# Curriculum Learning for Vision-Grounded Instruction Following

**Guan-Lin Chao**[1], **Ian Lane**[2]
Carnegie Mellon University
Electrical and Computer Engineering[1,2], Language Technologies Institute[2]
{guanlinchao, lane}@cmu.edu

## Abstract

Curriculum Learning is a training strategy where the the training examples are presented to the model in a specific order such that the model learns to perform the target task more efficiently. In this paper, we focus on the vision-grounded instruction following and explore both manual and automatic Reinforcement Learning curricula. To design a manual curriculum, we define three types of hyperparameters: subtask trajectory, pacing and ordering and experiment different hyperparameter selection strategies. We also explore Teacher-Student Curriculum Learning where the Teacher is trained to automatically generate a sequence of subtasks as the Student agent's training curriculum. Experiments show that training with the manual curricula with proper hyperparameter selection and Teacher-Student Curriculum Learning both lead to performance gain on the target task compared with training on the target task directly from beginning.

## 1 Introduction

Curriculum Learning (Bengio et al., 2009; Pentina et al., 2015; Sukhbaatar et al., 2018; Wu and Tian, 2017) is a training strategy where the machine learning models learn from the training examples presented in a specific order (*curriculum*). Ideally, a well-design learning curriculum can help the model acquire the skills which constitutes the final target task more efficiently than learning from random training example order.

In this paper, we consider vision-grounded instruction following to be our target task (Misra et al., 2018; Chevalier-Boisvert et al., 2019; Shridhar et al., 2020). In our setting, the agent receives a mission (switch an appliance or fetch an object) in text form and learns to navigate and manipulate objects to complete the mission in a simulated indoor multi-room environment, as shown in Figure 1. Vision-grounded instruction following involves understanding both the language and visual context (mission description, the objects and the environment surrounding the agent) and decision making (a policy that outputs a series of actions for the agent to carry out in order to complete the mission).

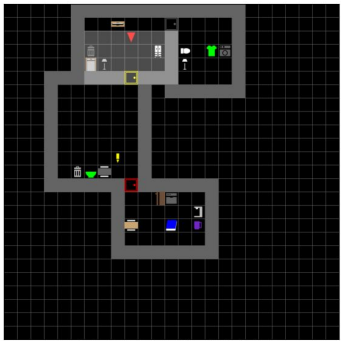


Figure 1: Example illustration of the instruction-following task. The goal of the agent (represented by the red arrow) is to complete the given mission by navigating the environment.

We explore both manual and automatic Reinforcement Learning curricula for the instruction-following task. Our contribution are as follows. Unlike studies on Vision-and-Language Navigation (Anderson et al., 2018; Das et al., 2018; Fried et al., 2018) which focus on understanding the complex and photo-realistic visual scenes from collections of real buildings scanned data, we train our instruction-following agent using a simulator which maximizes the randomness of the environment layout, object positions and appearances to enable Domain Randomization (Tobin et al., 2017). Despite simplifying the visual understanding aspect, our goal is to train an instruction-following agent that is able to generalize to act in diversified unseen environments (Section 2). We create a series of subtasks which are simplified from the target task along two complexity dimensions: mission and spatial layout. We train the agent with

| Room type | Furniture and Appliances |
|---|---|
| living room | sofa, table, TV, light, trash bin |
| kitchen | refrigerator, coffeemaker, dishwasher, table, light, trash bin |
| bedroom | bed, table, wardrobe, TV, light, trash bin |
| bathroom | washer, toilet, light, trash bin |
| **Object types** | pen, book, cup, bowl, shirt |
| **Object colors** | red, green, blue, yellow, purple |

Table 1: Furniture, appliances and object defined in our environment.

both manual and automatic curricula. In the manual curricula, the subtasks' trajectory, ordering and pacing are carefully designed with human heuristic. We utilize Teacher-Student Curriculum Learning to automatically generate learning curricula (Section 3). Experiments on the MiniGrid environment which specializes on domain randomization verify the efficacy of the proposed manual and automatic curriculum learning strategies (Section 4).

## 2 Training Framework

### 2.1 RL Environment

Our training environment is based on the Minimalistic Gridworld Environment (MiniGrid) (Chevalier-Boisvert et al., 2018) package's Multiroom task with several modifications. In each episode, the room number is randomly chosen from 2 to 4, rooms' widths and heights randomly chosen from 6 to 9 grid cells, and the room layout and door locations are also randomly generated. The rooms are randomly assigned to have different functions and the furniture and appliances are chosen and placed randomly. Household objects randomly chosen from 5 types and 5 colors are also placed at randomly locations. The number of objects is defined to be the number of rooms plus one. The details of furniture and appliances for different room types and the types and colors of objects are described in Table 1. The agent's starting position and orientation are also determined randomly for each episode.

The agent has 5 possible actions: *turn_left*, *turn_right*, *move_forward*, *toggle*, *pickup*. We define two types of missions: (1) *fetch* the specified object, *e.g.* "get the orange cup in the kitchen", and (2) switch the specified appliance (light or TV), *e.g.* "please switch on the light in the bedroom". For the *fetch* mission, task success is when agent picks up the specified object. For the *switch* mission, task success is when agent toggles the specified appliance on. Picking up the wrong object or toggling on the wrong appliance are counted as failure. If the agent takes more than *max_step* of steps before completing the task successfully, it is also counted as failure. Because in our environment the objects have more variety and makes the *fetch* mission more challenging, we define that each episode to have 75% probability with the *fetch* mission and 25% with the *switch* mission. The mission texts are generated using ten templates of natural language expressions to mimic real human's commands. In every step, the agent's observation includes the agent's partially observable *view image* (defined to be 7×7 grid cell area), and the *mission* text string which is given at the beginning of the episode.

We choose to extend the MiniGrid training framework for its strength in Domain Randomization (Peng et al., 2018). MiniGrid provide high flexibility to create complex room layouts and its object, furniture and appliance positions are completely random. By training the model that works across the highly diversified simulated environments, we expect the agent to generalize better to unseen room environments.

### 2.2 Agent Policy

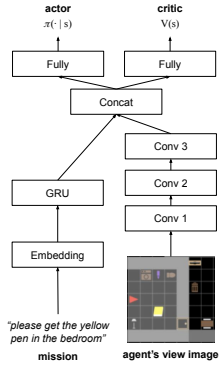


Figure 2: Agent's model architecture.

The agent's model architecture is shown in Figure 2. The network is trained using the Advantage Actor-Critic (A2C) (Wu et al., 2017) algorithm. The mission text string is encoded by a GRU cell, and the agent's view image is encoded by three convolutional layers. The textual and visual representations are concatenated and projected by a fully-connected layer respectively to output the actor's and the critic's prediction. The critic network learns to estimate the value function $V$, which is then used to calculate the advantage function:

$$\begin{aligned} A(s_t, a_t) &= Q(s_t, a_t) - V(s_t) \\ &= r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \end{aligned}$$

The actor network parameterizes the policy, and is updated in the direction suggested by the critic network, similar to other policy gradient algorithms:

$$\nabla_\theta J(\theta) \sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$
$$= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)$$

# 3 Curriculum Learning

## 3.1 Manual Curriculum

The principle of curriculum learning is starting with simpler training samples and gradually increasing the complexity of training data, with the hope that training on such a curriculum will help the performance on the target task (Hacohen and Weinshall, 2019; Weng, 2020). Our target task includes the missions of switch appliances (light and TV) and fetch objects (5 types in 5 colors) in a 2-to-4-room environment. We created the subtasks which will compose the curriculum by simplifying the target task along two complexity dimensions: *mission variety* and *spacial layout*, as shown in Figure 3. The two complexity dimensions are chosen based on human heuristic.

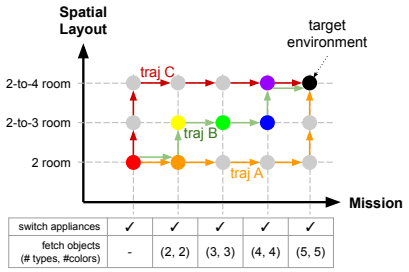


Figure 3: The target task can be simplied to easier subtasks along two complexity dimensions: mission variety and spatial layout. The subtasks are the grid points.

To design a learning curriculum, we consider the following hyperparameters: subtask trajectory, curriculum pacing and subtask ordering.

**Subtask Trajectory** Because the subtask complexity is two-dimensional, there exists multiple subtask trajectories from the simplest task to the target task in monotonically increasingly complexity, e.g. trajectory A, B, and C as shown in Figure 3.

**Curriculum Pacing** The pace of the curriculum determines the number of training steps to learn each subtask in the curriculum. We compare different training steps in a fixed pacing function, where all subtasks are trained with the same number of steps.

**Subtask Ordering** Does the ordering of subtasks in the learning curriculum affect the learning effectiveness? Besides the regular increasing complexity curriculum, we also consider an "anti" curriculum where the subtasks appear in decreasing complexity, and a "random" curriculum with random subtask order.

## 3.2 Automatic Curriculum

While it is possible to manually design a reasonable learning curriculum by carefully choosing the subtask trajectories, ordering and pacing, we also explore automatically learning the curriculum. Matiisen et al. (2019) proposed the Teacher-Student Curriculum Learning, where the agent model (Student) is trained according to the curriculum generated by the Teacher model. The Teacher model uses the Student's past episodes' returns as input, and at each timestep it predicts a subtask for the Student to practice for a few episodes and the episode returns are fed as input back to the Teacher model. The goal of the Teacher is to maximize the sum of performance for all the subtasks.

Learning the Teacher model can be formulated as a non-stationary multi-armed bandit problem. Let's denote the subtasks as $\{u_1, \cdots, u_K\}$. The horizon is denoted as $T$. In each timestep $t$, the Teacher picks a subtask $u_t = u_k, k \in [K]$. The Student trains on $u_k$ and returns the episode return $x_t$. The teacher's goal is to maximize $\sum_{t=1}^{T} x_t$. We adopt the following algorithms to train the Teacher model proposed by (Sutton and Barto, 2018).

**Online Algorithm** In Online Algorithm, we define the Teacher's reward $r_t$ as the change in student's episode return $r_t = x_t - x_{t'}$, where $t'$ is the previous timestep when $u_k$ was trained on. And we approximate the expected reward $Q$ for different subtasks using exponentially weighted moving average as $Q_{t+1}(u_t) = \alpha r_t + (1-\alpha)Q_t(u_t)$ ,where $\alpha$ is learning rate. The next subtask $u_{t+1}$ is drawn from the Boltzmann distribution $p(u) = \frac{e^{Q_{t+1}(u)/\tau}}{\sum_{k=1}^{K} e^{Q_{t+1}(u_k)/\tau}}$ ,where $\tau$ is the temperature of Boltzmann distribution.

**Window Algorithm** The Window Algorithm shares the same definition and update rule of the expected reward $Q$ and subtask selection policy. The only difference is that the reward $r_t$ is instead defined as the slope of the subtask $u_k$'s last $N$ episode returns $\{x_t, x_{t'}, \cdots, x_{t(N)}\}$, calculated using linear regression.

**Sampling Algorithm** In Sample Algorithm, the Teacher's reward has the same definition as Online Algorithm. Inspired by Thompson Sampling (Chapelle and Li, 2011), we store the last $N$ episode returns $\{x_t, x_{t'}, \cdots, x_{t(N)}\}$ in a buffer for each subtask $u_k$. Subtask selection is done by sampling a recent reward from each of the subtasks' buffer, and the subtask whose buffer yields the highest sampled reward is chosen as $u_{t+1}$.

## 4 Experimental Results

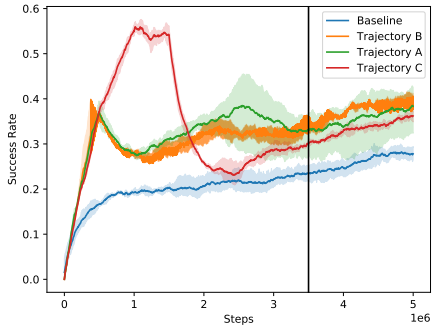


Figure 4: Learning curves comparing subtask trajectories. Training on the target task begins at the vertical line.

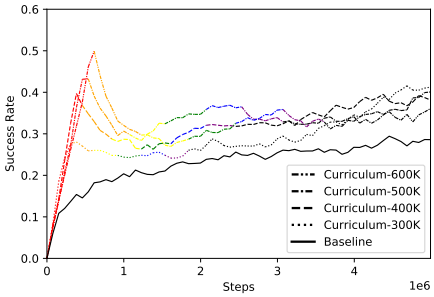


Figure 5: Learning curves comparing curriculum pacing step sizes. Each subtask is represented by one color, with the same color scheme as Figure 3. Training on the target task is represented using black color.

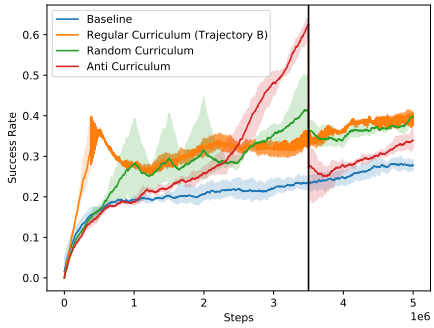


Figure 6: Learning curves comparing subtask ordering. Training on the target task begins at the vertical line.

The learning curves of manual curriculum learning are shown in Figure 4 through 6. For subtask trajectory, we compare three trajectories (A: mission-first, C: spatial-first, and B: alternating-axes) with increasing complexity, as shown in the Figure 3. It's shown that training with the curriculum following the trajectories A, B and C all outperform the baseline (only training on the target task from the beginning), and trajectory B has the best performance. In the following experiments, we use Trajectory B as the default manual curriculum. In Figure 5, we compare training each subtask for 300K, 400K, 500K or 600K steps respectively in a fixed pacing schedule. We observe that a moderate pacing which balances between under training and over training on subtasks shows the best performance and outperforms the baseline. In Figure 6, we observe that the "anti" curriculum provides a small performance gain compared with the baseline. It is noteworthy that a "random" curriculum' performance is comparable as the regular curriculum.

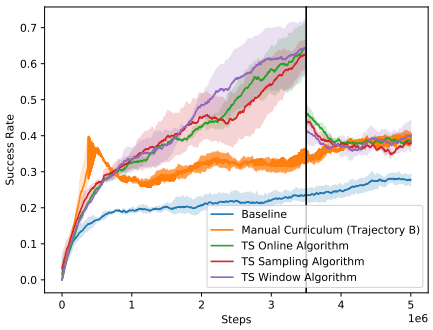


Figure 7: Learning curves comparing Teacher-Student Curriculum Learning algorithms. After the vertical line, the agent begins training on the target task.

To find the best hyperparameters to manually design a learning curriculum requires expert knowledge or a great amount of search efforts. In Figure 7, we evaluate learning the curriculum automatically with different Teacher-Student Curriculum Learning algorithms. It's shown that when spending the same number of training steps, automatic curricula achieve matching performance compared with the carefully-designed manual curriculum and outperform the baseline.

## 5 Conclusion

We propose to train a vision-grounded instruction following agent with manually-designed curricula and the automatic Teacher-Student Curriculum Learning. Experimental results provide insights of how to select hyperparameters of manual curricula and the efficacy of the learning a strong curriculum automatically.

## References


Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Computer Vision and Pattern Recognition (CVPR)*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning (ICML)*.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. Babyai: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations (ICLR)*.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Computer Vision and Pattern Recognition (CVPR)*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guy Hacohen and Daphna Weinshall. 2019. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning (ICML)*.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2019. Teacher-student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3d environments with visual goal prediction. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *International Conference on Robotics and Automation (ICRA)*.

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. 2015. Curriculum learning of multiple tasks. In *Computer Vision and Pattern Recognition (CVPR)*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *Computer Vision and Pattern Recognition (CVPR)*.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. 2018. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations (ICLR)*.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *International Conference on Intelligent Robots and Systems (IROS)*.

Lilian Weng. 2020. Curriculum for reinforcement learning. *lilianweng.github.io/lil-log*.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. 2017. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yuxin Wu and Yuandong Tian. 2017. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations (ICLR)*.