Good afternoon, everyone. My name is Yifei Zhou. I am a master’s student majoring in computer science at Brandeis University.

I am thrilled to discuss the recent advancements in my internship project: Enhanced robot pathfinding on Prime Vision floorplans using reinforcement learning. Today, I will walk you through my objectives, completed works, ongoing efforts, and the impact of my research.

My primary objective is to improve robot pathfinding on real Prime Vision floorplans using reinforcement learning. I aim to develop reinforcement learning algorithms, that can help the robots navigate complex environments efficiently to locate target bins, and avoid obstacles and congestion. With precise and efficient robotic movements, the reinforcement learning algorithms can bring significant improvements to the Prime Vision robot sorting by enhancing its throughput and improving its reliability.

Firstly, I would like to present the currently completed work in my internship project. I began by incorporating the floorplans into the reinforcement learning simulation environment in Python. I developed the FPenv class, which can be instantiated to transfer any floorplan JSON file to the reinforcement learning environment. This class first extracts nodes information from the JSON file, then creates the adjacency matrix according to the edges information, and finally generates a tensor dictionary as the reinforcement learning environment. Then, I trained the attention models on both computer-generated 10 by 10 grid maps and the real Butterfly 14 by 11 floorplan. I also tuned various parameters to enhance model performance. For example, I implemented the Policy Optimization with Multiple Optima (POMO), which uses a modified reinforce algorithm that forces diverse rollouts towards all optimal solutions. Besides, I also coded to allow temporary loops in training and testing, and increase the max number of steps to give the agent more chances to reach the target. I also increase the max number of training epochs to let the model continuously improve while closely monitoring the overfitting problem. Moreover, I also applied gradient clipping to avoid gradient exploding, and maintain numerical stability to improve the model’s overall performance. Finally, I also explored different decode types in the autoregressive policy for training, validation, and testing.

Secondly, I would like to show the current successful and failed results in different real floorplans. We compare the performance of our trained models with the famous A-star search baseline model. It’s like a reality check, but for AI in path-finding. In the following plots, the left subplot is the trained reinforcement learning model predictions, and the right subplot is the A\* search model predictions. The first image is the Butterfly 14 by 11 floorplan, which is also the training floorplan. As you can see, the reinforcement learning model found a much shorter path with a much smaller cost than the A\* search model in this scenario. The following floorplans are only used in the testing datasets. In the second image, the USPS Garside floorplan, the reinforcement learning model also found a much better path compared to the A\* search model. In the third image, the fp-factory floorplan, the reinforcement learning model found a path with a similar cost as the A\* search model, but it had different actions in some nodes, thus resulting in a different path. In the fourth image, the unstable testing floorplan, the reinforcement learning model found an identical path as the A\* search model. These four floorplans are some of the most complicated real floorplans in the Prime Vision floorplan repository, and I am thrilled to see the reinforcement learning model had better or at least comparable performance compared to the A\* search model.

However, there are also some scenarios in which our reinforcement learning model failed to reach the target or provided a significantly worse path compared to the A\* search model. For example, the USPS Colorado Springs floorplan, in which our reinforcement learning model was stuck in the loop and never reached the target. This is one of the major and common issues that our reinforcement learning model is facing. Another example is the USPS Torresdale floorplan, in which our reinforcement learning model finally reached the target, but had an unnecessary detour that resulted in a path with a much higher cost than the A\* search model.

Thirdly, I would like to share some in-progress work that aims to address the above issues. I am currently focusing on several key areas. First, I am improving the policy to avoid the robots being stuck in the loop. I am exploring alternative decode strategies, such as multi-start greedy, multi-start sampling, and beam search. Additionally, I am working on the codes to detect when a robot is stuck in a loop and developing strategies to help it exit the loop. Besides, I am also researching how to avoid crashing into other moving robots and robot congestion in the operations.