Towards Clear Evaluation of Robotic Visual Semantic Navigation

Carlos Gutiérrez-Álvarez*, Sergio Hernández-García†, Nadia Nasri*‡,
Alfredo Cuesta-Infante† and Roberto J. López-Sastre*
*University of Alcalá, Department of Signal Theory and Communications, Alcalá de Henares, Spain
Email: {carlos.gutierrezalva, nadia.nasri, robertoj.lopez}@uah.es
†Rey Juan Carlos University, Superior Polytechnic School of Computer Science, Móstoles, Spain
Email: {sergio.hernandez, alfredo.cuesta}@urjc.es
‡University of Alicante, Institute for Computer Research, Alicante, Spain

***Abstract*—In this paper we address the problem of visual semantic navigation (VSN), in which a robot needs to navigate through an environment to reach an object having only access to egocentric RGB perception sensors. This is a recently explored problem, where most of the approaches leverage last advances in deep learning models for visual perception, combined with reinforcement learning (RL) strategies. Nonetheless, after a review of the literature, it is complicated to perform direct comparisons between the different solutions. The main difficulties lie in the fact that the navigation environments in which the experimental metrics are reported are not accessible, and each approach uses different RL libraries. In this paper, we release a publicly available experimental setup for the VSN problem, with the aim of providing a clear benchmark. It has been constructed using pyRIL, an open source python library for RL, and two navigation environments: Miniwolrd-Maze from gym-miniworld, and one 3D scene from HM3D dataset using AI Habitat simulator. We finally propose a state-of-the-art VSN model, consisting in a Contrastive Language Image Pretraining (CLIP) visual encoder plus a set of two recurrent neural networks for producing the discrete navigation actions. This model is evaluated in the proposed experimental setup, with a careful analysis of the main VSN challenges, namely: the sparse rewards problem; and the exploitation-exploration trade-off. Code is available at: https://github.com/gramuah/vsn.**

***Index Terms*—navigation, reinforcement learning, robot, deep learning**

I. Introduction

One of the most important tasks that humans perform in their daily lives is semantic and goal-oriented navigation. This ability to navigate like a human towards a target in the environment is considered one of the "holy grail" goals of intelligent robots. There are countless applications that could be supported by a robotic platform with this capacity. From assistive robots that can accompany a person to perform a specific task, to platforms that navigate autonomously in complex work environments, such as logistics centers.

In this work, we address the problem of visual semantic navigation (VSN). The goal is to make a robot capable of navigating through an environment to reach a particular object (the target) in the surroundings, such as a chair, mainly using vision-based sensors. Technically, a VSN approach is a learning-based navigation model, where no geometry-based traditional techniques are applied. Nor the map of the environment is known a priori, neither the map is built on the fly. The majority of methods integrate reinforcement learning (RL) techniques with current developments in deep learning models for visual perception.

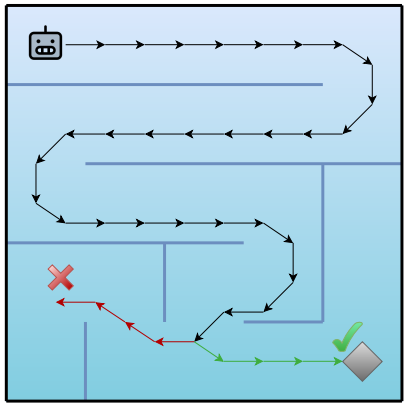


Fig. 1: Can an agent pinpoint a target in a maze using a visual semantic navigation (VSN) model based on state-of-the-art RL and deep learning models? We explore and analyze the solutions for the main challenges in VSN, *i.e.* unknown environments, visibility of targets and path planning.

The main questions we want to address in this work are: Is it possible to offer accurate experimental evaluation settings so that different RL-based VSN models may be clearly compared to one another?; What are the main challenges associated to these RL-based VSN models? With respect to the former, after reviewing the literature, we come to the conclusion that it is difficult to make direct comparisons between the many solutions provided. The key challenges are that the navigation settings in which the experimental metrics are given are not available, and that each technique employs a separate set of RL libraries. With respect to the second question, three are the main challenges that every VSN model needs to tackle. See Figure 1.

This research was funded by projects: AIRPLANE, with reference PID2019-104323RB-C31; EYEOT, with reference PID2021-128362OB-100; and POLLUTWIN, with reference TED2021-129162B-C22 from the Ministry of Science and Innovation of Spain.

Is it possible for an agent to localize a target in a maze with a VSN model based on state-of-the-art deep learning and RL models? To begin with, it must be taken into consideration that the environment is, or might be, unknown to the agent. In our experiments, we expose our agent to different mazes. In this situation, the robot would need to explore the environment to learn more about it. Second, how does the agent deal with the visibility of the targets? For a VSN model, the object we have to navigate to may not be visible at the beginning or during navigation. How does the agent learn a search strategy to find the object in the maze? And third, even if the target is visible, the robot must devise a feasible route to reach it.

The main contributions of our work are as follows:

1) We propose a VSN model which leverages state-of-the-art Contrastive Language Image Pretraining (CLIP) encoders [1]. Technically, we have designed a model that combines a CLIP encoder with a set of two recurrent neural networks for producing the discrete navigation actions that our agent needs to take (Section III).
2) The agent is trained following a RL paradigm for VSN. We propose to evaluate the impact of each of the VSN challenges mentioned above, using different techniques that have been proposed in the literature: reward shaping [2], [3] to deal with the sparsity of the reward signal naturally associated to the navigation problem; and ϵ-$greedy$ [4] as a mechanism to balance exploitation and exploration.
3) We have designed a thorough experimental evaluation setup (Section IV) with which we aim to offer a clear experimental environment in which to compare different VSN approaches. It has been implemented using pyRIL [5], an open source python library for RL, and two navigation environments: a maze navigation setup of Miniwolrd-Maze from *gym-miniworld* [6]; and a navigation through a 3D photorealistic scan indoor space provided by HM3D dataset [7] in Habitat [8] simulator. We release the codes to reproduce our experiments, as well as the whole experimental setup, so that others can compare their work with our results.

II. Related work

Visual Semantic Navigation. We can identify in the literature the following groups of works for the VSN problem, depending on the learning paradigm. In the first group, there are those works that focus on the task of navigating to an object in realistic indoor environments, *e.g.* [3], [9]–[11], using simulators and an agent based on CNNs as visual encoders and RNNs as the actor-critic head, following a RL paradigm. The second group consists of the works that address the VSN problem using imitation learning [12], [13] to build navigation policies from expert demonstrations. Finally, in the third set we have the approaches using meta-learning techniques in order to be able to quickly adapt to new environments [14]–[16].

Our work belongs to the first group. In fact, our proposal is a simplification of the approach in [11], where we build a model based on a CLIP feature extractor and two LSTMs encoders for the agent state, introducing also reward shaping [3] and ϵ-$greedy$ [4].

Sparsity and exploration methods. To address the sparse reward and exploration problems, different approaches have been proposed. Auxiliary tasks [17], [18] help the agent to explore the environment and gather extrinsic reward by maximizing pseudo-reward functions. Curiosity-driven exploration [19] leverages on the error of the agent's ability to predict the next state to introduce a new intrinsic reward that enables the agent to explore the environment. When dealing with procedurally-generated environments, a curriculum learning mechanism can be incorporated so the episodes are ordered by an exploration score [20], and then the agent imitates the best ones. We also use procedurally-generated environments, but we rely on a RL approach combined with reward shaping [21], [22] and ϵ-$greedy$ [4] techniques to learn to navigate in them.

III. RL For Navigation

A. Problem Formulation

We address the VSN problem by using Reinforcement Learning (RL). Thus, navigation can be described as a partially observable Markov decision process (POMDP), in which the agent, *i.e.* , a robot, navigates through an environment and tries to reach a determined object. This problem is known in the literature as the ObjectNav task [23].

Formally, given an initial observation distribution p_0, for the step t the agent receives an observation $o_t \sim p_0(o)$ based on state s_t, which in our case is just an RGB image of what the robot observes. The agent takes action a_t, obtains reward r_t from the environment and receives a new observation $o_{t+1} = \mathcal{T}(o_{t+1}|o_t, a_t)$, where $\mathcal{T}$ is the transition function. An episode is a sequence of (o_t, a_t, r_t) tuples that form a trajectory. The episode ends when the agent reaches the goal or the maximum number of steps (H). An episode is considered a success if the agent reaches the goal within the step horizon H.

The goal is to find an optimal policy π^* that maximizes the cumulative reward over an episode. This policy maps observations to a probability distribution over actions that is specified as follows,

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\mathcal{T}\sim\pi}[R_H], \tag{1}$$

where $R_H = \sum_{t=1}^{H} \gamma^{t-1} r_t$ is the return, *i.e.* the cumulative reward over an episode, and γ is a discount factor. In navigation tasks, neural networks with parameters θ are often used to parameterize the policy π_θ.

B. Visual Semantic Navigation

Learning to navigate in a given environment is a challenging task. First, the reward signal coming from the environment is usually sparse [2], [19]. These sparse rewards lead to a quite difficult training process. Second, we need to find a balance between the exploration and exploitation of the environment to achieve successful experiences that drive the agent's learning process [2], [4]. Finally, the agent architecture has a direct

impact on how it learns. State-of-the-art approaches use a feature extractor followed by recurrent units to process temporal information coming from the images.

Sparse rewards and long horizon. Sparse rewards are a common issue due to the nature of the navigation tasks, *i.e.* reaching a specific target in an environment. The most straightforward way to define a reward in navigation problems is to let the environment provide a fixed amount when the agent reaches the goal. This means the agent has to face an environment in which: 1) in the best case, most of the reward signal is zero except for the step in which the agent reaches the goal and obtains a certain amount of reward; and 2) if the agent does not reach the target it does not receive any reward. This situation worsens with large temporal horizons, because the more steps, the higher the sparsity of the reward is.

To mitigate the sparse reward problem, we use a technique called reward shaping. It consists in modifying the original reward signal via incorporating domain knowledge. For navigation, we leverage on the *distance reward* [3], defined as:

$$r_t = -d(s_t, target) + d(s_{t+1}, target) - r_s + r_T, \quad (2)$$

where $d(s_t, target)$ computes the geodesic distance between agent's position at state s_t and $target$'s position. r_T is the *terminal reward*, a fixed amount given only when the agent reaches the target and $r_s = 0.01$ is the *slack reward*, also a fixed amount that penalizes each step. The goal of the *distance reward* function is to give a constant reward signal to the agent that increases as the agent approaches the target. In section IV-B we compare the *distance reward* against what is usually referred to as the *navigation reward*, which consists only of the slack reward and the terminal reward $r_t = -r_s + r_T$.

Exploration vs. Exploitation. As we have mentioned, the exploration process has to be managed to encourage the agent to choose actions that it would not otherwise select. To address this issue, we leverage the technique known as ϵ-$greedy$ [4]. This solution *controls* the action that is being selected by the agent, usually during the learning process. Given an $\epsilon \in [0, 1]$, an action a_t is selected as

$$a_t = \begin{cases} \arg\max \pi_\theta & \text{with probability 1-}\epsilon, \\ rand(a) \in \mathcal{A} & \text{with probability } \epsilon, \end{cases} \quad (3)$$

where $\mathcal{A}$ defines the action space. Typically, ϵ starts at 1 and it decays with the iterations. In the beginning of the learning process, *i.e.* when ϵ is high, random actions are sampled more often, encouraging the agent to explore the environment. As the training process advances, lower ϵ values permit the agent to exploit the model knowledge to select the best action. This introduces a balance between exploration and exploitation.

Agent architecture. We encode the agent as a parameterized model consisting in a CLIP [11] visual encoder connected to two actor-critic LSTMs that output a discrete distribution over the action space and the value, respectively. A diagram of the implemented agent can be found in figure 2. To train the models, we use Proximal Policy Optimization (PPO) [24], an on-policy RL algorithm.

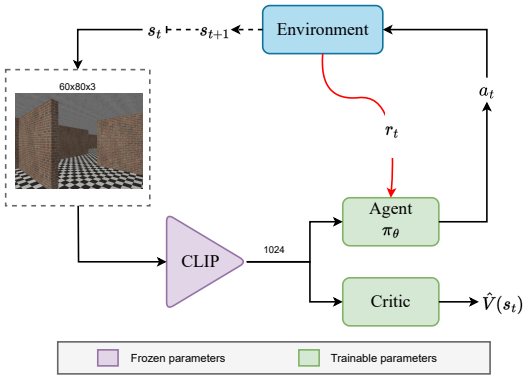


Fig. 2: **Model diagram**. This figure contains a high level representation of the model used: a visual encoder followed by an actor-critic module encoded by LSTMs. The visual encoder is frozen and we only train the actor-critic module.

IV. Experiments

In our experiments, we aim to answer the following questions:

1) How does a state-of-the-art navigation model (CLIP + LSTM + PPO) behave in a not-so-complex maze-based environment? See section IV-B.
2) What is the real impact of reward shaping and ϵ-$greedy$ techniques on such a model? See section IV-B.
3) When faced with a more realistic robotic navigation scenario, such as the one proposed with HM3D [7] dataset scenes in Habitat [8], what is the performance of the model under analysis?
4) Qualitatively, how does the model navigate through the proposed environments?
5) Is it feasible to provide clear experimental comparison environments to establish benchmarks between different RL-based visual semantic navigation models?

A. Experimental setup

a) Navigation benchmarks: We conducted our main experimentation in the Miniwolrd-Maze environment from *gym-miniworld* [6]. This is a minimalistic 3D interior environment simulator for RL and robotics, where 3D mazes can be procedurally generated. In this benchmark, the agent receives an egocentric 3D view of the environment and has to navigate to a red cube representing the target. A schematic top-view representation can be found in figure 1. We propose two configurations, Maze-S3 and Maze-S5, that correspond to a 3×3 and 5×5 tiles maze environments, respectively. The agent and the target are initialized in opposite corners of the Maze, and in every episode, a new wall distribution is randomly generated. The action space $\mathcal{A}$ consists of the following actions: $move\_forward$, $turn\_left$, $turn\_right$. To establish future comparisons with new navigation models in this benchmark, we provide 100 procedurally generated mazes and use them as a separate test set.

The second environment is AI Habitat [8], which allows for the training of embodied AI agents, such as virtual robots, in a

highly photorealistic and efficient 3D simulator. This scenario is particularly relevant because it will allow us to evaluate how a robot would behave in a more realistic navigation scenario than the one posed with the mazes. We use one 3D scene from HM3D [7] dataset (see figure 8). We follow an *oracle stop* configuration in Habitat, in which the environment is in charge of telling the agent when to stop, so the action space $\mathcal{A}$ consists of the following actions: $move\_forward$, $turn\_left$, $turn\_right$, $look\_up$ and $look\_down$.

As for the evaluation metrics, we evaluate the Maze models using Success Rate (SR) and Steps Per Episode (SPE) metrics. Additionally, for Habitat models we also employ Shortest Path Length (SPL) and Distance To Goal (DTG). All these are the standard metrics for the ObjectNav problem in Habitat Challenge [23].

b) Implementation details: We leverage on the state-of-the-art RL approach for embodied navigation in [11] with some minor simplifications. As it is shown in figure 2, the first part of our model consists in a pre-trained CLIP plus RestNet50 module as feature extractor, which receives an RGB image and produces a latent vector of size 1024. We then compute the embeddings for the last 10 time steps and pass them through an LSTM layer with 128 neurons. Finally, we concatenate two hidden linear layers of 128 neurons with a tanh unit for activation. Our agent has two separate networks, one for the actor and one for the critic. Both networks share the feature extractor. The output layer of the actor consists in a linear layer of the same dimension as the number of actions with a softmax function. The output layer of the critic consists in a linear layer with one neuron and linear activation.

We use the PPO [24] agent provided by pyRIL reinforcement learning library [5]. This is a lightweight python library which contains a collection of state-of-the-art deep reinforcement and imitation learning methods, environment wrappers, modularity and different prototyping options.

As our codes are publicly released, we provide a set of tools to improve the reproducibility of RL experiments, with clear and standardized evaluation protocols.

B. Miniworld-Maze results

First, we study how the state-of-the-art CLIP + LSTM model behaves in the Miniworld-Maze environment. Learning curves for Maze-S3 and Maze-S5 are shown in figure 3. These learning curves correspond to our best model, *i.e.* , a model trained with an ϵ-$greedy$ strategy and *distance reward* (defined in section III-B). We can observe how on Maze-S3 the agent rapidly figures out how to resolve the maze in most cases, even getting a good reward from the beginning. On the other hand, Maze-S5 learning curve shows that it is a more challenging scenario. The maze is bigger so the distance that the agent has to travel in order to reach the target is larger, as well as the number of paths to explore. This translates into a slower learning curve that takes significantly more time to achieve its peak reward.

We report the performance or our models in the proposed test set in table I. We compare between two output strategies

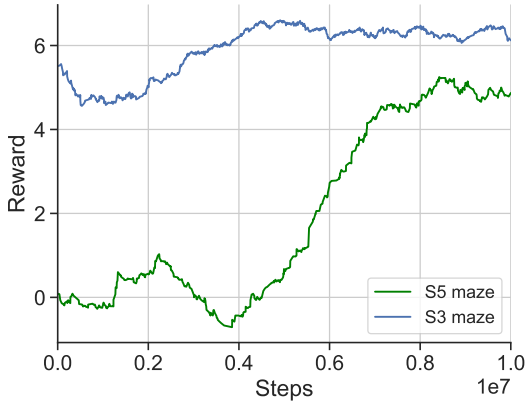


Fig. 3: **Learning curves for Maze-S3 and Maze-S5.** These curves show that the bigger the maze, the higher the complexity. On Maze-S3 the agent already starts at the saturation value around 6.5, but for Maze-S5 the agent needs more steps until it reaches its peak reward around a value of 5.

of the model to generate the actions: 1) using ϵ-$greedy$ with $\epsilon = 0.2$ during the evaluation; and 2) sampling an *stochastic* action from the final layer weights of the agent as a probability distribution. We also include a random agent as control case. Both output options obtain the best results using ϵ-$greedy$ exploration in the two mazes. We explain this fact considering how the ϵ-$greedy$ exploration is treated during training. At the beginning of the training process ϵ starts at a value of 1 and is annealed until a final value of 0.2, the same value used for evaluation. We can also see that our model achieves 3 times more success in Maze-S3 than in Maze-S5. This indicates that larger mazes are more challenging and need specific learning mechanisms.

To study the impact of reward shaping and ϵ-$greedy$ techniques we perform an ablation study as shown in table II and figure 4. The best results are obtained when *distance reward* and ϵ-$greedy$ techniques are combined, which demonstrates that both components are important in order to navigate in large environments. Note that this analysis is done in the S5 mazes. When only the *distance reward* technique is used, its performance is not enough to make the agent navigate, achieving only a 2% of success rate. On the other hand, just using the ϵ-$greedy$ strategy, the model achieves a better performance by itself, indicating that in a Maze environment it is key to explore to find the correct path to the target.

Figure 5 shows the importance of the ϵ-$greedy$ technique. When the agent reaches a corner near the target (red square), it can get stuck and run out of steps (figure 5a), but using the ϵ-$greedy$ technique lets the agent to continue exploring.

C. Habitat results

Experiments in the Habitat benchmark enable us to assess how an agent would act in a more realistic scenario than the one presented by the mazes. The agent has to navigate through the 3D scanned scene shown in figure 8. The agent is

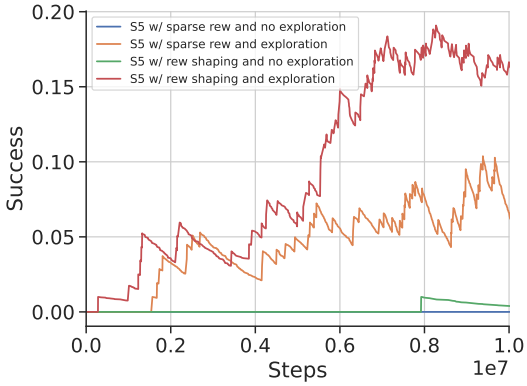


Fig. 4: **Ablation study on Maze-S5 during learning process.** Curves show that the best success rate is obtained using reward shaping and exploration techniques.

Output type	Maze	Success	SPE	Reward
Ours + ϵ-*greedy*	$S3$	**0.75 ± 0.44**	**120.59 ± 111.85**	**6.80 ± 2.29**
	$S5$	**0.18 ± 0.38**	534.40 ± 130.20	**5.24 ± 5.73**
Ours + *stochastic*	$S3$	0.63 ± 0.49	127.42 ± 132.98	6.59 ± 2.41
	$S5$	0.17 ± 0.38	**521.39 ± 182.66**	5.14 ± 5.70
random	$S3$	0.18 ± 0.39	278.04 ± 51.55	0.37 ± 3.66
	$S5$	0.02 ± 0.14	596.07 ± 32.83	-2.09 ± 4.06

TABLE I: **Evaluation performance for the best models on 100 test mazes.** We compare the evaluation between using ϵ-*greedy* with $\epsilon = 0.2$ and using an *stochastic* output, *i.e.* , sampling actions from the last layer of the agent. In both mazes the best result is obtained with ϵ-*greedy*.

initialized from random positions, and aims to locate one of the chairs present in the environment.

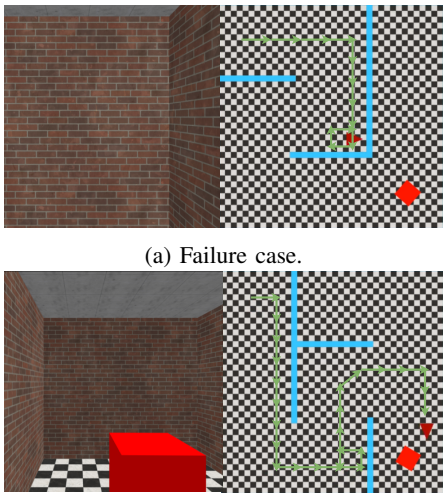
Figure 6 shows the reward obtained by the agent during the training process. The first million steps correspond to an early stage of exploration. Then, the reward quickly grows until the agent behavior becomes stable after 3 million steps.

Table III shows a comparison between our best agent under the same two different output options as in the previous experiment (ϵ-*greedy* with $\epsilon = 0.2$ and *stochastic*), and a random agent as control case. Results show how the ϵ-greedy approach reports a success rate of 96%, while the stochastic output approach only reaches the target 73% of the times.

Figure 7 provides qualitative results for our agent. It shows the final state (left image) and the top view with the agent's trajectory in blue. These figures clearly show how in both cases a different valid goal is reached (the agent reaches two different chairs) and how the ϵ-greedy strategy leads the agent

Reward function	Exploration strategy	Success	SPE	Reward
distance reward	ϵ-*greedy*	**0.18 ± 0.38**	**534.40 ± 130.20**	**5.24 ± 5.73**
navigation reward	ϵ-*greedy*	0.09 ± 0.29	575.86 ± 91.94	0.08 ± 0.26
distance reward	No	0.02 ± 0.14	588.66 ± 79.78	-1.24 ± 4.18
navigation reward	No	0.00 ± 0.00	600.00 ± 0.00	0.00 ± 0.00

TABLE II: **Ablation study for S5 mazes on 100 test mazes.** The results show that the best performance is obtained when *distance reward* and ϵ-*greedy* techniques are used.



(a) Failure case.

(b) Success case.

Fig. 5: **Qualitative results for Miniworld-Maze agent.** We show agent final state and trajectory for a fail case (5a) and a success case (5b). In the success case, ϵ-*greedy* forces the agent to select a random action, thus exploring the environment, escaping the corner and finally reaching the target.

Output type	Success	SPL	DTG	SPE	Reward
Ours + ϵ-*greedy*	**0.96 ± 0.19**	0.66 ± 0.25	0.25 ± 0.85	**189.99 ± 116.97**	**4.96 ± 1.99**
Ours + *stochastic*	0.73 ± 0.45	0.58 ± 0.36	0.63 ± 1.17	231.23 ± 188.13	3.52 ± 3.90
random	0.05 ± 0.22	0.02 ± 0.10	4.49 ± 1.72	495.50 ± 26.96	−4.68 ± 2.16

TABLE III: **Best agent performance on 100 test episodes in Habitat.** The ϵ-greedy output mode reports the best results.

to do coarser movements.

V. Conclusions

In this paper, we offer a thorough experimental evaluation setup for the VSN problem based on the open-source pyRIL library and two navigation environments: Miniworld-Maze; and a 3D scanned scene from HM3D database using Habitat simulator. Following this proposal, we offer a detailed analysis

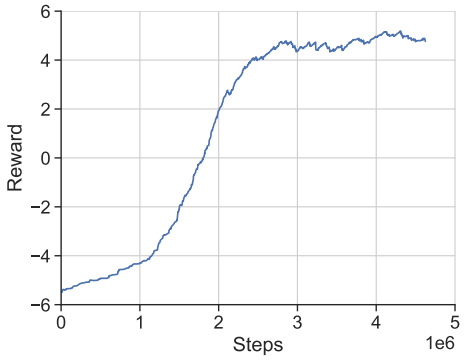
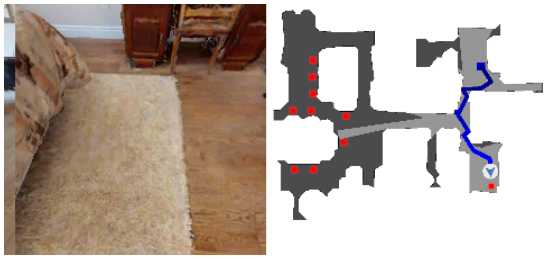
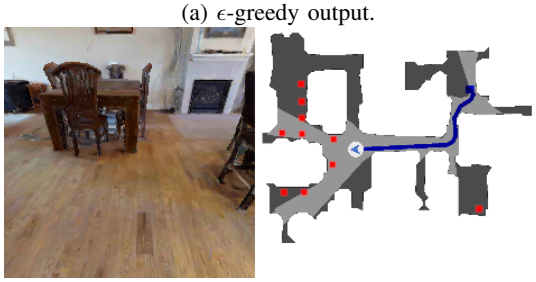


Fig. 6: **Learning curve of Habitat experiment.** This curve shows how the agent starts with a sub-optimal policy, receiving low rewards around -5. Then, the rewards increase until a value around 5, once the agent gets an optimal policy.



(a) ϵ-greedy output.



(b) $Stochastic$ output.

Fig. 7: **Qualitative results for Habitat agent.** Here we show the agent final state and trajectory on the scene using ϵ-greedy with $\epsilon = 0.2$ (7a) and $stochastic$ output (7b). Note that the $stochastic$ output produces a smother trajectory.

Fig. 8: **Scene 00744-1S7LAXRdDqK from HM3D dataset.** Scene used for Habitat experiments.

of a state-of-the-art VSN model based on a CLIP feature extractor and LSTMs encoders for the agent state, introducing also reward shaping and ϵ-$greedy$ techniques. Our results confirm the impact of these techniques in both environments. The models, data and codes are publicly available at https://github.com/gramuah/vsn to encourage much needed further research on VSN.

REFERENCES

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *ICLR*, vol. 139, 18–24 Jul 2021, pp. 8748–8763.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, second edition ed., ser. Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts: The MIT Press, 2018.

[3] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," in *ICLR*, 2020.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013. [Online]. Available: https://arxiv.org/abs/1312.5602

[5] S. Hernandez-Garcia, "pyril: Python reinforcement and imitation learning library," https://github.com/SergioHdezG/pyRIL, 2022.

[6] M. Chevalier-Boisvert, "Miniworld: Minimalistic 3d environment for rl and robotics research," https://github.com/maximecb/gym-miniworld, 2018.

[7] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. Chang, M. Savva, Y. Zhao, and D. Batra, "Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI," *NeurIPS*, 2021.

[8] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training Home Assistants to Rearrange their Habitat," in *NeurIPS*, 2021.

[9] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning," in *ICLR*, 2017.

[10] M. Chang, A. Gupta, and S. Gupta, "Semantic Visual Navigation by Watching Youtube Videos," in *NeurIPS*, 2020.

[11] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but Effective: CLIP Embeddings for Embodied AI," in *CVPR*, 2022, p. 10.

[12] Q. Wu, X. Gong, K. Xu, D. Manocha, J. Dong, and J. Wang, "Towards target-driven visual navigation in indoor scenes via generative imitation learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 175–182, 2020.

[13] R. Ramrakhya, E. Undersander, D. Batra, and A. Das, "Habitat-Web: Learning Embodied Object-Search Strategies from Human Demonstrations at Scale," in *CVPR*. New Orleans, LA, USA: IEEE, Jun. 2022, pp. 5163–5173.

[14] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv:1611.05763 [cs, stat]*, Jan. 2017.

[15] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to Learn How to Learn: Self-Adaptive Visual Navigation Using Meta-Learning," *CVPR*, pp. 6743–6752, 2019.

[16] S. Zhang, W. Li, X. Song, Y. Bai, and S. Jiang, "Generative meta-adversarial network for unseen object navigation," in *ECCV*, Cham, 2022, pp. 301–320.

[17] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *ICLR*, 2017.

[18] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable ObjectGoal navigation," in *ICCV*, Oct. 2021, pp. 16 117–16 126.

[19] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *ICML*, vol. 70, 2017, pp. 2778–2787.

[20] D. Zha, W. Ma, L. Yuan, X. Hu, and J. Liu, "Rank the Episodes: A Simple Approach for Exploration in Procedurally-Generated Environments," in *ICLR*, Sep. 2020.

[21] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICLR*, 1999, pp. 278–287.

[22] C. Jestel, H. Surmann, J. Stenzel, O. Urbann, and M. Brehler, "Obtaining robust control and navigation policies for multi-robot navigation via deep reinforcement learning," in *ICARA*, 2021, pp. 48–54.

[23] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects," in *arXiv:2006.13171*, 2020.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, Aug. 2017.