Object Goal Navigation using Data Regularized Q-Learning

Nandiraju Gireesh[1], D. A. Sasi Kiran[1], Snehasis Banerjee[2], Mohan Sridharan[3]
Brojeshwar Bhowmick[2], Madhava Krishna[1]

[1]Robotics Research Center, IIIT Hyderabad, India
[2]TCS Research, Tata Consultancy Services, India
[3]Intelligent Robotics Lab, University of Birmingham, UK

***Abstract*— Object Goal Navigation requires a robot to find and navigate to an instance of a target object class in a previously unseen environment. Our framework incrementally builds a semantic map of the environment over time, and then repeatedly selects a long-term goal ('where to go') based on the semantic map to locate the target object instance. Long-term goal selection is formulated as a vision-based deep reinforcement learning problem. Specifically, an Encoder Network is trained to extract high-level features from a semantic map and select a long-term goal. In addition, we incorporate data augmentation and Q-function regularization to make the long-term goal selection more effective. We report experimental results using the photo-realistic Gibson benchmark dataset in the AI Habitat 3D simulation environment to demonstrate substantial performance improvement on standard measures in comparison with a state of the art data-driven baseline.**

***Index Terms*— Deep Reinforcement Learning, Data Augmentation, Q-value Regularization, Object Goal Navigation.**

I. Introduction

Object-goal navigation (ObjectNav) is a key task in robotics, which requires a robot to navigate to an instance of a target object class in a previously unseen environment [1]. This is a challenging task because the robot needs to address multiple problems such as understanding the current scene from sensor input (e.g., RGB-D observation of the scene), computing a suitable path to the likely location of an instance of the target object, and navigating reliably to the desired location. Conventional methods for the ObjectNav task usually require comprehensive domain knowledge, e.g., map of the environment, which is not feasible in many practical domains. Although navigation methods based on end-to-end learning methods have resulted in promising results on specific datasets, they incur high computational costs and tend to generalize poorly to previously unseen scenes.

Modular reinforcement learning (RL)-based methods for ObjectNav have emerged as a strong competitor to end-to-end RL methods with better sample efficiency, generalization to new scenes, and transfer from simulation to the real world. They rely on separate modules for mapping, exploration, and navigation. The state of the art modular RL method for ObjectNav, Goal-Oriented Semantic Exploration (SemExp) [2], has low overall accuracy due to failure cases that can be attributed to poor selection of *long-term goals*, i.e., intermediate regions to explore for the target object.

This paper poses the 'Where to go?' (long-term goal selection) problem as a vision-based RL problem. Deep Reinforcement Learning (DRL) methods represent the state of the art in multiple domains such as video games, robot manipulation, and visual navigation. However, these methods, by themselves, often make it difficult to compute policies that generalize to unseen environments, even when the new environments are similar to those used to compute the policies. Strategies to address this limitation are based on learning better low-dimensional representations using auto-encoders, variational inference, contrastive learning, self-prediction, or data augmentation. Drawing on these insights, we introduce a DRL framework for ObjectNav task, which makes the following contributions:

- Incorporates an *Encoder Network* to learn rich, high-level features from an estimated semantic (domain) map, and an *Actor-Critic Network* that is trained with these features to provide effective long-term goals for the ObjectNav task.
- Incorporates (image) data augmentation in the context of the semantic domain map, and regularizes the Q(value) function in the update process of the encoder, actor, and critic networks in the framework, leading to better generalization and long-term goal selection.

We evaluate the framework's capabilities using benchmark (photo-realistic) scenes from the Gibson dataset in the AI Habitat 3D simulation environment [3] and show that our approach provides a 10-12% relative improvement in established performance measures in comparison with a state of the art baseline for ObjectNav task. Additional qualitative results and supporting material are available online: `https://user432.github.io/objnav-drq/`

II. Related Work

We review related work in Object Goal Navigation and the learning of representations through reinforcement.

Object Goal Navigation. There has been much work on developing learning algorithms for vision-based recognition and navigation. Data-driven deep (reinforcement) learning methods are state of the art for learning to map pixels directly to actions [4], [5]. These methods find it difficult to generalize to previously unseen scenarios since they do not build a representation of the environment. Methods that seek to promote better generalization with such methods construct an allocentric map that encodes semantic priors [2], [6], [7]. In this paper, we draw inspiration from this insight

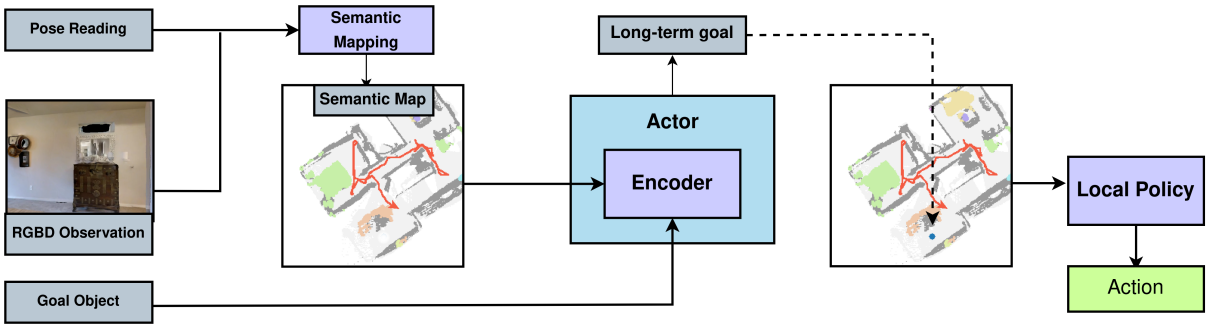

Fig. 1: Our framework consists of three main components. The *Semantic Mapping* module uses RGB-D observations and odometry pose readings to build an allocentric semantic map of the local world. Our *Encoder Network* extracts high-level features from the estimated semantic map. These features are used to train an actor-critic network which samples a long-term goal to reach the target object class instance efficiently. Analytical planners are used by the deterministic *Local Policy* to compute low-level navigation actions to reach any given long-term goal.

to pose the 'where to go?' decision as a vision-based deep RL problem, and build an explicit semantic map to set an effective long-term goal for the ObjectNav task.

Self-supervised Representation Learning. Many self-supervised methods have been developed in the computer vision literature to learn representations for different tasks; state of the art methods include contrastive methods [8], [9] as well as predictive methods [10], [11]. The learned representations have been shown to improve the performance of the corresponding supervised learning system, particularly when the amount of labeled training data available for the associated tasks is limited. The key insight from these methods, which we build on, is that good representations can be acquired from visual input through unsupervised learning or data augmentation, and that the learned representation can significantly improve performance on the associated task.

Representation Learning in RL. There has been a lot work on combining the ideas of self-supervised learning and RL in order to improve sample efficiency and performance. In particular, unsupervised learning of representation in conjunction with data augmentation methods have helped improve the efficiency of RL methods. For example, researchers have used auto-encoders [12] to improve efficiency of RL for visual information processing tasks [13]. Other self-supervised learning methods have helped link state-based and image-based RL, e.g., using contrastive learning [14], self-prediction [15], and augmented data [16], [17]. Our work in this paper is inspired by the DrQ system [16], which demonstrated the use of regularization to significantly improve the performance of the Soft Actor Critic method [18] when trained on image pixels in the context of tasks from the DeepMind Control (DMC) suite [19]. However, unlike their focus on solving continuous control tasks based on information encoded in image pixels, we pursue a modular approach for better generalization and efficiency, and focus on effectively sampling long-term goals by introducing a learned semantic map in the pipeline.

III. Problem Description and Approach

The ObjectNav task initializes a robot in a random location in a previously unexplored environment and requires the robot to navigate to the closest instance of a target object class (e.g., chair, bed). More specifically, the robot's inputs at each time step t include 640×480 RGB-D images (s_t), odometer readings (x, y, θ), and the target object category o. The robot has to navigate within $d_s = 1.0m$ of the target object class instance for successful task completion.

A. Framework Overview

Figure 1 provides an overview of our framework for the ObjectNav task. It has three key modules: *Semantic Mapping*, *Encoder Network*, and a deterministic *Local Policy*; the novelty is in the adaptation and proposed combination of these modules. The Semantic Mapping module builds a semantic map from the RGB-D images and pose observations (Section III-B). This map and the object goal are passed to the Encoder Network, which extracts high-level features and sends them to the Actor-Critic Network to select a long-term goal that improves performance on the ObjectNav task (Section III-C). An analytical planner is used to compute a deterministic local policy that provides low-level navigational actions to reach the long-term goal (Section III-D). We describe the individual modules below.

B. Semantic Mapping

Figure 2 summarizes the operation of the Semantic Mapping module, which is responsible for constructing and maintaining a semantic metric map m_t and pose of the robot p_t. The map m_t is a $K \times M \times M$ matrix where $M \times M$ is the map size. Each element of this spatial map represents a cell of size $25cm^2$ $(5cm \times 5cm)$ in the physical world. The number of channels in the semantic map is $K = C + 2$, where C is the total number of semantic categories. As shown in the top right of Figure 2, the first two channels represent obstacles and the explored area, and each of the remaining channels represent an object category. For each cell in the map, the channels indicate whether the corresponding location is an obstacle, has been explored, or

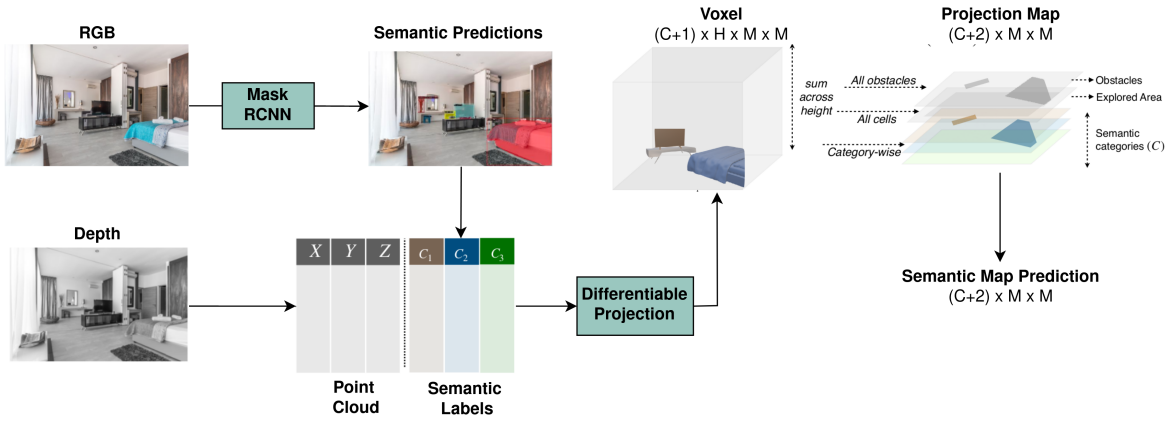

Fig. 2: **Semantic Mapping.** A sequence of RGB and Depth images are processed through a sequence of operations to produce a top-down Semantic Map.

contains an object of a particular category. The pose $p_t \in R^3$ denotes the (x, y) coordinates and the orientation of the robot at time t. The initial position of the robot is at the map's center, facing East at the start of the episode, i.e., $p_0 = (M/2, M/2, 0.0)$.

The procedure used to create the semantic map is a variant of the process followed by the SemExp method, the state of the art baseline for ObjectNav task [2]. First, we use a pretrained Mask R-CNN [20] model to estimate the semantic categories from the RGB (image) observations. Next, the depth observations are used to compute point-clouds that are registered in an allocentric coordinate system using the robot's sequence of poses $(p_0, ..., p_t)$. Each point in the point cloud is associated with the estimated semantic categories. A voxel representation is then built using differentiable geometric computations over each point in the point cloud. This voxel representation is converted into a $(C + 2) \times M \times M$ semantic map m_t. As stated earlier, channels 1 and 2 correspond to obstacles and explored areas, while the remaining channels correspond to the C object categories.

One of the innovations in our framework is the use of image augmentation methods with the semantic map to promote generalization. We explored four such methods:

- **Random Shift:** semantic maps of size 240×240 are padded on each side by four pixels (by repeating boundary pixels) and then randomly cropped back to 208×208 size.
- **Horizontal/Vertical Flip:** this method simply flips the semantic map either horizontally or vertically with probability 0.1.
- **Grayscale:** this method converts our RGB semantic map into a grayscale image.
- **Rotate:** this method rotates our semantic map by r degrees, where r is uniformly sampled from $[-5, +5]$.

In Section V, we discuss the experimental evaluation of these methods and justify the choice of the *random shift* method for data augmentation in our work.

C. Encoder Network

The estimated semantic map is passed to our encoder network, along with the robot's current and past locations and the target object class, to learn better representations. *This introduction of an encoder is a key innovation that, as we show later, contributes to a performance improvement in the ObjectNav task*. The encoder extracts high-level features that are used by the robot's actor network to generate a long-term goal, i.e., the region the robot should travel to next to look for an instance of the target object class.

The encoder network architecture comprises of 4 convolutional layers with 3×3 kernels and 32 channels [16]; ReLU activation is applied after each convolution layer. We use stride length of one everywhere. The output of these layers is fed into a single fully-connected layer normalized by LayerNorm [21]. Finally, we apply the hyperbolic tangent nonlinear transform to the 50-dimensional output of the fully-connected layer. We initialize the weight matrix of the fully-connected and convolutional layers through the orthogonal initialization method [22] and set the bias as zero.

The actor-critic network is set up to operate on the output of the encoder network. The actor and critic components have separate encoders, although they share the same weights in the convolution layers. Also, only the optimizer in the critic is allowed to update these weights, e.g., the gradients from the actor do not propagate to the shared convolution layers. We employ the clipped double Q-learning method [23] for the critic. In this method, each Q-function is parameterized as a three-layer multi-layer perceptron (MLP) with ReLU activations after each layer except the last one. The actor is also a three-layer MLP with ReLU activations; it outputs the mean and covariance for the diagonal Gaussian that represents the policy. The hidden layer's dimension is 1024 for both the critic and the actor.

As mentioned earlier, we regularize the Q-function in the update process of the relevant networks to promote generalization. Regularizing the Q-function of the critic network, for example, results in different transformations of the same semantic map to have the same Q-function values [16].

Our framework uses a replay buffer to store all the

transition states, including the semantic map, action, reward, next semantic map, goal category, and next goal category. A batch of transitions (s, a, r, s', g, g') are sampled from the replay buffer and used with the augmented semantic map as input to the encoder and actor-critic networks. A long-term goal is sampled once every 25 timesteps. During training, the decrease in the distance to the nearest target object instance after the completion of the episode is used as reward to revise the parameters of the networks.

Algorithm 1 describes the steps for updating the actor-critic network. There are two key differences from the standard soft actor-critic update steps. During the update of the critic network, we average the target Q-values over $K = 2$ image transformations:

$$y_i = r_i + \gamma \frac{1}{K} \sum_{k=1}^{K} Q_\theta(f(s'_i, v'_{i,k}), a'_{i,k}, g'_i), \quad \text{where } a'_{i,k} \sim \pi(\cdot|f(s'_i, v'_{i,k}), g'_i) \quad (1)$$

where state s can be treated as some function $f(\cdot)$ with parameters v. Also, we then average the Q-function itself over $M = 2$ image transformations:

$$\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{NM} \sum_{i=1,m=1}^{N,M} (Q_\theta(f(s_i, v_{i,m}), a_i, g_i) - y_i)^2 \quad (2)$$

where N is the sampled batch size. The other update steps remain unchanged.

D. Local Policy

The local policy enables the robot to navigate to the long-term goal g_t sampled by the actor after the actor-critic network is trained. It is obtained using the Fast Marching Method [24], which computes the shortest path from the present location to the long-term goal using the obstacle channel from the semantic map m_t. The robot invokes the local policy to execute deterministic actions and move along the computed shortest path. If a physical robot is performing the ObjectNav task, we could use (for example) a sampling-based motion planner for navigation; we do not do so here because that is beyond the scope of this paper.

IV. Experimental Setup

We use the Gibson benchmark dataset of scenes in the AI Habitat simulator [3] for our experiments. Gibson consists of scenes which are 3D reconstructions of real-world environments. We use the *train* and *val* splits of the *Gibson Tiny* dataset for training and testing respectively.We do not use the validation set for hyper-parameter tuning. From the train split, we only trained on 10 scenes at a time due to limitations on the computational resources available for use. Instead we repeated the training and evaluation five times. In each such repetition, we randomly sampled 10 scenes from the Gibson train split dataset, and trained our framework and the baseline for 1 $million$ frames. In each repetition, the trained models were evaluated on the same (standard) Gibson val split (with 5 $scenes$).

Algorithm 1 Steps for updating actor-critic network

Hyperparameters: Total number of environment steps T, mini-batch size N, learning rate λ, target network update rate τ, image transformation f, number of target Q augmentations K, number of Q augmentations M.
for each timestep = $1...T$ **do**
  $a_t \sim \pi(\cdot|s_t, g_t)$
  $s'_t \sim p(\cdot|s_t, a_t)$
  $D \leftarrow D \cup (s_t, a_t, r_t, s'_t, g_t, g'_t)$
  UpdateCritic(D)
  UpdateActor(D)
end for
procedure UpdateCritic(D)
  $(s_i, a_i, r_i, s'_i, g_i, g'_i)_{i=1}^{N} \sim D$ ▷ Sample a mini batch
  for each $i = 1...N$ **do**
    $a'_i \sim \pi(\cdot|f(s'_i, v'_{i,k}), g'_i),\ k = 1...K$
    $\hat{Q}_i = \frac{1}{K} \sum_{k=1}^{K} Q_{\theta'}(f(s'_i, v'_{i,k}), a'_{i,k}, g'_i)$
    $y_i \leftarrow r_i + \gamma \hat{Q}_i$
  end for
  $J_Q(\theta) = \frac{1}{NM} \sum_{i=1,m=1}^{N,M} (Q_\theta(f(s_i, v_{i,m}), a_i, g_i) - y_i)^2$
  $\theta \leftarrow \theta - \lambda \nabla_\theta J_Q(\theta)$ ▷ Update the critic
  $\theta' \leftarrow (1 - \tau)\theta' + \tau\theta$ ▷ Update the critic target
end procedure
procedure UpdateActor(D)
  $(s_i, g_i)_{i=1}^{N} \sim D$ ▷ Sample a mini batch
  $a_i \sim \pi(\cdot|f(s_i, v_{i,k}), g_i),\ k = 1...K$
  $J(\phi) = \log \pi(a_i|s_i, g_i) - Q_\theta(f(s_i, v_{i,k}), a_i, g_i)$
  $\phi \leftarrow \phi - \lambda \nabla_\phi J(\phi)$ ▷ Update the actor
end procedure

TABLE I: Overview of our framework's **hyperparameters**.

Parameter	Setting
Replay buffer capacity	40000
Minibatch size	8
Discount(γ)	0.99
Optimizer	Adam
Critic Learning rate	10^{-3}
Critic Q-function soft-update rate (τ_Q)	0.01
Critic encoder soft-update rate (τ_{enc})	0.05
Actor Learning rate	10^{-3}
Actor log std-dev bounds	[-20,2]
Init temperature	0.1
Features Dimension	50
Hidden Dimension	1024

As stated earlier, the observation space consisted of the RGB-D images of size $4 \times 640 \times 480$, and the success threshold $d_s = 1m$. The maximum episode length was 500 steps. For the target object categories, we used six object categories which were common between the Gibson dataset and the MS-COCO dataset: 'chair', 'couch', 'potted plant', 'bed', 'toilet' and 'tv' (television).

We experimentally evaluated the following hypotheses about the capabilities of our framework:

H1: Our framework provides better performance on the ObjectNav task in comparison with the baselines.
H2: Our framework provides better long-term goals than the

state of the art DRL baseline.

We evaluated these hypotheses by comparing our method with the following two baselines.

1) **Random:** The robot chooses actions randomly from the set of Habitat simulator actions: *move_forward, turn_right, turn_left,* and *stop*.
2) **Semantic Exploration (SemExp):** state of the art modular DRL method for ObjectNav, which won the ObjectNav challenge at CVPR 2020 [2]. It uses a Proximal Policy Optimization method to process a semantic map and target object class, directly providing a policy to sample long-term goals. We used the open-source implementation made available by the authors.

The experimental evaluation used well-established **measures** for the ObjectNav task in the research literature [2]:

1) **Success**: ratio of the number of successful episodes to total number of episodes. An episode is successful if the robot is within a fixed distance (1m) of an instance of the target object class.
2) **SPL** (Success weighted by path length): measures the efficiency of path taken by robot compared with optimal path; it is is computed as:

$$SPL = \frac{1}{N}\sum_{i=1}^{N} S_i . \frac{l_i}{max(p_i, l_i)}$$

where N is the number of test episodes, S_i is a binary success indicator, l_i is the length of shortest path to closest instance of target object from the robot's initial position, and p_i is the length of path traversed by robot.

3) **Distance to Success (DTS)**: denotes the distance between the robot and the permissible distance to target for success at the end of an episode.

$$DTS = max(\|x_T - G\|_2 - d_s, 0)$$

where $\|x_T - G\|_2$ is the $L2$ distance between robot and current goal location at the end of the episode; $d_s = 1.0m$ is the success threshold.

The full list of **hyperparameters** used in the corresponding algorithms is provided in Table I. We kept the settings identical for every experiment of our framework and conducted paired trials to evaluate the different methods (i.e., our framework, baselines) under similar conditions.

V. Experimental Results

Recall that training involved five repetitions of using $1\ million$ frames from 10 randomly sampled scenes from the benchmark dataset, with each the training models being evaluated in each such repetition on a separate set of five scenes. We conducted 200 evaluation episodes per scene, leading to a total of 1000 episodes in Gibson (with 5 *scenes* for evaluation). The corresponding quantitative results are summarized in Table II. In addition, Table III summarizes the performance of our framework during each of the five different splits of scenes, considering 10 scenes in each split. We observe that our framework outperformed the baselines; in particular, it achieved a success rate of 63.7% compared with the 57.9% of SemExp [2], the established state of the art method for the ObjectNav task.

TABLE II: **Results** of our framework compared with the baselines, when trained on the train split and evaluated on the val split of the Gibson Tiny dataset. Our framework performs substantially better on all measures.

Method	Success ↑	SPL ↑	DTS (m) ↓
Random	0.004	0.004	3.893
SemExp [2]	0.579	0.280	2.050
Our framework	**0.637**	**0.313**	**1.568**

TABLE III: **Results** of our framework in each of the five different splits (of 10 training scenes) from the Gibson dataset.

Method	Success ↑	SPL ↑	DTS (m) ↓
Split 1	0.637	0.313	1.568
Split 2	0.638	0.307	1.607
Split 2	0.616	0.301	1.777
Split 3	0.624	0.300	1.655
Split 4	0.631	0.308	1.665
Average	0.629	0.305	1.654

TABLE IV: Performance of our framework with each of the four different **image augmentation** methods described in Section III-B; *random shifts* provides the best balance between simplicity and performance.

Augmentation	Success ↑	SPL ↑	DTS (m) ↓
Flip	0.597	0.272	1.845
Grayscale	0.596	0.284	1.839
Rotate	0.604	0.284	1.706
Random Shift	**0.637**	**0.313**	**1.568**

Recall that we described four different image augmentation methods in Section III-B. Table IV summarizes the results of evaluating the effectiveness of each of these methods within our framework. These results indicated that the *random shifts* method provides a good balance between simplicity and performance. We thus only used this method for data augmentation in all our experiments, including those summarized in Table II and Table III.

Finally, Figure 3 shows a qualitative comparison between our framework and the SemExp baseline in a particular paired trial in which the robot had to locate an instance of the target object class 'bed'. Snapshots from the baseline are provided on the left, and those from our framework are shown on the right. Our framework resulted in the closest instance of the target object class being found much faster, primarily because it generated better long-term goals and supported better generalization. Additional qualitative results and supporting material are available online:
`https://user432.github.io/objnav-drq/`

VI. Conclusion and Future Work

In this paper, we described a modular deep reinforcement learning (DRL) method for the challenging object goal navigation (ObjectNav) task. Our key idea was to treat long-term goal selection (i.e., to determine 'where to go')

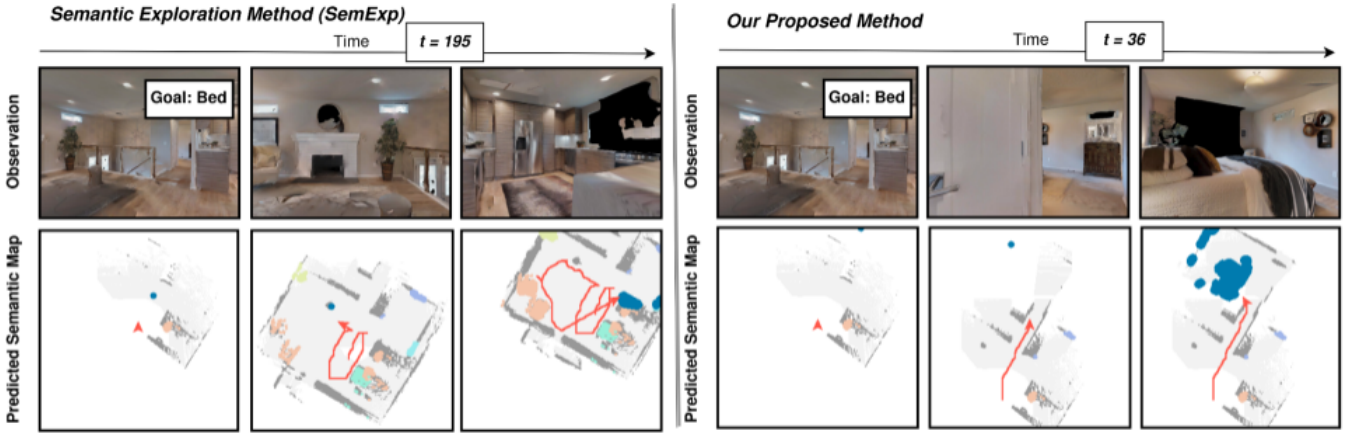

Fig. 3: **Qualitative Comparison with SemExp baseline:** Snapshots from a paired trial comparing our framework (right) with the SemExp baseline [2] (left) in a particular scene. With each method, the robot starts at the same location with the same target object class 'bed'. Our framework results in an instance of the target object class being found much faster.

as a DRL problem, using a combination of an Encoder network and an Actor-Critic network to extract high-level (abstract) representations from an estimated semantic map. In addition, we incorporated simple data augmentation methods and value function regularization methods to improve generalization. Experimental results obtained with the Gibson benchmark dataset in the AI Habitat 3D simulation environment demonstrated that our framework substantially improves performance on standard measures in comparison with a state of the art baseline for the ObjectNav task. Future work will further explore and seek to understand the abstract representations that have contributed to the improved performance. We will also consider other benchmark datasets of scenes for evaluation, and conduct ablation studies to understand the contribution of the actor-critic network to the framework's performance. Furthermore, we will explore the use of the networks trained in simulation on a physical robot operating in an indoor environment.

REFERENCES

[1] 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," *arXiv*, vol. 2006.13171, 2020.

[2] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *Neural Information Processing Systems*, 2020.

[3] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *International Conference on Computer Vision*, 2019.

[4] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," *arXiv*, vol. 1805.06066, 2019.

[5] 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," *International Conference on Robotics and Automation*, May 2017.

[6] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *International Conference on Computer Vision and Pattern Recognition*, 2017.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning*, 2020, pp. 1597–1607.

[7] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *International Conference on Learning Representations*, Apr 26-May 1, 2020.

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[10] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *IEEE/CVF Computer Vision and Pattern Recognition Conference*, 2021, pp. 9650–9660.

[11] X. Chen and K. He, "Exploring simple siamese representation learning," in *IEEE Computer Vision and Pattern Recognition Conference*, 2021, pp. 15 750–15 758.

[12] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *IEEE International Conference on Robotics and Automation*, May 2016.

[13] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 674–10 681.

[14] M. Laskin, A. Srinivas, and P. Abbeel, "CURL: Contrastive Unsupervised Representations for Reinforcement Learning," in *International Conference on Machine Learning*, 2020, pp. 5639–5650.

[15] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-efficient reinforcement learning with self-predictive representations," in *International Conference on Learning Representations*, 2021.

[16] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *International Conference on Learning Representations*, 2021.

[17] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," in *Neural Information Processing Systems*, 2020.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, 2018.

[19] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess, "dm˙control: Software and Tasks for Continuous Control," *Software Impacts*, vol. 6, 2020.

[20] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision*, Oct 2017.

[21] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv*, vol. 1607.06450, 2016.

[22] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *International Conference on Learning Representations*, 2014.

[23] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv*, vol. 1509.06461, 2015.

[24] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts." *National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.