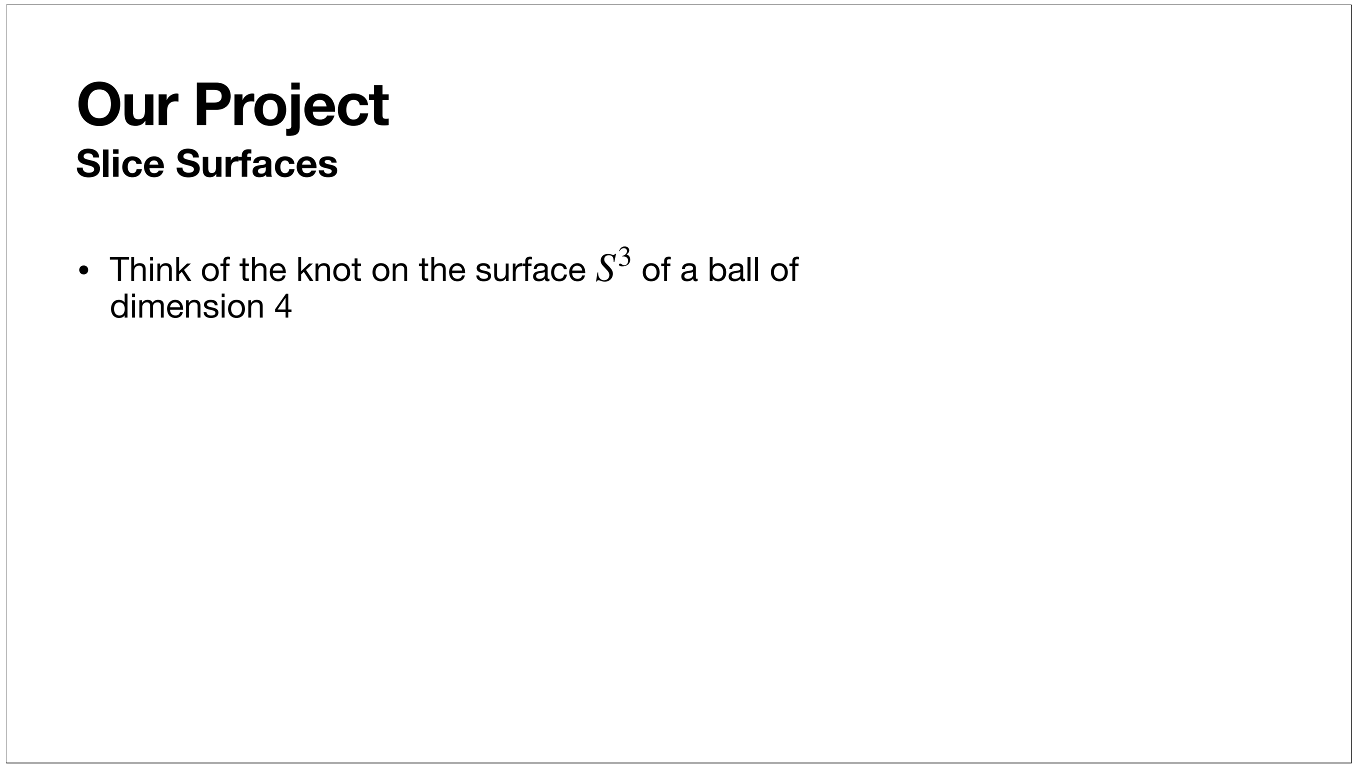
Our Project

Seifert Surfaces

- 6^1 knot also bounds a punctured torus in S^3
- Can we do better if we add an extra dimension?



What if we move this into the 4th dimension. Can we get better results?
- In fact we can! The 4D genus of the 6^1 knot is 0!



Our Project

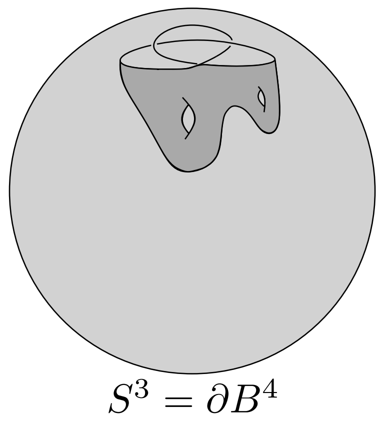
Slice Surfaces

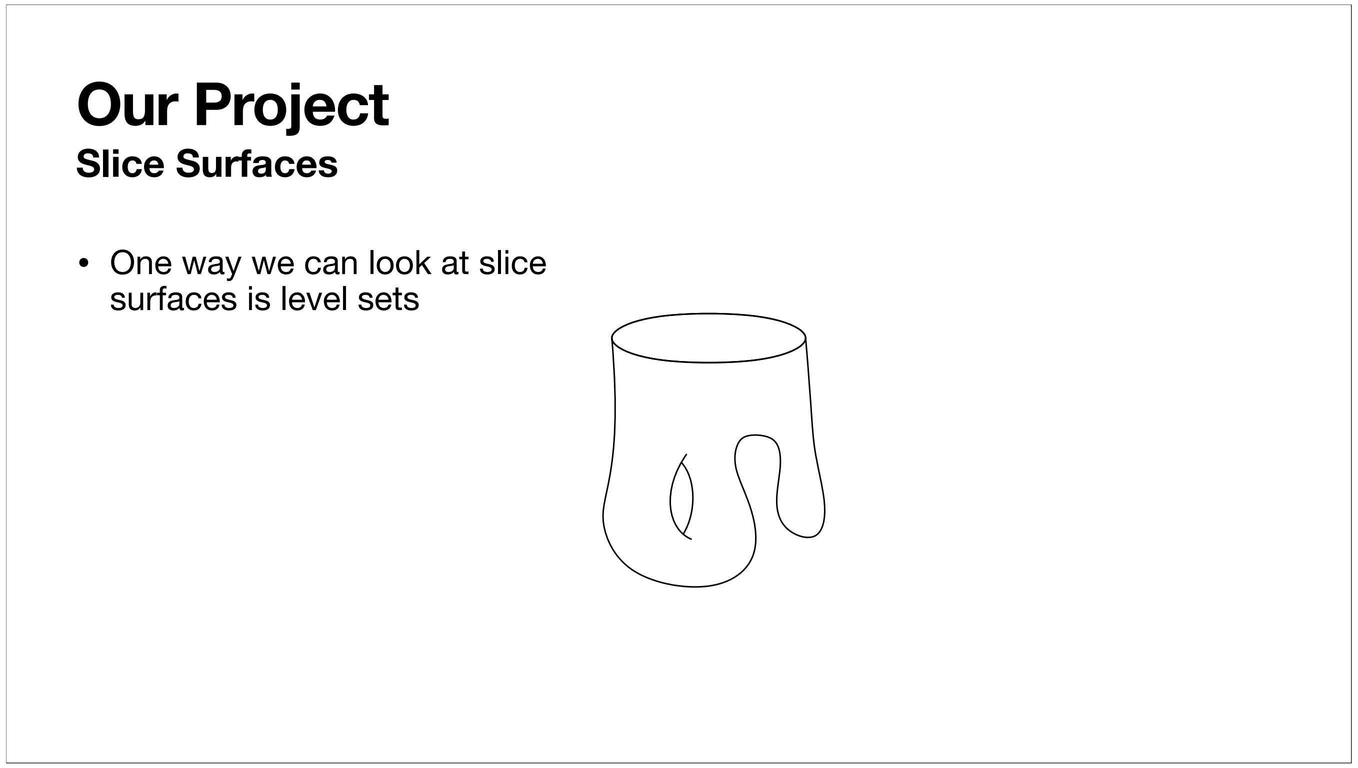
- Think of the knot on the surface S^3 of a ball of dimension 4

- The knot sitting on the surface of the 4-ball looks just like the 3D surface I showed at the beginning of my slides.
- The dark gray part slips into the 4th dimension and allows for the knot to exist there.
- But how can we look at 4-dimensional things? The answer is level sets!

Our Project
Slice Surfaces

- Think of the knot on the surface S^3 of a ball of dimension 4

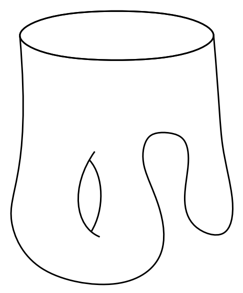




Our Project

Slice Surfaces

- One way we can look at slice surfaces is level sets

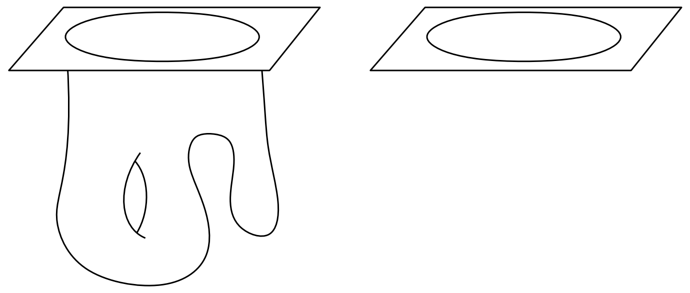


- Take cuts of our 4-dimensional slice surface at various points.
- You can see at the third level set that something has happened; maybe a saddle point has been added.
- In the fourth level set, it looks like another saddle point has been added.
- 5th level set has something disappear and a saddle point is added again (to close the hole that was created).
- Final level set shows that surface has ended.
- Of course you will make more level sets than this, but this works.

Our Project

Slice Surfaces

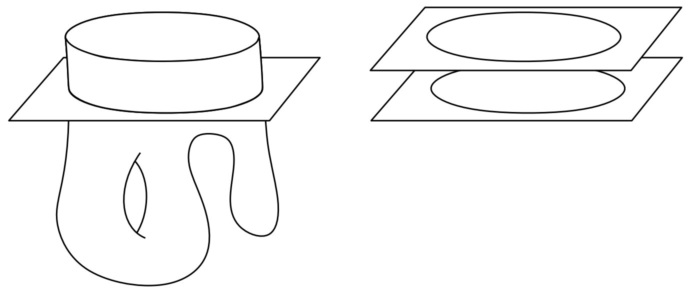
- One way we can look at slice surfaces is level sets



Our Project

Slice Surfaces

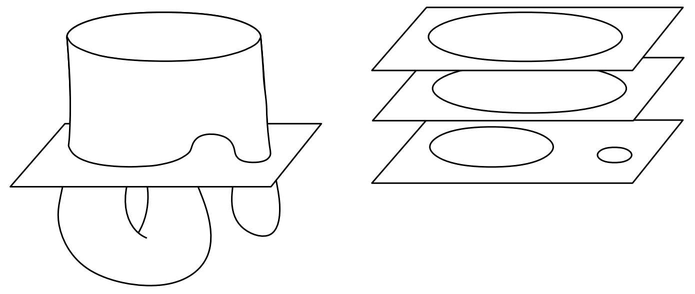
- One way we can look at slice surfaces is level sets



Our Project

Slice Surfaces

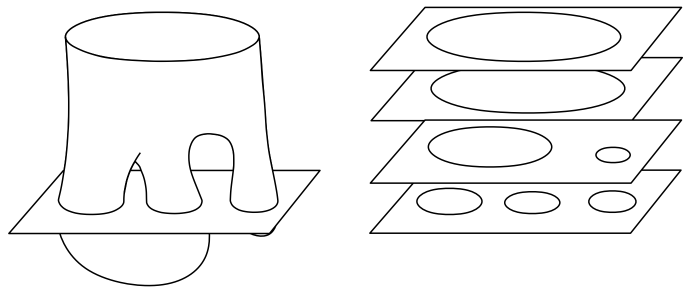
- One way we can look at slice surfaces is level sets



Our Project

Slice Surfaces

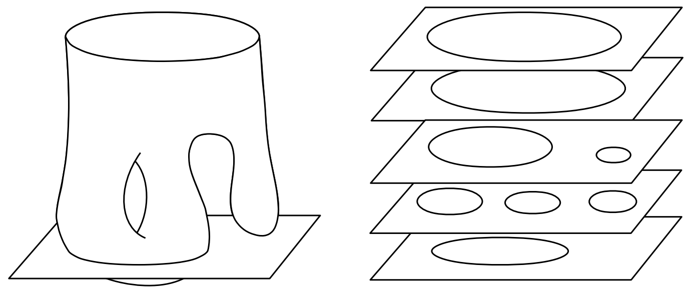
- One way we can look at slice surfaces is level sets



Our Project

Slice Surfaces

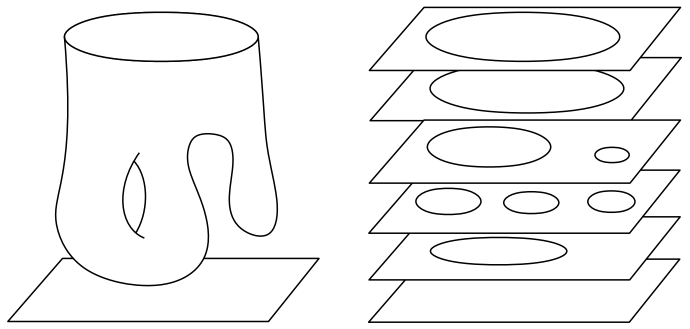
- One way we can look at slice surfaces is level sets



Our Project

Slice Surfaces

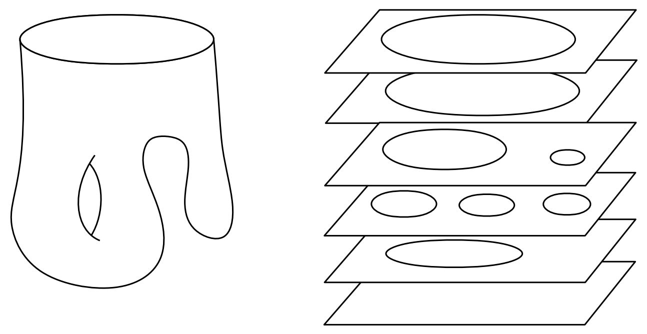
- One way we can look at slice surfaces is level sets



Our Project

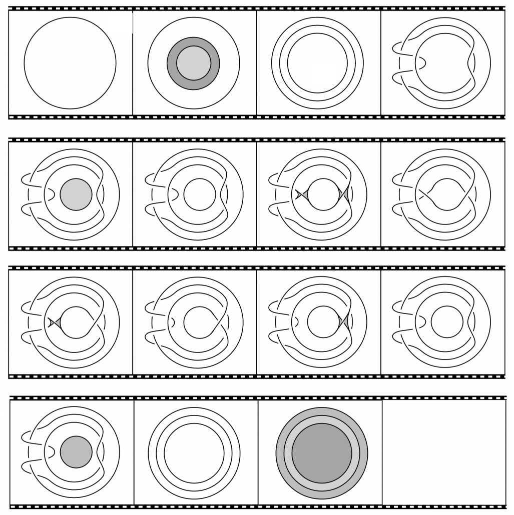
Slice Surfaces

- One way we can look at slice surfaces is level sets



Our Project

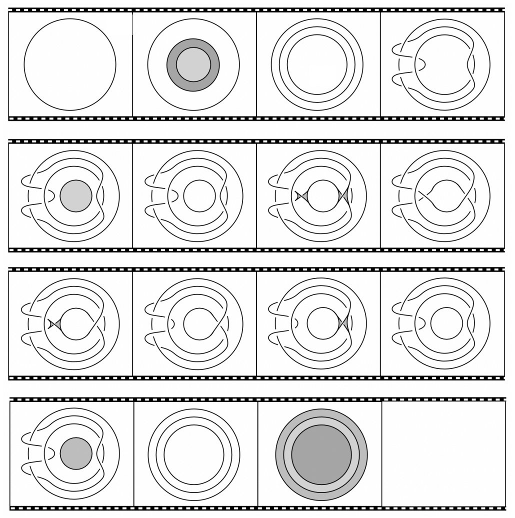
Slice Surfaces



Our Project

Slice Surfaces

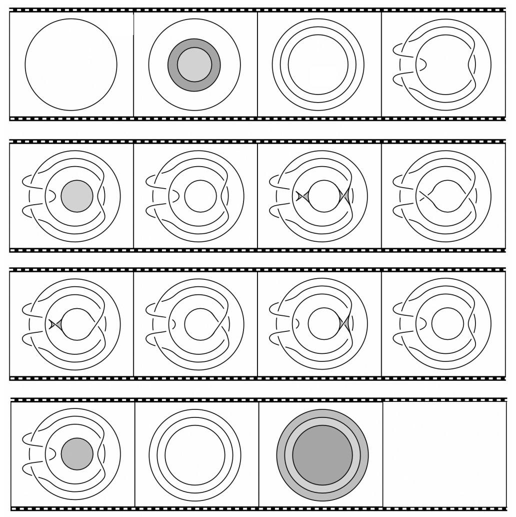
- Start with a strand



Our Project

Slice Surfaces

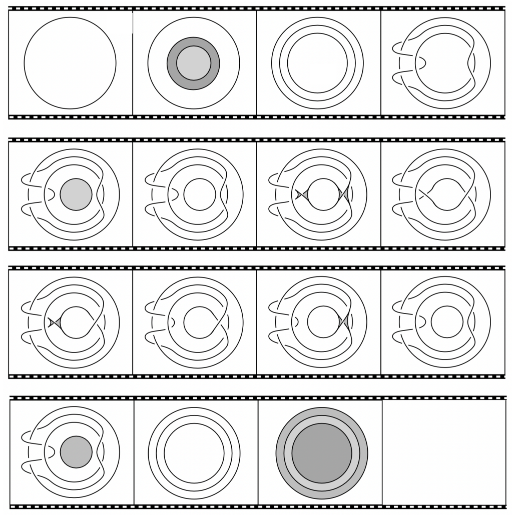
- Start with a strand
- Add some other strands



Our Project

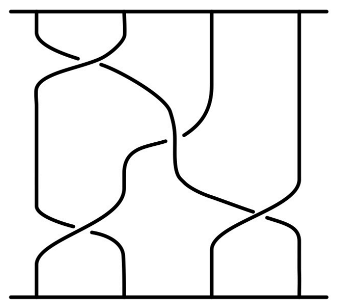
Slice Surfaces

- Start with a strand
- Add some other strands
- Move the strands around (according to a set of moves)



Our Project

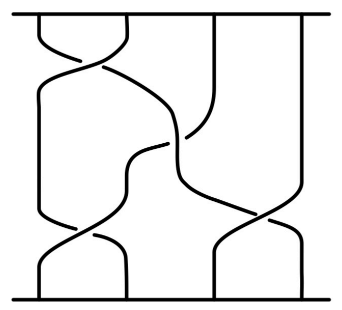
Braids



Our Project

Braids

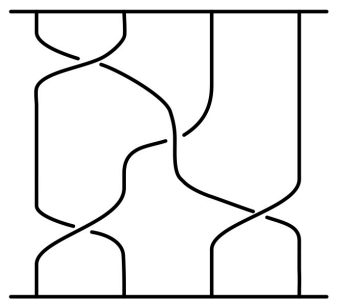
- Every knot or link can be represented in braid form



Our Project

Braids

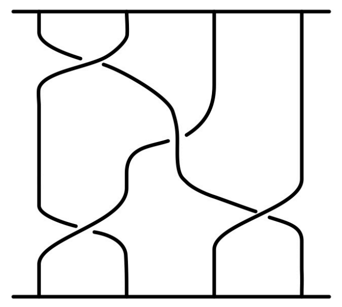
- Every knot or link can be represented in braid form
- A braid is simply a set of n strings attached to a horizontal bar at the top and the bottom



Our Project

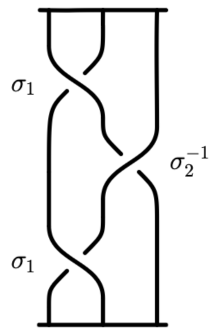
Braids

- Every knot or link can be represented in braid form
- A braid is simply a set of n strings attached to a horizontal bar at the top and the bottom
- If you cut a slice surface at one point, you can attach it to two bars to make a braid



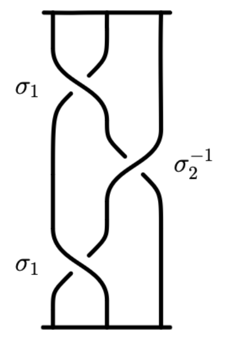
Our Project

Braids



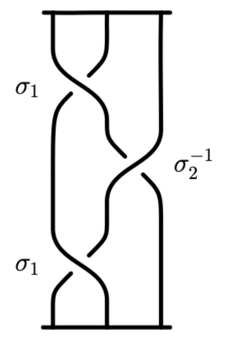
Our Project
Braids

- We represent braids through braid words



Our Project
Braids

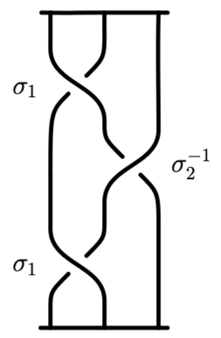
- We represent braids through braid words
- If the nth strand crosses over the $n + 1$th strand, represent as σ_n

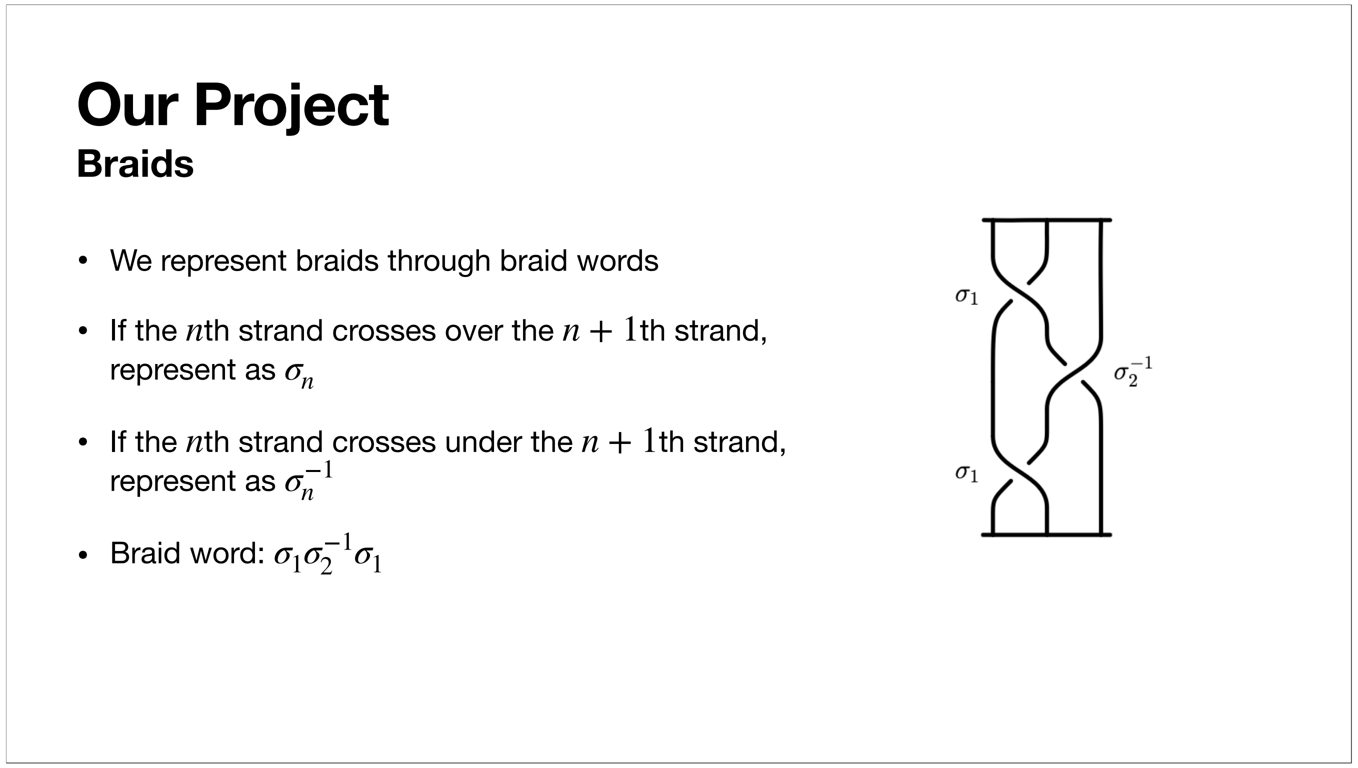


Our Project

Braids

- We represent braids through braid words
- If the nth strand crosses over the $n + 1$th strand, represent as σ_n
- If the nth strand crosses under the $n + 1$th strand, represent as σ_n^{-1}

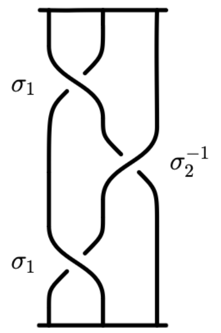




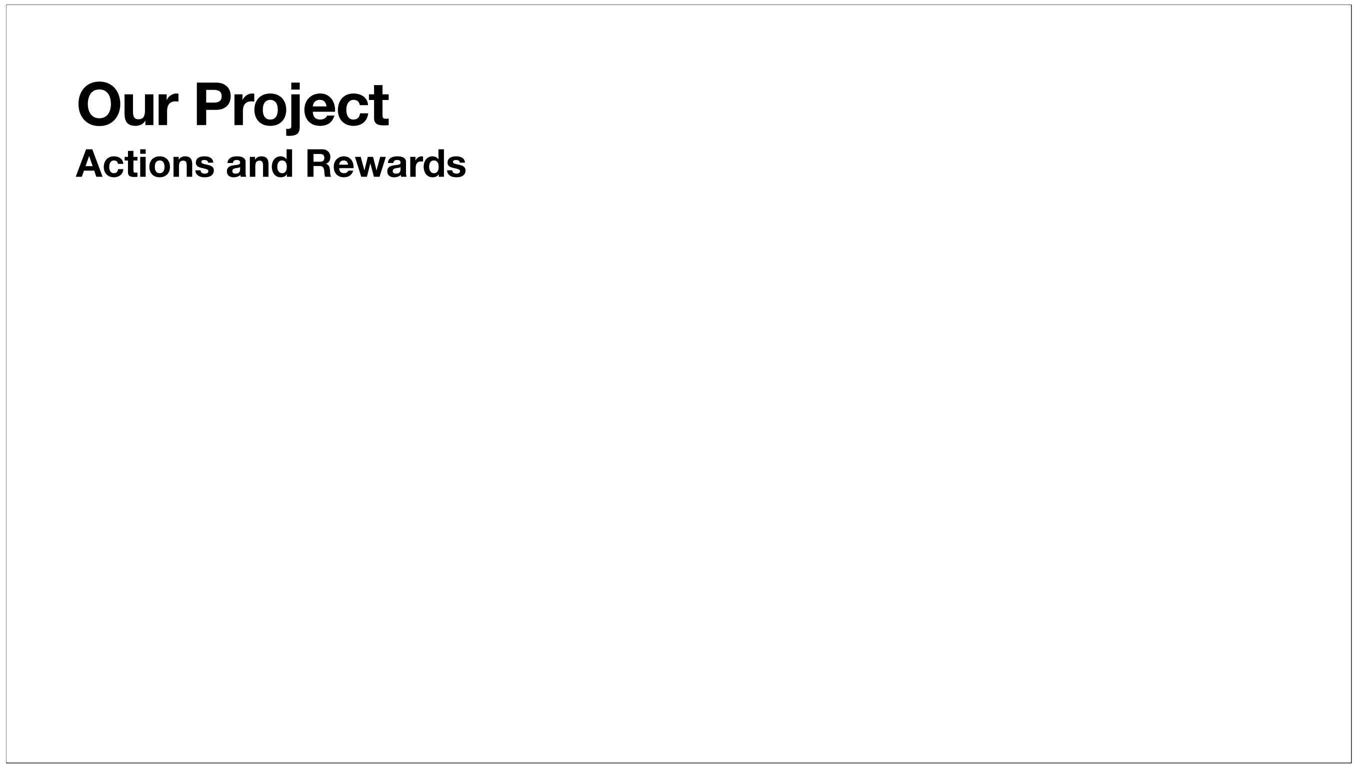
Our Project

Braids

- We represent braids through braid words
- If the nth strand crosses over the $n+1$th strand, represent as σ_n
- If the nth strand crosses under the $n+1$th strand, represent as σ_n^{-1}
- Braid word: $\sigma_1\sigma_2^{-1}\sigma_1$



- When constructing the braid word, take the sigma notation from top to bottom, and write it like that. So any reordering of the braid word would be incorrect.
- For our reinforcement learning algorithm, we represented braids by their braid word, and plugged them into our environment to be looked at.



Our Project

Actions and Rewards

- We have several moves that our agent can use. The first one is to remove a crossing.
- The next set of moves (1-10) simply change the way the braid is presented. So that could be move the cursor, move a string. Essentially anything that does not actually change the representation of the braid.
- Moves 11-12 add crossings (either over or under a strand)
- These are only negative rewards, so let’s talk about some positives.

Our Project

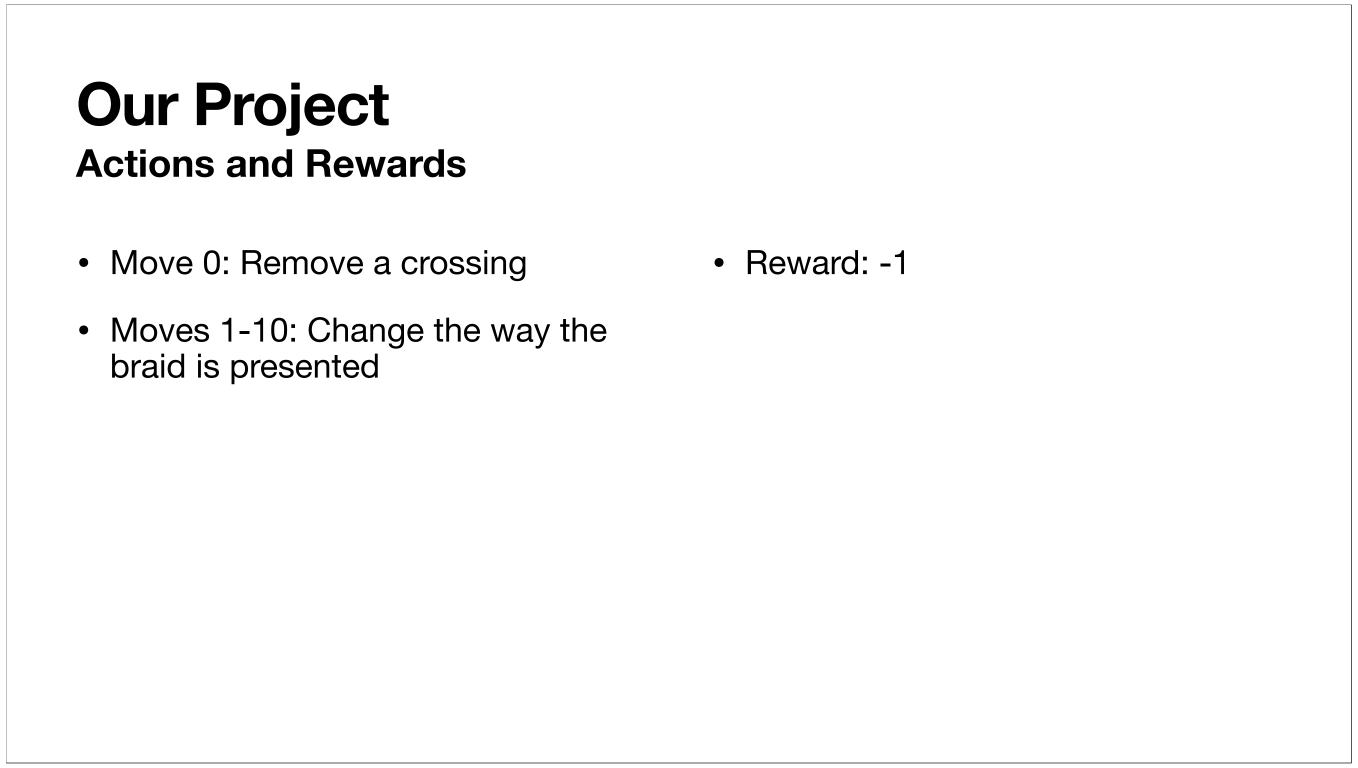
Actions and Rewards

- Move 0: Remove a crossing

Our Project

Actions and Rewards

- Move 0: Remove a crossing
- Reward: -1



Our Project

Actions and Rewards

- Move 0: Remove a crossing
- Moves 1-10: Change the way the braid is presented

- Reward: -1

- We have several moves that our agent can use. The first one is to remove a crossing.
- The next set of moves (1-10) simply change the way the braid is presented. So that could be move the cursor, move a string. Essentially anything that does not actually change the representation of the braid.
- Moves 11-12 add crossings (either over or under a strand)
- These are only negative rewards, so let’s talk about some positives.

Our Project

Actions and Rewards

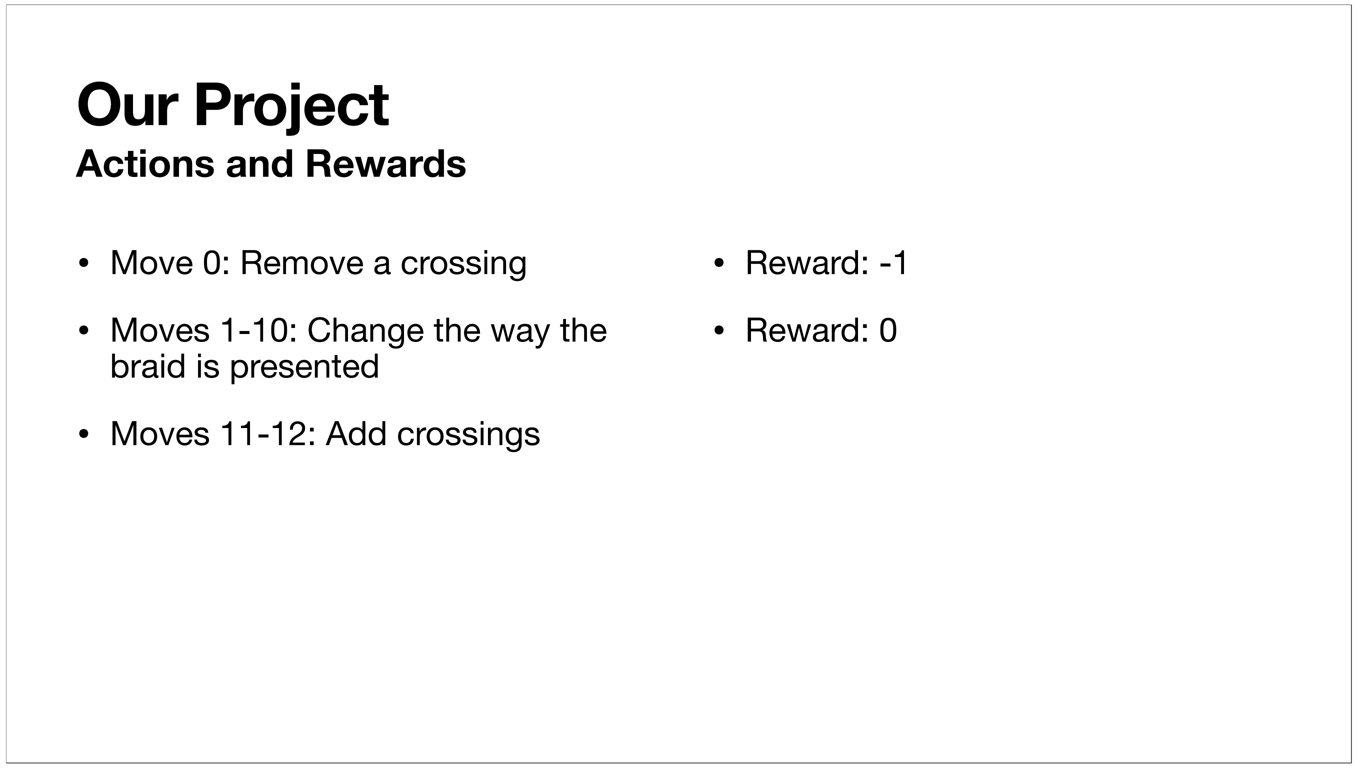
- Move 0: Remove a crossing
- Moves 1-10: Change the way the braid is presented

- Reward: -1

Our Project

Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0



Our Project

Actions and Rewards

- Move 0: Remove a crossing
- Moves 1-10: Change the way the braid is presented
- Moves 11-12: Add crossings

- Reward: -1
- Reward: 0

- We have several moves that our agent can use. The first one is to remove a crossing.
- The next set of moves (1-10) simply change the way the braid is presented. So that could be move the cursor, move a string. Essentially anything that does not actually change the representation of the braid.
- Moves 11-12 add crossings (either over or under a strand)
- These are only negative rewards, so let’s talk about some positives.

Our Project

Actions and Rewards

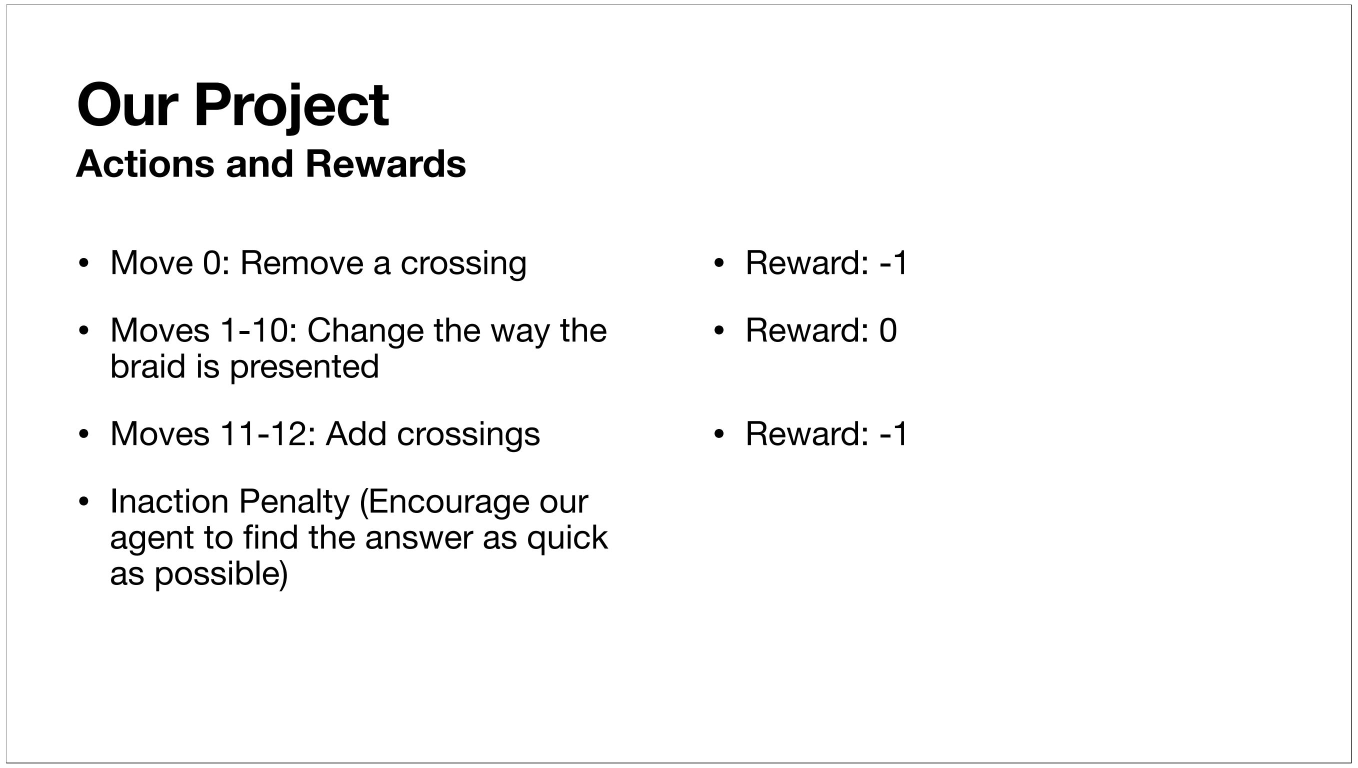
• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	

Our Project
Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	

Our Project
Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	• Reward: -1



Our Project

Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	• Reward: -1
• Inaction Penalty (Encourage our agent to find the answer as quick as possible)	

- We have several moves that our agent can use. The first one is to remove a crossing.
- The next set of moves (1-10) simply change the way the braid is presented. So that could be move the cursor, move a string. Essentially anything that does not actually change the representation of the braid.
- Moves 11-12 add crossings (either over or under a strand)
- These are only negative rewards, so let’s talk about some positives.

Our Project

Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	• Reward: -1
• Inaction Penalty (Encourage our agent to find the answer as quick as possible)	

Our Project

Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	• Reward: -1
• Inaction Penalty (Encourage our agent to find the answer as quick as possible)	

Our Project

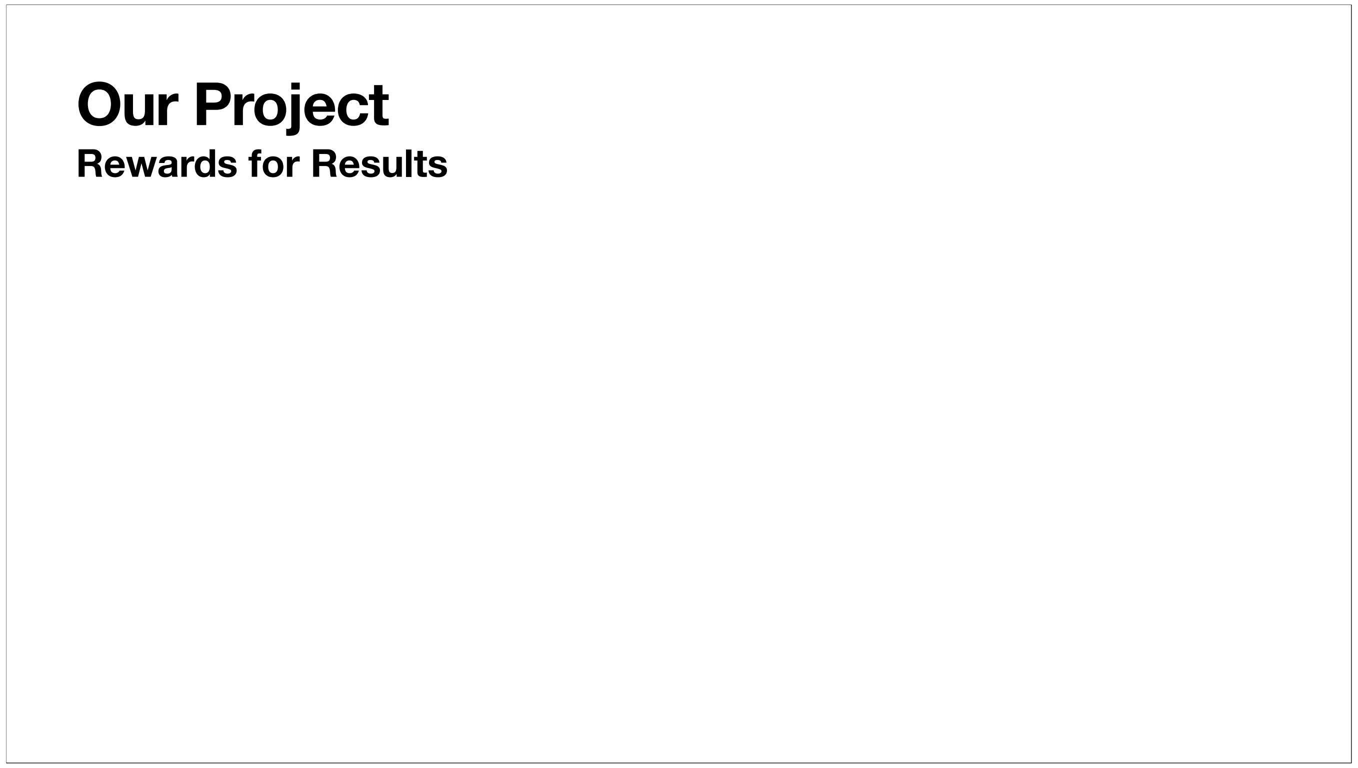
Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	• Reward: -1
• Inaction Penalty (Encourage our agent to find the answer as quick as possible)	

Our Project

Actions and Rewards

• Move 0: Remove a crossing	• Reward: -1
• Moves 1-10: Change the way the braid is presented	• Reward: 0
• Moves 11-12: Add crossings	• Reward: -1
• Inaction Penalty (Encourage our agent to find the answer as quick as possible)	• Reward: -0.05 (Hyperparameter)



Our Project

Rewards for Results

- If the agent figures out how to solve the problem in less than the allotted steps, a reward of +100 is given.
- If the agent is unable to figure out how to solve the problem in the right number of steps, a penalty of -350 is given.

Our Project

Rewards for Results

- If a move produces an unknotted, unlinked component, that component is removed and a reward of +1 is received

Our Project

Rewards for Results

- If a move produces an unknotted, unlinked component, that component is removed and a reward of +1 is received
- If the agent builds a surface with the maximal Euler characteristic, reward of +100 is given

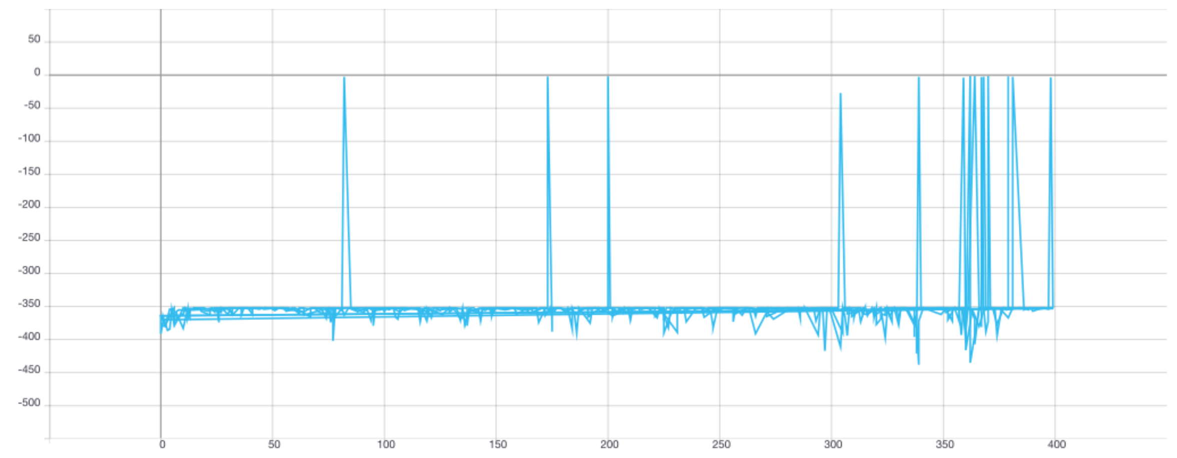
Our Project

Rewards for Results

- If a move produces an unknotted, unlinked component, that component is removed and a reward of +1 is received
- If the agent builds a surface with the maximal Euler characteristic, reward of +100 is given
- If the agent fails to build such a surface, a penalty of -350 is given

Our Project

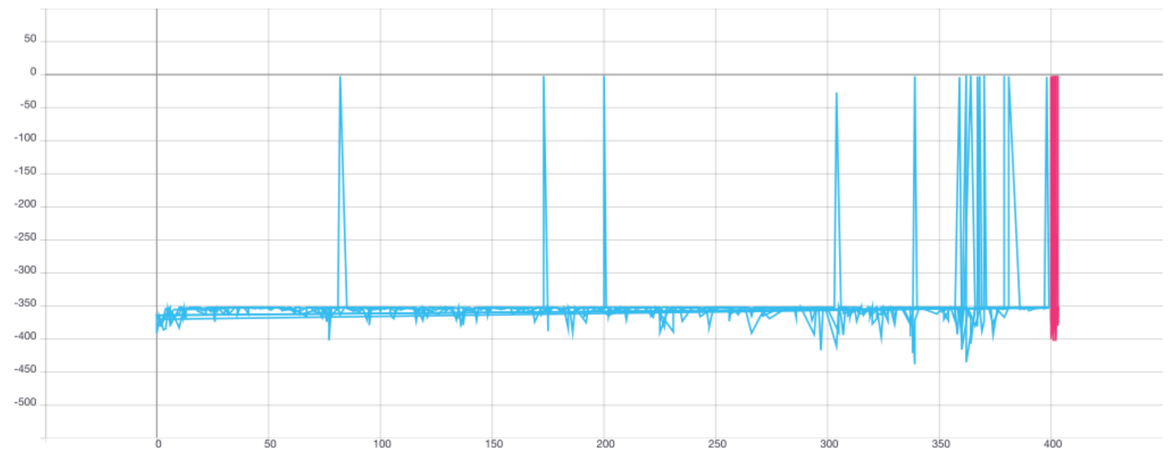
Results 4_1 Knot



Total blue training time: 15 hours 24 minutes
Total training time: 15 hours 24 minutes

Our Project

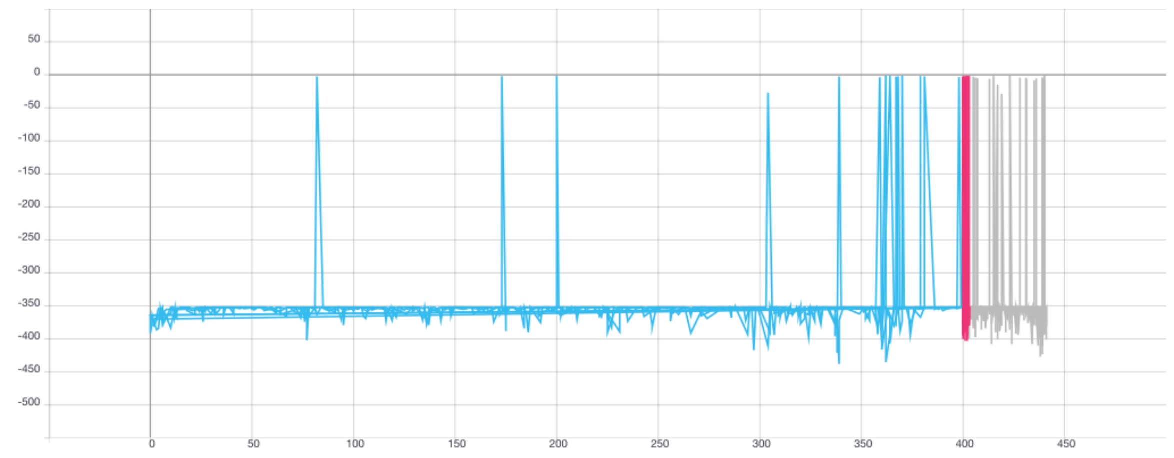
Results 4_1 Knot



Total pink training time: 5 minutes
Total training time: 15 hours 29 minutes

Our Project

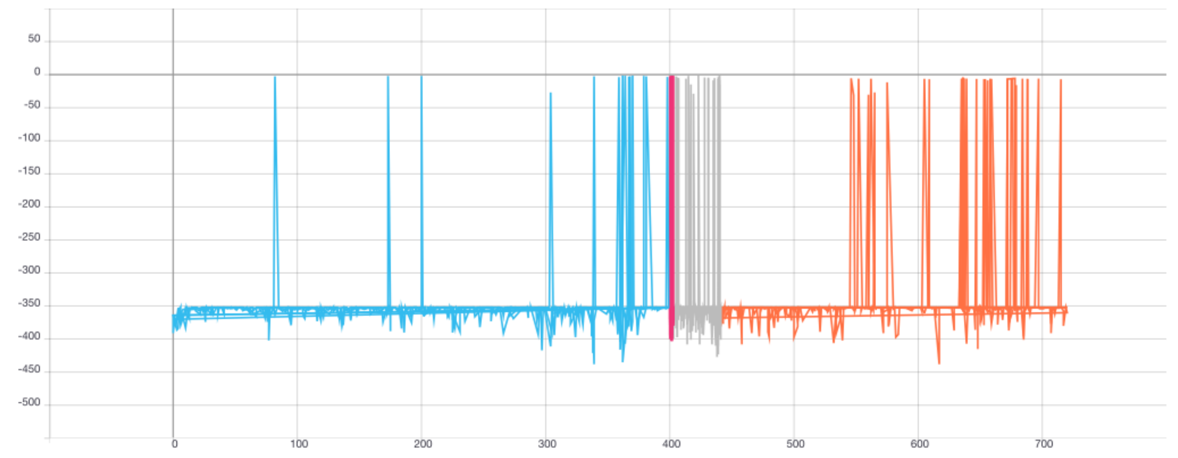
Results 4_1 Knot



Total gray training time: 1 hour 42 minutes
Total training time: 17 hours 11 minutes

Our Project

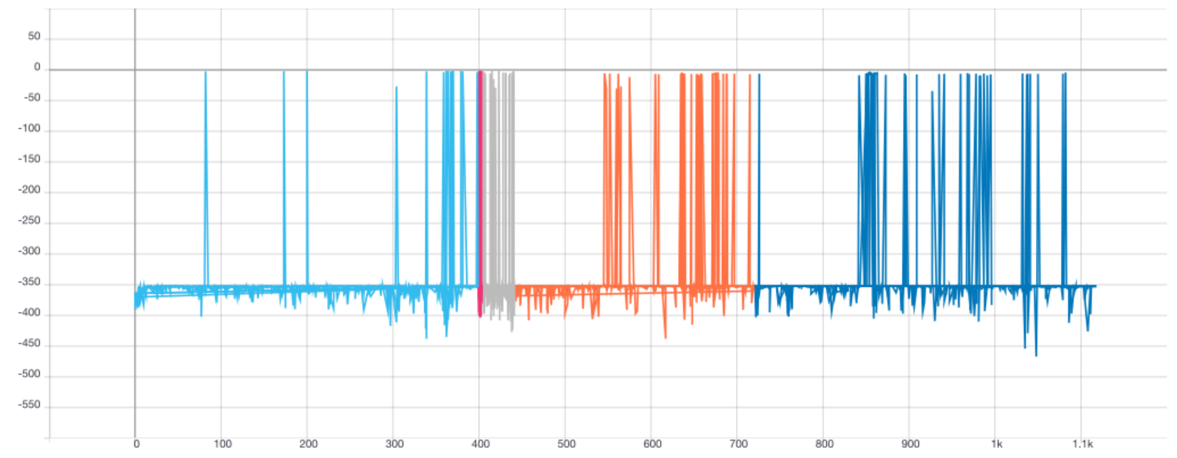
Results 7_1 Knot



Total orange training time: 24 hours
Total training time (4_1 knot): 17 hours 11 minutes
Total training time (7_1 knot): 24 hours
Total training time (in general): 41 hours 11 minutes

Our Project

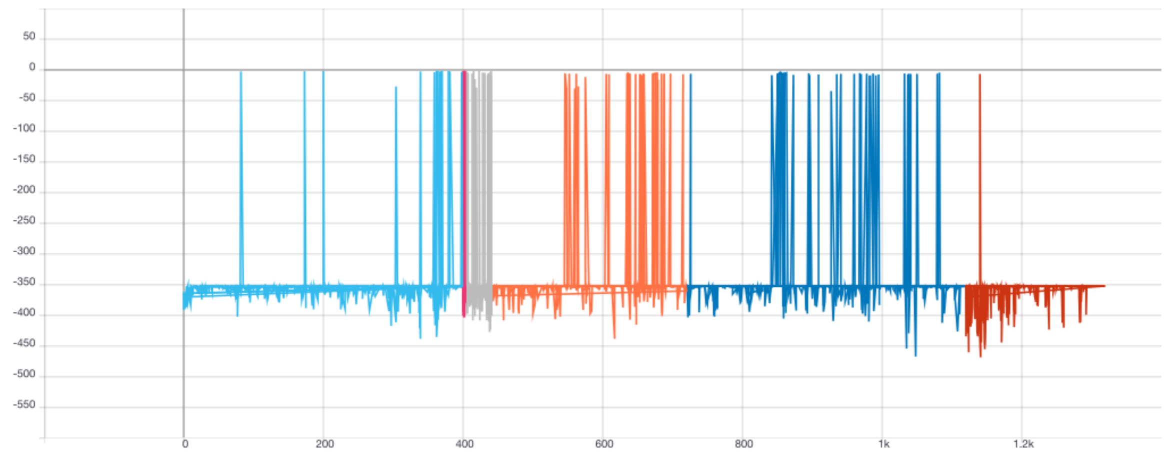
Results 7_2 Knot



Total dark blue training time: 33 hours 9 minutes
Total training time (4_1 knot): 17 hours 11 minutes
Total training time (7_1 knot): 24 hours
Total training time (7_2 knot): 33 hours 9 minutes
Total training time (in general): 74 hours 20 minutes

Our Project

Results 8_1 Knot



Total red training time: 6 hours
Total training time (4_1 knot): 17 hours 11 minutes
Total training time (7_1 knot): 24 hours
Total training time (7_2 knot): 33 hours 9 minutes
Total training time (8_1 knot): 6 hours
Total training time (in general): 80 hours 20 minutes