Embodied Navigation with Auxiliary Task of Action Description Prediction

Supplementary Material

In this supplementary material, we provide additional details about:

1. Details of the tasks addressed in this paper.
2. Details of the architectures of the models to which the proposed method has been applied.
3. Details of training time.
4. Detials of the VideoLLaMA2 prompt.
5. Analysis of the proposed method through additional experiments.

1. Tasks

In this study, we deal with three major navigation tasks.

1.1. Object-goal Navigation (ObjNav)

Object-goal navigation is the most primary navigation task. In this task, the agent must consider where a given object is likely to be and navigate to the location based on visual information.

Formally, an episode is defined by $\{E, L_s, \theta_s, L_g, C_g\}$ where E is the scene environment, $L_s \in \mathbb{R}^2$ and $\theta_s \in [0, 2\pi)$ are the starting point and rotation of the agent respectively, $L_g \in \mathbb{R}^2$ is the goal position, and C_g is the goal object category. There are 21 categories that include, for example, bed or chair. At each time t, the agent has a visual observation, a first-person RGBD image $V_t \in \mathbb{R}^{480\times640\times4}$ and a goal category, and information on its own pose $p_t = (x, y, \theta) \in \mathbb{R}^3$. The action space is defined by $\mathcal{A} = \{MoveForward, TurnLeft, TurnRight, Stop\}$. Selecting $MoveForward$ will move forward by 0.25 m, and selecting $TurnLeft, TurnRight$ will rotate 30 degrees to the left or right respectively. Also, selecting $Stop$ will end the episode. Navigation is successful if $Stop$ is selected within a radius of 1.0 m of the goal object.

1.2. Vision-and-langauge Navigation (VLN)

In vision-and-language navigation [2], the agent must navigate in an indoor environment while observing visual information according to given a natural language instruction.

Formally, an episode is defined by $\{(\mathcal{V}, \mathcal{E}), L_s, L_g, \mathcal{W}\}$ where $(\mathcal{V}, \mathcal{E})$ is an undirected graph, $\mathcal{V}, \mathcal{E}$ are navigatable nodes and connectivity edges respectively, and $\mathcal{W} = \{w_1, \ldots, w_L\}$ is a sentence. At each time t, the agent has its current node $V_t \in \mathcal{V}$ and a visual observation, a panorama view composed by 36 d-dimentional image features $\mathcal{R}_t = \{\mathbf{r}_i^t\}_{i=1}^3 6 \in \mathbb{R}^{d\times36}$. The action space $\mathcal{A}$ is defined by the union of set $\{Stop\}$ and its neighboring nodes $\mathcal{N}(V_t)$. Navigation is successful if $Stop$ is selected within a radius of 3.0 m of the goal position L_g.

1.3. Semantic Audio-visual Navigation (SAVNav)

In semantic audio-visual navigation [4], the agent must navigate to a sounding object by observing visual and auditory information. The agent must infer what sound is being played from auditory information and navigate by considering where the goal object will likely be located, even if the sound stops.

As in object-goal navigation, an episode is formally defined by $\{E, L_s, \theta_s, L_g, C_g\}$. In this task, there are 21 sound categories that include, for example, the sound of water dropping in a sink or a bed creaking. The length of the sound differs depending on the category. At each time t, the agent has a visual observation, a first-person RGBD image $V_t \in \mathbb{R}^{128\times128\times4}$, an auditory observation, a binaural sound spectrogram $A_t \in \mathbb{R}^{65\times26\times2}$, and information on its own pose $p_t = (x, y, \theta) \in \mathbb{R}^3$. The action space is also defined by $\mathcal{A} = \{MoveForward, TurnLeft, TurnRight, Stop\}$. Selecting $MoveForward$ will move forward by 1 m, and selecting $TurnLeft, TurnRight$ will rotate 90 degrees to the left or right respectively. Also, selecting $Stop$ will end the episode. Navigation is successful if $Stop$ is selected within a radius of 1 m of the goal object.

1.4. Action Description Generation

In addition to solving these navigation task, we also solve the task of describing its own actions in natural language. In the conventional task, at time t, the robot receives the observation described above and outputs only the next action $a_t \in \mathcal{A}$. On the other hand, in this study, the robot outputs an action description sentence $D_t = (w_1^t, \ldots, w_{l_t}^t)$ that verbalizes its own actions and thoughts in addition to the next action a_t. Here, for each $i \in \{1, \ldots, l_t\}$, w_i^t represents the ith word of the sentence D_t.

2. Network Architecture Details

This section provides detailed descriptions of the architectures of the three models to which the proposed method has been applied.

2.1. SMT w/ DescRL for ObjNav

The architecture of the proposed method applied to SMT [7] is shown in Fig. 7. The major differences from SAVi w/ DescRL are that sound information is not input and there is no Goal Descriptor Network to predict the position $\hat{L}_t$ and category $\hat{C}_t$ of the goal object. Instead, a true goal category C is given. By embedding this C, we obtain the vector v_a^t for input to the policy and the vector $E_{\hat{w}_0^t}$ of the begining of

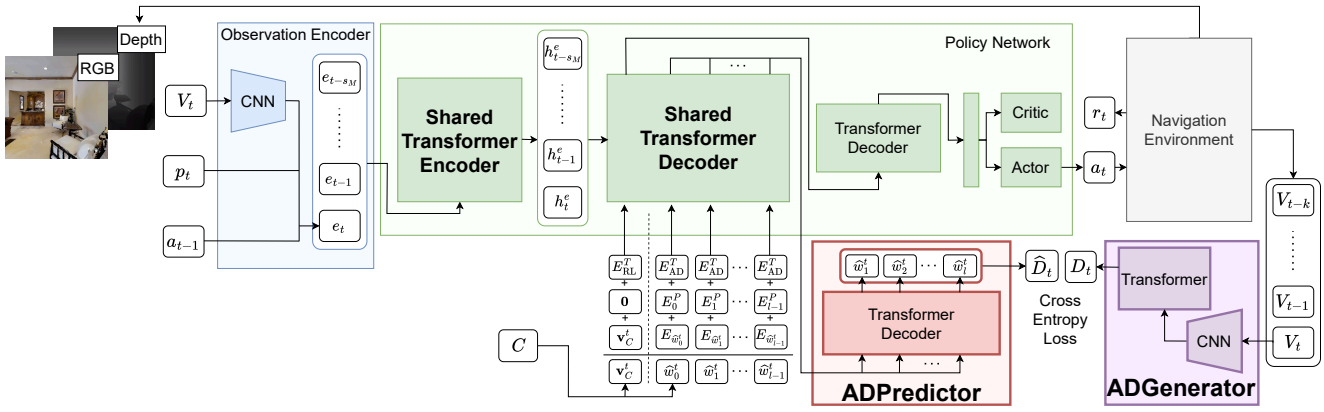


Figure 7. Overview of DescRL applied to SMT [7], which is a method for ObjNav. The reinforcement learning agent receives visual observation V_t, posture p_t, previous action a_{t-1} and the category of the goal C at time t and outputs the next action a_t and the action description $\hat{D}_t$.

Table 7. Details of the transformer architecture used within the SMT w/ DescRL. Hyperparameters for implementation in PyTorch. TF represents Transformer.

Module	d_model	nhead	num_layers	dim_feedforward	dropout
TF Encoder for ADGen	512	8	3	512	0.3
TF Decoder for ADGen	512	8	3	512	0.3
Shared TF Encoder	256	8	2	256	0.0
Shared TF Decoder	256	8	2	512	0.2
TF Decoder for Policy	256	8	1	256	0.0
TF Decoder for ADPred	256	8	1	512	0.2

sentence (BOS) token for input to ADPredictor. Based on these, an action a_t and an action description $\hat{D}_t$ are output.

ADGenerator: The hyperparameters for each module are as follows. CNN for processing images consists of 3 convolutional blocks, each with a kernel size of $8 \times 8, 4 \times 4, 3 \times 3$ and strides of $4 \times 4, 2 \times 2, 2 \times 2$. The respective output channels are $32, 64, 64$. An activation function ReLU is applied to each layer. The final layer is flattened and passed through a linear layer so that the output size is 508. The structure of each transformer is as shown in rows 1 and 2 of Table 7. Adam [8], where $\mathrm{lr} = 1.0 \times 10^{-4}$, $\mathrm{weight\_decay} = 5.0 \times 10^{-4}$, $\mathrm{betas} = (0.9, 0.98)$, $\mathrm{eps} = 1.0 \times 10^{-9}$, was used to update parameters.

RL model: The hyperparameters for each module are as follows. CNN for processing images consists of 3 convolutional blocks, all with a kernel size of 8×8 and a stride of 4×4, each with $32, 64, 64$ output channels. An activation function ReLU is applied to each layer. The final layer is flattened and passed through a linear layer so that the output size is 508. The structure of each transformer is as shown in rows 3 to 6 of Table 7. Adam [8], where $\mathrm{lr} = 2.5 \times 10^{-4}$, $\mathrm{weight\_decay} = 0.0$, $\mathrm{betas} = (0.9, 0.999)$, $\mathrm{eps} = 1.0 \times 10^{-5}$, was used to update the parameters.

Table 8. Details of the transformer architecture used within the ADGenerator and ADPredictor used in DUET w/ DescRL. Hyperparameters for implementation in PyTorch. TF represents Transformer.

d_model	nhead	num_layers	dim_feedforward	dropout
768	8	3	768	0.1

2.2. DUET w/ DescRL for VLN

The architecture of the proposed method applied to DUET [5] is shown in Fig. 8. ADPredictor and ADGenerator are newly added, and the rest of the structure is exactly the same as in the original paper [5].

ADGenerator: The detailed structure of the ADGenerator is shown in Fig. 9. The input is a topological map that is created sequentially based on the agent's observation. In order to process it, Coarse-scale Cross-modal Encoder [5] excluding cross-attention is used. The reason cross-attention is excluded here is that no instruction is input to the ADGenerator. Also, a transformer decoder is added to generate an action prediction. The detailed hyperparameters for this are shown in the row 1 of Table 8. AdamW [10], where $\mathrm{lr} = 1.0 \times 10^{-5}$, $\mathrm{betas} = (0.9, 0.999)$, $\mathrm{eps} = 1.0 \times 10^{-8}$, $\mathrm{weight\_decay} = 1.0 \times 10^{-2}$, was used to update the parameters.

RL model: ADPredictor is represented by a transformer decoder, the detailed structure of which is shown in the row 2 of Table 8. AdamW [10], where $\mathrm{lr} = 1.0 \times 10^{-5}$, $\mathrm{betas} = (0.9, 0.999)$, $\mathrm{eps} = 1.0 \times 10^{-8}$, $\mathrm{weight\_decay} = 1.0 \times 10^{-2}$, was used to update the parameters.

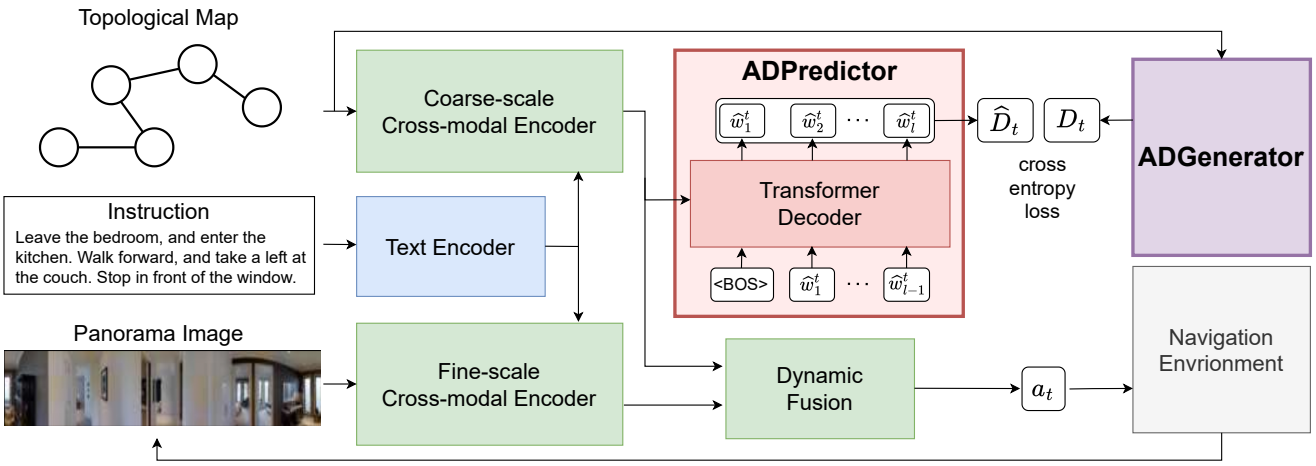


Figure 8. Overview of DescRL applied to DUET [5], which is a method for VLN. The imitation learning agent receives a topological map create by itself, a panoramic image and an instruction.

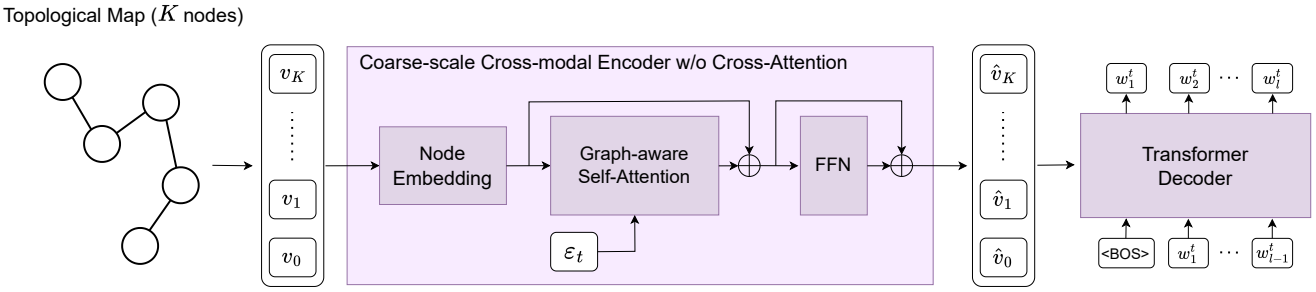


Figure 9. Structure of ADGenerator for VLN. It is based on Coarse-scale Cross-modal Encoder in DUET [5].

2.3. SAVi w/ DescRL for SAVNav

The detailed hyperparameters for each module are as follows. The CNN for processing sounds is also built with three convolutional blocks, each with a kernel size of $5 \times 5, 3 \times 3, 3 \times 3$ and strides of $2 \times 2, 2 \times 2, 1 \times 1$. The respective output channels are $32, 64, 64$. An activation function ReLU is applied to each layer except the final layer. The final layer is flattened and finally passed through a linear layer so that the final output size is 128. The CNN for processing the image also consists of three convolutional blocks, each with a kernel size of $8 \times 8, 4 \times 4, 3 \times$ and strides of $4 \times 4, 2 \times 2, 2 \times 2$. All other details are the same as SMT w/ DescRL.

3. Training Time

Training times for experiments are shown in Table 9. Our methods take a long runtime. This is mainly due to the autoregressive generation of natural language sentences by ADGenerator.

Prompt:
Describe how the camera wearer moves around the indoor environment in 40 words or less. Follow the format of the output as shown in the example below.

[Example of output format]
Turn left and go down the steps on the left. Turn right and wait near the unicycle.
[/Example of output format]

VideoLLaMA2:
Move forward and enter the modern kitchen. Turn left to explore the dining area with a large table and chairs. Proue left to find the living room with a comfortable couch and a large window.

Figure 10. Exemplar input and output of VideoLLaMA2 [6].

4. VideoLLaMA2 Prompt

The prompt provided to VideoLLaMA2 is shown in the top of Fig. 10. The example output is shown in the bottom of Fig. 10.

Table 9. Training time. The unit is GPU hour.

Section	Method	Time
??	SMT [7]	160
??	ADGenerator	96
??	ADPredictor for SMT	220
??	SMT w/ DescRL	384
??	DUET [5]	24
??	ADGenerator for DUET	20
??	ADPredictor for DUET	2.6
??	DUET w/ DescRL	24
??	ScaleVLN [12]	24
??	ADGenerator for ScaleVLN	10.5
??	ADPredictor for ScaleVLN	0.5
??	ScaleVLN w/ DescRL	24
??	SAVi [4]	76
??	ADGenerator for SAVi	96
??	ADPredictor for SAVi	300
??	SAVi w/ DescRL	180
??	KSAVEN [11]	64
??	ADGenerator for KSAVEN	96
??	ADPredictor for KSAVEN	384
??	KSAVEN w/ DescRL	208
??	Other auxiliary tasks	76
??	ADPredictor for VideoLLaMA2 [6]	192
??	DescRL w/ VideoLLaMA2	192
??	ADPredictor for Qwen2.5-VL [3]	240
??	DescRL w/ Qwen2.5-VL	240

5. Additional Experiments

5.1. Incorporating the action description to zero-shot baseline

We investigated whether the proposed method is effective even for zero-shot navigation methods. We incorporated the action-description into NavGPT [14], a typical method for solving VLN with zero-shot. We used Llama3.3 70B for LLM. In the case of the future action description and past action description, we added to the prompt "Describe how you should move around the indoor environment to achieve the given instruction in the future in 40 words or less." and "Describe how you have already moved around the indoor environment since you started navigating in 40 words or less.", respectively. As shown in Table 10, the proposed method leads to a substantial improvement in both SR and SPL. This result indicates that our method is also effective in zero-shot settings

5.2. Quantitative Evaluation of Action Description

Evaluation method: We perform a quantitative evaluation of the generated action descriptions. We input the generated action descriptions into a VLN model and check how well the VLN model follows the trajectory of the original

Table 10. Incorporating the action description to zero-shot baseline.

Method	NE ↓	SR ↑	SPL ↑
NavGPT	**8.35**	15.0	12.9
w/ Past-DescRL	8.75	19.0	15.2
w/ Past-Future-DescRL	8.46	**22.0**	**18.7**
w/ F-DescRL	8.42	20.0	17.1

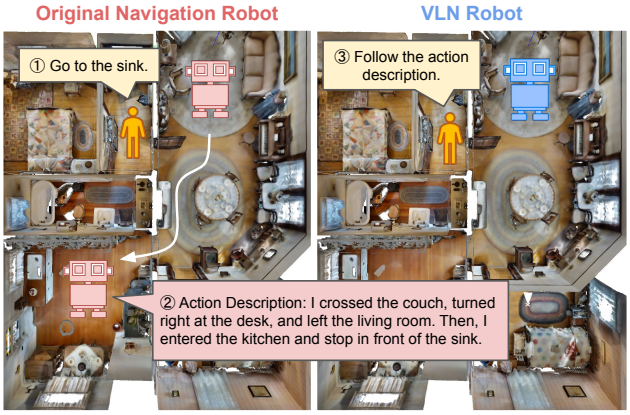


Figure 11. Overview of the method for qualitative evaluation of action descriptions.

Table 11. Quantitative evaluation of action descriptions on VLN.

	Method	Model	Val Seen		Val Unseen	
			VLN-SR ↑	VLN-SPL ↑	VLN-SR ↑	VLN-SPL ↑
1)	Human	-	82.57	77.09	78.63	69.15
2)	ADGen	DUET [5]	73.95	**68.18**	**73.82**	**62.50**
3)	ADPred	DUET [5]	**74.93**	67.11	72.16	60.15
4)	ADGen	ScaleVLN [12]	**71.20**	**65.01**	65.26	53.34
5)	ADPred	ScaleVLN [12]	64.74	56.83	**67.48**	**56.03**

navigation model (Fig. 11). The first step in the process is to determine the original navigation model. Specifically, first, in the original navigation task, we obtain all observations $o_1, \ldots, o_T$ in the Past settings or the first few observations in the Future setting by selecting actions according to the optimal policy (Fig. 11 left). Second, by inputting them to an agent, ADPredictor generates an action description $\hat{D}$. Finally, by inputting this action description $\hat{D}$ into the learned vision-and-language model, we check whether this model can follow the optimal trajectory followed by the original agent (Fig. 11 right). The SR and SPL of this VLN model to the goal of the original agent are called VLN-SR and VLN-SPL, respectively. These values indicate how well the system is able to generate action description sentences that are faithful to its own actions. It should be noted, however, that the poor performance of the learned vision-and-language model may result in poor performance

Table 12. Quantitative evaluation of action desciptions on SAVNav. CNN-TF, VL2, and FT VL2 mean using ADGenerator based on CNN+Transformer, VideoLLaMA2 [6] and VideoLLaMA2 fine-tuned by R2R dataset. Step 1 represents the result at step 1 of phase 2, i.e., when the agent has not yet learned navigation. Here, each method is evaluated using CMA [9] on VLN in Continuous Environment [9]. Note that Human results are on the R2R dataset, so there is a domain gap between VLN and SAVNav.

			Heard		Unheard	
	Method	ADGen Model	VLN-SR ↑	VLN-SPL ↑	VLN-SR ↑	VLN-SPL ↑
1)	Human	-	30.2	28.2	30.2	28.2
2)	ADGen	CNN-TF	7.7	6.4	7.7	6.4
3)		VL2	6.6	5.6	6.6	5.6
4)		FT VL2	6.9	5.8	6.9	5.8
5)	Past ADPred	CNN-TF : Step 1	4.5	3.5	4.3	3.1
6)		CNN-TF : Step 2	5.5	4.6	6.6	5.4
7)		VL2 : Step 1	6.4	5.2	5.6	4.5
8)		VL2 : Step 2	6.6	5.6	6.7	5.4
9)	Future ADPred	CNN-TF : Step 1	5.2	3.9	5.4	4.1
10)		CNN-TF : Step 2	7.0	5.8	6.4	5.4

of VLN-SR and VLN-SPL.

Results: We tested the VLN agent and the SAVNav agent, comparing the performance of ADGenerator and ADPredictor, and comparing the performance of ADPredictor in Step 1 and Step 2 of Phase 2, to see how navigation learning can affect action description generation learning.

In the VLN setting (Table 11), we found that ADPredictor is better performance in the Val Unseen than ADGenerator in the ScaleVLN, where navigation performance is higher. This suggests that better navigation learning is associated with better action description generation learning. Also, when comparing the performance of ADGenerator and ADPredictor in Table 11, ADGenerator tends to perform better in DUET-based cases (rows 2, 3), while in ScaleVLN-based cases (rows 4, 5), ADGenerator performs better in Val Seen and ADPredictor performs better in Val Unseen. This may seem inconsistent at first glance, but from the perspective explained below, this is not inconsistent. When we look at the ScaleVLN-based ADGenerator result (row 4), the value is excessively high for Val Seen and excessively low for Val Unseen. This indicates that overfitting has occurred. If the simultaneous learning of navigation and action description generation reduces the gradient bias, the ADPredictor would have better generalization performance than the ADGenerator. This would result in lower performance for Val Seen but higher performance for Val Unseen. Our conclusion is that when the generalization performance of the ADGenerator is high, the results exhibit a trend similar to those based on DUET. Conversely, when the generalization performance of the ADGenerator is low, the results align more closely with those based on ScaleVLN.

In the SAVNav setting (Table 12), we found that step 2 is consistently better performance than step 1 overall. This suggest that the introduction of navigation learning is associated with better action description generation learning.

Table 13. Comparison of varying the size of the image observed by the agent. Size represents the size of the image. N_{SD} indicates the number of sharing decoders.

			Heard					Unheard				
Method	Size	N_{SD}	SR↑	SPL↑	SNA↑	DTG↓	SWS↑	SR↑	SPL↑	SNA↑	DTG↓	SWS↑
SAVi [4]	128	-	31.6	28.5	24.6	11.8	12.5	24.7	22.4	18.9	11.8	10.2
	336	-	33.1	30.3	26.6	11.3	13.5	22.1	19.8	17.4	11.4	8.7
Ours	128	0	**37.6**	**32.6**	**28.0**	**7.8**	18.6	26.6	23.7	20.3	**8.7**	11.2
	336	0	37.1	32.3	26.5	8.5	**20.1**	29.5	25.1	20.1	9.0	**15.9**
	128	2	37.4	32.4	**28.0**	8.4	19.1	**31.4**	**26.9**	**22.5**	**8.7**	15.1
	336	2	33.3	29.7	24.2	8.6	15.2	25.2	21.5	17.0	9.4	12.1

Table 14. Quantitative evaluation of action description on SAVNav. Size represents the size of the image. Here, each method is evaluated using CMA [9] on VLN in Continuous Environment [9]. Note that Human results are on the R2R dataset, so there is a domain gap between VLN and SAVNav.

			Heard		Unheard	
Type	Size	N_{SD}	VLN-SR ↑	VLN-SPL ↑	VLN-SR ↑	VLN-SPL ↑
-	Human	-	30.2	28.2	30.2	28.2
ADGen	128	-	7.7	6.4	**7.7**	**6.4**
	336	-	6.4	5.2	6.4	5.2
Past ADPred	128	0	6.8	5.8	7.0	5.8
	336	0	**8.7**	**7.2**	6.8	5.6
	128	2	5.5	4.6	6.6	5.4
	336	2	7.6	6.1	6.5	5.4

5.3. Increasing Image Size

In the original SAVNav paper [4], the input image size is 128×128. Here, however, this was scaled up to 336×336 to investigate the impact of resolution on action descriptions. It is expected that increasing the image quality will improve the performance of object recognition and environment understanding, thereby increasing generating action descriptions performance.

Quantitative evaluation of navigation: The impact on navigation is shown in Table 13. First, the proposed method outperforms the baseline method even as image size improves. Second, we also found that when the number of decoder shares is 0, the performance tends to improve further in the Unheard setting, while when the number of decoder shares is 2, the overall performance decreases. When decoders as well as encoders are shared, it becomes more important to embed the observation information in a space common to the policy and ADPredictor. In addition, when the number of pixels is large, the input information increases and the process of processing it becomes more complex. This makes it more difficult to embed the observed information into the space common to the policy and the

Table 15. Comparison of Past-DescRL with and without Replay Buffer for ADPredictor (RB-ADP). N_{SD} indicates the number of sharing decoders.

		Heard					Unheard				
RB-ADP	N_{SD}	SR↑	SPL↑	SNA↑	DTG↓	SWS↑	SR↑	SPL↑	SNA↑	DTG↓	SWS↑
×	0	**37.6**	**32.6**	**28.0**	**7.8**	**18.6**	26.6	23.7	**20.3**	8.7	11.2
✓	0	34.8	31.1	23.9	7.9	18.3	**29.1**	**25.4**	19.9	**8.5**	**14.7**
×	2	**37.4**	**32.4**	**28.0**	8.4	**19.1**	**31.4**	**26.9**	**22.5**	8.7	15.1
✓	2	34.7	30.4	25.6	**7.9**	16.1	31.2	26.5	21.7	**8.4**	**16.0**

Table 16. Quantitative evaluation of action descriptions on SAV-Nav. Here, each method is evaluated using CMA [9] on VLN in Continuous Environment [9].

		Heard		Unheard	
RB-ADP	N_{SD}	VLN-SR ↑	VLN-SPL ↑	VLN-SR ↑	VLN-SPL ↑
×	0	**6.8**	**5.8**	**7.0**	**5.8**
✓	0	5.7	4.6	5.8	4.6
×	2	5.5	4.6	6.6	5.4
✓	2	**8.2**	**6.9**	**6.9**	**5.7**

ADPredictor, which may describe why the performance did not improve.

Quantitative evaluation of action descriptions: The impact on action descriptions is shown in Table 14. We found that in the heard setting, larger size increased action description performance, while in the unheard setting it had little effect. This suggests that for action description generation, it is important to improve recognition performance not only for visual information but also for auditory information.

5.4. Replay Buffer for ADPredictor

In the above experiments, since the policy was updated using the on-policy method DD-PPO [13], ADPredictor was also updated using the on-policy method. In other words, the batch for updating ADPredictor is collected by the current policy $\hat{\pi}_k$. In the Past setting, the data d_t for updating the ADPredictor consists of the observations $\{o_0, \ldots, o_t\}$ of the agent, and $X(o_0, \ldots, o_t)$, which is a linguistic representation of them by the ADGenerator. Suppose we have data $d_{t+1} = (\{o_0, \ldots, o_t, o_{t+1}\}, X(o_0, \ldots, o_t, o_{t+1}))$ that is off by one step. At this time, there is little difference between $X(o_0, \ldots, o_t)$ and $X(o_0, \ldots, o_t, o_{t+1})$. Thus, two data d_t, d_{t+1} that are off by one step have almost no difference, since both the input and output are almost the same. Therefore, when updating ADPreditor using the on-policy method, data with almost no difference like this are frequently included in the batch. This raises the concern that the diversity within the batch will be insufficient, the gradient will be biased, and ADPredictor will fall into a locally optimal solution.

Here, since ADPredictor does not necessarily need to be updated in an on-policy manner, Replay Buffer for ADPredictor (RB-ADP) can be introduced (Fig. 12 (b)). That is, data $D^{\hat{\pi}_k}$ collected by the current policy $\hat{\pi}_k$ is stored in RB-ADP, and ADPredictor is updated with batch data S_k sampled from this RB-ADP. This allows the batch for updating ADPredictor to include a variety of episode data obtained by all past polices $\hat{\pi}_1, \ldots, \hat{\pi}_k$.

Quantitative evaluation of navigation: The impact on navigation is shown in Table 15. First, we found that in the Unheard setting, the performance was better with RB-ADP when the decoder was not shared, but when the decoder was shared, there was little change in performance. This may be because the lack of diversity was not a major issue when the decoders were shared, since the ADPredictor could use the gradient for policy. Second, in the Heard setting, we found that RB-ADP reduced performance. This suggests that overfitting was suppressed due to the diversity created in the batch.

Quantitative evaluation of action descriptions: The impact on action descriptions is shown in Table 16 We found that when the decoder is shared, performance is better with RB-ADP, but when the decoder is not shared, performance is better without RB-ADP. This may be due to the better quality of data collected on the current policy than on past policies. It is expected to be difficult for the ADGenerator to verbalize trajectories collected by near-random, unlearned policies. Thus, the RB-ADP may contain a lot of poor quality data. This may have worsened action description performance when the decoder is not shared. On the other hand, when the decoder is shared, ADPredictor may have mitigated this problem because information on the current policy can be obtained from the policy network.

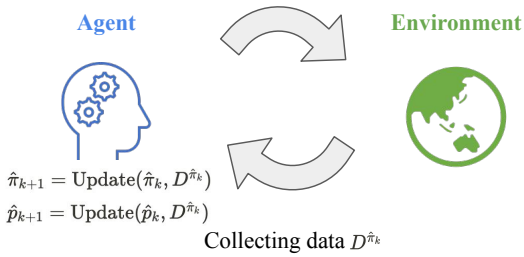
5.5. Additional Qualitative Evaluation of Action Description

5.5.1 Finetuned VideoLLaMA2

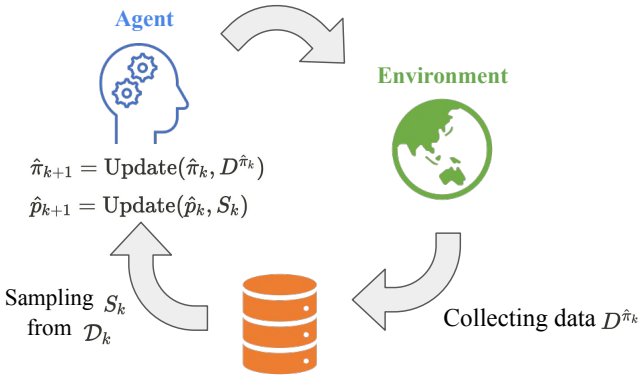
Figure 13 shows the results of fine-tuning VideoLLaMA2 on the R2R dataset using QLoRA, an efficient approach for fine-tuning large models. Although the sentence is mostly accurate, it occasionally commits word-level errors. Indeed, although both a statue and a fireplace appear in the video, the statue is not of a woman, nor does the sequence involve passing through a fireplace.

5.5.2 ADGenerator

Figure 14 shows the qualitative evaluation of generating action description performance of ADGenerator consisting of CNN and transformer. We found that there are many mistakes between right and left. Also, although there is no refrigerator, we found that the silver object near the kitchen is associated with a refrigerator in step 13. In addition, rep-



(a) w/o Replay Buffer for ADPredictor



(b) w/ Replay Buffer for ADPredictor

Figure 12. Overview of replay buffer for ADPredictor. $\hat{\pi}_k$ and $\hat{p}_k$ represent the policy and ADPredictor updated k times, respectively. Also, $\text{Update}(A, B)$ represents updating A with the batch B.

Figure 13. When a video composed of these images is input, the fine-tuned VideoLLaMA2 generates the sentence shown in the lower right.

etition of sentences and the same sentences being output between similar steps were observed.

5.5.3 Vision-and-language Navigation

Figure 15 and **??** show the results of the qualitative evaluation of action descriptions of ScaleVLN [12] w/ Past-DescRL on VLN. Here, it is evaluated with Val Unseen dataset. Figure 15 shows the results for successful navigation and Fig. **??** shows the results for failed navigation. In Fig. 15, the agent generated the sentence "turn right" even though it did not actually turn right. Like this, hallucinations regarding its own behavior were sometimes observed. In the episode in Fig. **??**, although no stairs were observed in the observations, the word "stairs" is often found in action descriptions. This may be because the agent has learned the common sense that stairs are often found in hallways.

5.5.4 Semantic Audio-visual Navigation

Figure 16 and 17 show the results of the qualitative evaluation of action desciptions of SAVi [4] w/ Past-DescRL on SAVNav. Here, it is evaluated with Unheard dataset. Figure 16 shows the results for successful navigation and Fig. 17 shows the results for failed navigation.

In SAVNav, we found that the agent's ability to generate action descriptions was comparatively low. In particular, errors such as object misrecognition and left-right misrecognition were common. For example, the word "bed" is often seen in Step 3, 4, 5, 6 in Fig. 16 even though it does not exist. This is assumed to be due to the misrecognition of a desk as a bed, which is slightly observed in Step 1. Also, "enter the room in your right" in Step 22 and "turn right" in Step 23 in Fig. 16 show that the left and right are misrecognized. In addition, action descriptions that have nothing to do with the agent's actions, such as the action description in Step 18 in Fig. 16 and "walk down the stairs" in Step 19 in Fig. 17, or hallucination, was also frequently observed.

The reasons for these results are, first, that the generation of action descriptions is a more difficult task in SAVNav than in VLN because of the large amount of observed information. In SAVNav, the number of steps is relatively high because the actions are low-level. As a result, long time-series information must be properly converted into

language, making the task of generating action descriptions difficult. In addition, while in VLN the input is given in the form of a graph, in SAVNav the agent deals with low-level actions, which may have made it difficult to recognize space. Furthermore, in VLNs, agents observe visual information for 360 degrees, whereas in SAVNav, only 90 degrees of visual information is observed. As a result, objects are often seen only partially rather than completely, which is thought to have increased object misrecognition.

A possible way to address these problems is to increase the amount of instructions data created by humans. By increasing the data, it is expected that learning difficult tasks will become possible, and misrecognition will be reduced. Alternatively, knowledge distillation from a higher-performing VLM could be improved by the proposed method.

5.5.5 Trajectory Comparison

Here, as in the quantitative evaluation of action descriptions, first, the SAVNav w/ Past-DescRL agent navigated and the action description was input to the learned VLN agent (BEVBERT [1]) to perform a qualitative evaluation to see if it could follow the same trajectory. We qualitatively check how well the learned VLN agent is able to follow the SAVNav agent, and qualitatively check how faithful the description is. If the accuracy of BEVBERT is 100% (actually 58.2%) and the SAVNav agent is able to provide a completely faithful description of actions, then these two trajectories are considered to be the perfectly same.

The results shown in Fig. 18 indicate that the two trajectories often do not coincide perfectly. This is often likely due to the fact that BEVBERT does not perform 100% accurate. On the other hand, if we look at the bottom left of Fig. 18, the VLN model turns right for a moment, then turns left and enters another room. The action description of SAVNav agent at this point is "go straight until you get to the couch. turn right and go out the door. wait near the bed." In reality, the agent did not go through the door after turning right, but it did output "turn right and go out the door." Therefore, it is assumed that this caused the VLN model to momentarily turn to the right, then turn back to the left, take the door, and enter the room on the left. In other words, in this case, the VLN model chose the wrong path because the SAVNav agent's action description was incorrect. Thus, the limited action description performance of the agent may be one of the reasons why these two trajectories do not match.

References

[1] Dong An, Yuankai Qi, Yangguang Li, Yan Huang, Liang Wang, Tieniu Tan, and Jing Shao. Bevbert: Multimodal map pre-training for language-guided navigation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2737–2748, 2023. 8

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv*, 2025. 4

[4] Changan Chen, Ziad Al-Halah, and Kristen Grauman. Semantic audio-visual navigation. In *CVPR*, 2021. 1, 4, 5, 7, 11, 12, 13

[5] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16537–16547, 2022. 2, 3, 4

[6] Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, and Lidong Bing. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv*, 2024. 3, 4, 5

[7] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 538–547, 2019. 1, 2, 4

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 2

[9] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 104–120. Springer, 2020. 5, 6

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 2

[11] Gyan Tatiya, Jonathan Francis, Luca Bondi, Ingrid Navarro, Eric Nyberg, Jivko Sinapov, and Jean Oh. Knowledge-driven scene priors for semantic audio-visual embodied navigation. *arXiv preprint arXiv:2212.11345*, 2022. 4

[12] Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12009–12020, 2023. 4, 7, 10

[13] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 6

[14] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *AAAI*, 2024. 4

Figure 14. Qualitative evaluation of generating action description performance of ADGenerator consisting of CNN and transformer.

Figure 15. Qualitative evaluation of generating action descpritions perforcemance of ScaleVLN [12] w/ Past-DescRL on VLN. In this episode, the agent successfully navigated. In each step, the agent observes each panoramic image and generates the sentences shown above it. Red arrows indicate actions taken by the agent. Areas marked in green are those that can be judged to be correct action descriptions.

Figure 16. Qualitative evaluation of generating action description performance of SAVi [4] w/ Past-DescRL on SAVNav. In this episode, the agent successfully navigated. The agent observes the RGBD image on the left. The top-down map on the right is not observed. The red and light blue dots on the map represent the true goal location and the goal location predicted by the agent in the Goal Descriptor Network in SAVi, respectively. The red and light blue letters represent the true goal category and the goal category predicted by the Goal Descriptor Network in SAVi, respectively.

Figure 17. Qualitative evaluation of generating action description performance of SAVi [4] w/ Past-DescRL on SAVNav. In this episode, the agent failed to navigate. The goal category is cabinets, but we can see that it has stopped in front of another cabinet. The action description generation does not learn the specific type of cabinet, such as shape. Therefore, we believe that the proposed method cannot solve a problem such as stopping in front of different instances of the same category.

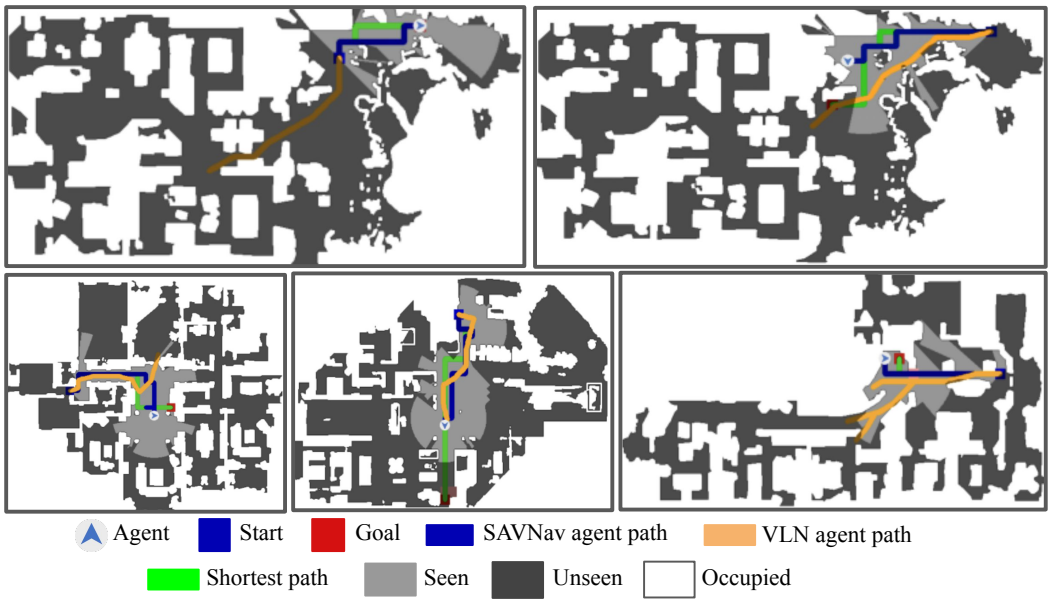


Figure 18. Comparison of SAVi [4] w/ Past-DescRL trajectories in SAVNav and trajectories of a learned VLN agent following the output of the ADPredictor. The blue trajectories represent the trajectories of SAVi w/ Past-EPRL, and the orange trajectories represent the trajectories of the VLN agent.