A APPENDIX

A.1 IMPLEMENTATION DETAILS

We used a variety of environments from MiniGrid and BabyAI (Chevalier-Boisvert et al., 2018) that provide a partial and egocentric view of the state of the environment to the agent. The reward is sparse and a positive reward is received only if the agent successfully reaches the goal. A penalty is awarded based on the number of steps taken to reach the goal, calculated as $1 - 0.9n/n_{max}$, where n_{max} is the maximum number of steps allowed for a given environment and depends on the difficulty of the environment such that more difficult environments have a larger value of n_{max}. If the agent is not able to complete the task within n_{max} steps, the episode ends and it gets a zero reward. The environments have an increasing level of difficulty in an systematically incremental manner. These settings of partial observability, sparse rewards and a systematic increase in the difficulty levels make the task for reinforcement learning algorithms sufficiently difficult.

The observation received by the agent consists of two parts: (1) an RGB image for the partial observation of the agent's field of view, and (2) a language instruction containing the mission statement that defines the task. The image part of the observation is processed through an encoder, and an embedding is computed for the textual mission statement, which are then fed to an ensemble of recurrent modules; see Fig. 1 for a full visualization of the architecture layout.

We used the Proximal Policy Optimization (Schulman et al., 2017) with parallelized data collection of rollouts collected by multiple parallel processes. For generalized advantage function, we used $\lambda = 0.99$, and discounted future rewards by a factor of $\gamma = 0.99$. Throughout the experiments, we present the mean-reward (R) and success-rate (S) of the agent, where the mean reward is the average reward across multiple runs, and the success rate represents the percentage of times the agent is able to successfully reach the goal within the n_{max} timesteps. For all of our environments, we used $n = 5$ total modules, with only $k = 3$ of them active at any given time. Further details on the specifics of each environment are provided in the Section A.2.

A.2 ENVIRONMENTS: MINIGRID AND BABYAI

We trained our agents on several environments from MiniGrid and BabyAI (Chevalier-Boisvert et al., 2018), such as GoToLocal, PickupDist, Dynamic Obstacles, DoorKey, PutNear, Fetch, FourRoomsS13, GoToObj, GoToRedBall, GoToRedBallGrey, MultiRoom, LavaCrossingS9N1 and PickUpLoc. In each of these environments, the observation consists of an RGB image of the agent's partial field of view, and an instruction textual string that contains the mission that the agent needs to solve. Fig. 1 visualizes the model architecture and the processing of the input observation. Further details on these environments, with the hyperparameters used for the recurrent modules, are provided below.

Hyperparameters: For all of the environments, we used a total of $n = 5$ modules with $k = 3$ active at any give time. More details on each environment are provided below.

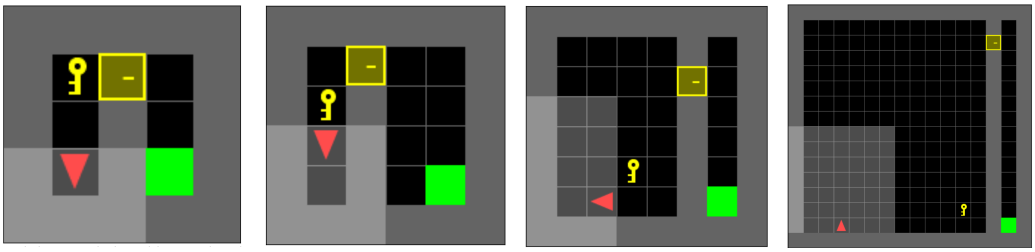

Figure 8: **Zero-shot transfer environments**: A series of DoorKey environments with an increasing level of difficulty from left to right; as the rooms get larger, the trajectories become longer and the rewards become sparser, making the tasks progressively more difficult. At the same time however, these environments share an underlying structure, making them suitable for evaluating policy generalization and transfer. We trained our agent in the leftmost (easiest) setting and evaluated it on the more difficult environments.

GoToLocal: The agent is presented with a number of objects in a single room with no doors, and is asked to go to one of them as specified in the mission statement.

PickupDist: A single room, with no doors, has a number of objects and the agent needs to pick the object instructed in the mission statement.

Dynamic Obstacles: This environment contains moving obstacles in a single room, and the agent has to reach the goal located in a corner of the room while not colliding with these dynamically moving obstacles. A large penalty is subtracted if the agent collides with an obstacle. This environment is particularly useful to train moving robots in a dynamically moving objects setting.

DoorKey: In this environment, the agent needs to find a key in the current room, pick the key and unlock a door, and then reach the goal located in the other room.

PutNear: The mission statement specifies an object that has to be put near another object. In a room of multiple objects, the agent has to find the correct objects and put them in the right location as specified by the instruction statement, thus requiring both spatial and visual understanding of multiple objects.

Fetch: The agent is presented with multiple objects of different colors and shapes, and it needs to find the object that is instructed to be fetched in the mission instruction string. A negative reward is given if a wrong object is picked.

MemoryS13Random: The agent starts in a room in which it sees an object, and has to remember this object when it reaches the end of the hallway which ends in a split. One of the sides of this split randomly contains the same object as it saw in the beginning and it has to choose that matching object.

FourRoomsS13: This is a multi-room environment, consisting of four rooms, connected by four gaps in the walls, such that the goal is randomly placed in one of the rooms. The location of the agent is also random when the episode starts.

GoToObj: This is a single room environment with a single object, specified in the mission string, that the agent needs to reach to.

GoToRedBall: The goal is to reach a red ball in a room containing distractors and other obstacles.

GoToRedBallGrey: The room consists of multiple grey distractors and the agent needs to reach the red ball, without requiring any unblocking.

MultiRoom: The environment consists of a series of interconnected rooms, and the agent has to unlock the door to navigate to the goal in the next room.

LavaCrossingS9N1: The environment consists of horizontal and vertical strips of lava running across the room, and the agent has to reach the goal while avoiding them. If the agent touches the lava, it dies and the episode ends with no reward.

PickUpLoc: A single room environment in which the goal is to pick up an object described by its location.

A.3 Ablation: Visualizing Module Activations

In the proposed setup, the modules are dynamically selected and activated based on their relevance to the current input. In order to visualize the diversity and frequency of module activations, and to analyze the effect of environment dynamics on the activation patterns of the modules, we tracked the module activations of a trained agent along a fixed length input sequence obtained by tracking

the agent navigating across the environment. We plotted these activated modules for two different environments - DynamicObstacles and PutNear, see Fig. 6 (b).

We find that for environment DynamicObstacles, in which the agent has to take more frequent decisions to figure out the best action due to quite engaging environment dynamics, such as multiple distractor objects and dynamically moving obstacles, the module activations are more diversely activated. In both the environments, different modules are active for different inputs, and all of them get activated at some point depicting an active engagement throughout the agent's interaction. However for the environment, PutNear, which has relatively fewer moving parts during the interaction, some modules are consistently more active than the others. However, all modules do get activated at some point, leaving no dead or always inactive modules.

This study has a direct analogy with how humans engage with their environments: a visual input with an enriched set of objects and more frequent interactions with the environment will need a higher engagement and continuous application of a wider set of skill sets. On the contrary, an environment which is relatively simple and has simpler input observations would need a lower engagement, and only a few components / skill-sets would mostly be able to handle and decide the best course of actions.