Source-Free Elastic Model Adaptation for Vision-and-Language Navigation

Mingkui Tan, Peihao Chen, Hongyan Zhi, Jiajie Mai, Benjamin Rosman, Dongyu Ji, Runhao Zeng

***Abstract*—Vision-and-Language Navigation (VLN) requires an agent to follow given instructions to navigate. Despite the significant progress, the model trained on seen environments has a performance drop on unseen environments due to distribution shift. To improve the generalization, existing method attempts to apply test-time adaptation to VLN. However, it needs to access the training data and all testing data for updating the model before inference. The setting is not suitable for the real application because it is hard for the agent to access training data and all testing data when the agent is applied in a new environment. In this paper, we consider a more practical setting with source-free and online-inference test-time adaption. In other words, the model can only access one testing sample for test-time adaptation. In this setting, the model may suffer from catastrophic forgetting of the learned knowledge and unstable parameter update issues. To solve these challenges, we propose an elastic adaptation model (EAM) that consists of an auxiliary decision model and a sample replay mechanism. We use the online testing samples to adapt the auxiliary decision model to new environments, which cooperates with the frozen original model to make better action decisions. The sample replay mechanism stores the historical testing samples to make the adaptation process more stable. Our method is model-agnostic and is effortless to be applied to most existing methods. Experimental results show that our method achieves stable performance improvement based on three existing methods on three VLN benchmark datasets.**

***Index Terms*—Multi-Modal, Vision-and-Language Navigation, Test-Time Adaptation.**

I. Introduction

In vision-and-language navigation (VLN) task, an agent is required to follow natural language navigation instructions and navigate to a specific target location [2]. Over recent years, it has gained significant progress. However, existing methods still suffer from the problem of insufficient generalization ability. Specifically, there are large differences between seen scenes in the training stage and unseen scenes in the testing stage. Due to the data distribution shift, the model that performs well at training has a certain performance drop at testing [3]. To improve the generalization ability, researchers propose diversified data augmentation methods to expand the training data [4]–[7]. However, it is still hard to represent the target data distribution accurately, and it requires extra training costs. Recently, researchers proposed a Test-Time Adaptation (TTA) method to solve the problem of distribution shift in the field of image classification [8]–[11]. At test time, the model parameters are no longer fixed but adjusted dynamically to adapt to the target data distribution. It inspires us to apply TTA method to VLN task.

The existing method [1] has made a preliminary attempt on it. To overcome the distribution shift, they devise a two-stage training strategy. First, they train the model with supervised and self-supervised objectives simultaneously over training data. Second, they use the self-supervised module to further update the model over testing data before inference. But the existing method still has the following shortcomings. 1) It designs a self-supervised task, which needs to retrain the model using the training data. However, the training data may be inaccessible due to privacy security issues in practical scenarios. 2) It needs to obtain the complete test samples to adjust the model parameters, and then conduct the final inference using the adjusted model. However, it is usually hard to obtain all test samples in advance especially in the actual navigation process.

In order to solve the above problems, we consider a more practical setting to apply TTA method. The setting has the following characteristics, which are also shown in Figure 1. 1) **Source-Free**. We only obtain the trained model and do not alter the training process, and thus our method can be applied to most existing VLN methods. 2) **Online-Inference**. The test samples arrive as data streams, with only a single test sample available at each testing step, which is more in line with reality. However, it is challenging to adapt the model parameters under this setting. 1) The model may suffer a catastrophic forgetting of the learned knowledge in the parameters adaptation process. 2) It is hard for a single test sample to represent the target data distribution accurately. It will lead to unstable updates when only using the single test sample to adjust the model parameters directly.

We propose an elastic adaptation model (EAM) method to solve the above challenges. Specifically, we design an auxiliary decision model and combine it with the original model to form a two-branch structure. It avoids the forgetting of old

This work was partially supported by National Natural Science Foundation of China (NSFC) under Grants 62202311 and 62072190, the Shenzhen Natural Science Foundation (the Stable Support Plan Program) under Grant 20220809180405001, Excellent Science and Technology Creative Talent Training Program of Shenzhen Municipality under Grant RCBS20221008093224017, Ministry of Science and Technology Foundation Project 2020AAA0106900, Key-Area Research and Development Program Guangdong Province 2019B010155002, the Guangdong Basic and Applied Basic Research Foundation under Grants 2023A1515011512.

Mingkui Tan, Peihao Chen, Hongyan Zhi, and Dongyu Ji are with the School of Software Engineering, South China University of Technology, Guangzhou, China. E-mail: mingkuitan@scut.edu.cn

Runhao Zeng is with Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen, China, and also with Guangdong-Hong Kong-Macao Joint Laboratory for Emotional Intelligence and Pervasive Computing, Shenzhen MSU-BIT University, Shenzhen, China.

Jiajie Mai is with City University of Hong Kong, Hongkong, China.

Benjamin Rosman is with School of Computer Science and Applied Mathematics, the University of the Witwatersrand, Johannesburg South Africa.

Mingkui Tan, Peihao Chen are corresponding authors.

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2025.3535356

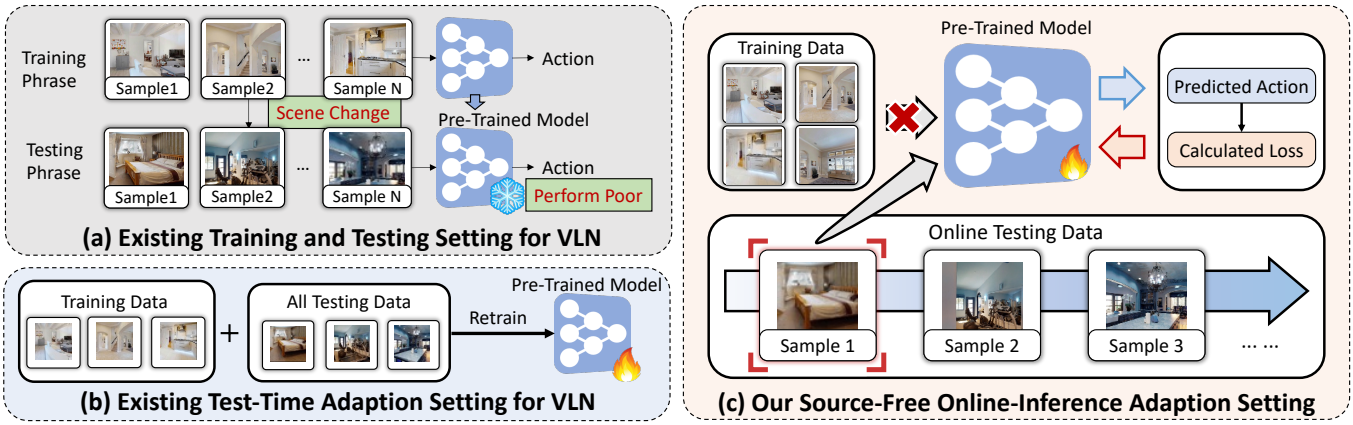

Fig. 1: (a) Since the large discrepancy between training and testing scenes, directly using the trained model for testing performs poorly. (b) Existing method [1] attempts to mitigate this gap by retraining the model with a visual consistency self-supervised loss on training data and all testing data. Although performance improvement is observed, this setting requires access to source data, which is not available in the adaption phrase in most cases. Best, when we apply a trained model for VLN, the testing data is observed in an online manner. It is hard for us to access all testing data at once. (c) We consider a practical setting to apply elastic model adaptation to VLN. The adaptation of the pre-trained model does not depend on the training data. The testing data is provided in the form of data streams. The model parameters are adjusted online based on the current test sample.

knowledge by keeping the original model and promotes the learning of new knowledge by introducing an auxiliary decision model. Besides, we design a sample replay mechanism to fully utilize historical test samples. These historical test samples help us to estimate the target data distribution more accurately. Experimental results on multiple VLN benchmark datasets, namely R2R [2], RxR [12] and REVERIE [13], show our proposed method obtains stable improvement on several existing methods, namely RecBERT [14], HAMT [15], and DUET [16].

To sum up, our main contributions are as follows:

- We introduce a more practical test-time adaptation setting (*i.e.*, without access to the training data and complete testing data) to vision-and-language navigation task for adapting the model to new environments more robustly.
- We propose an elastic adaptation model (EAM), which has a two-branch architecture and a reply mechanism, to handle the catastrophic forgetting and unstable update problems in the above setting.
- Our method is model-agnostic and plug-and-play, outperforming existing methods in multiple VLN datasets.

II. Related Work

Vision-and-Language Navigation. Visual navigation aims at navigating to a specific position in an environment based on RGB information [17]. As natural language is a natural way to interact with agents and inform it of the navigation target position, vision-and-language navigation [2] task has drawn increasing attention in recent years. Existing work can be categorized into three main approaches to improve performance in Vision-and-Language Navigation (VLN) tasks: multimodal feature learning, designing efficient action decision strategies, and data augmentation.

Vision-and-Language Navigation requires agents to accurately comprehend instructions and the visual features of the environment, where feature learning plays a crucial role. Existing methods [18] commonly use ResNet networks pre-trained on ImageNet to extract first-person RGB-D observation features and BERT networks to extract natural language instruction features. These features from both modalities are then concatenated along the channel dimension to achieve environmental perception. Typically, these features focus on the characteristics of objects in the scene and instructions, helping the agent recognize surrounding objects and the scene structure from a first-person perspective [19]. However, this approach of separately extracting single-modality features before fusion might lead to the visual feature extractor failing to capture the most relevant features as required by the instructions [20], [21], thereby affecting navigation performance. To address this, Gao et al. [22] proposed using a cross-attention mechanism to align and fuse the two types of modality features. This fusion alignment helps the agent focus more on the part of the instruction currently being executed, enhancing the agent's overall understanding of the task [23].

With the development of pre-trained models, researchers have attempted to use multimodal models pre-trained on large-scale internet image-text data [7], [14] or first-person multimodal data [7] to extract visual and instruction features, achieving good results. For instance, Hao et al. [24] enhanced the joint representation ability of the feature extractor for image-text inputs by using two pre-training tasks, masked text prediction and action prediction, on indoor scene data. To further expand the pre-training data, Hong et al. [14] and Guhur et al. [7] proposed using image-text pairs from the web and room descriptions from the AirBnB website for multimodal representation model pre-training, enabling the agent to perform excellently with only a few training samples. Additionally, some researchers have tried to improve the agent's multimodal feature learning ability by modifying the model structure.

Action decision strategies typically involve a deep neural network aimed at receiving encoded multimodal information and predicting a series of low-level navigation actions to drive the agent to complete the navigation task. Early work [2] used Recurrent Neural Networks (RNNs) to encode all multimodal information from the navigation history process, outputting a low-level navigation action at each time step and training using reinforcement learning. The reward function mainly depends on the distance between the agent and the goal, whether the goal is reached, and the number of navigation steps. Due to the sparse reward signals from the environment, it is challenging for the agent to optimize its decision strategy [25]. Therefore, researchers designed dense intrinsic rewards to provide clear learning signals for the agent. For example, Wang et al. [25] proposed using the match degree between instructions and navigated trajectories as intrinsic rewards; Jain et al. [26], Ilharco et al. [27], and Landi et al. [28] suggested that evaluation metrics can also serve as valuable reward signals. Besides reinforcement learning, Zhang et al. [29] found that alternating between imitation learning and reinforcement learning can effectively improve the agent's navigation performance. However, solely imitating real navigation trajectories can make it difficult for the agent to adapt to imperfect trajectories during testing [30]. To address this, Krantz et al. [30] proposed the data aggregation strategy, alternating between using real navigation trajectories and the current model-predicted navigation trajectories to train the agent. Instead of directly predicting actions, Hong et al. [14] proposed converting the navigation action prediction problem into an instruction-path matching problem, pre-collecting multiple paths from the environment and selecting the most matching one. Although this method increases the success rate, it also introduces the extra cost of pre-exploring the environment. Additionally, to enhance the generalizability of decision strategies in new environments, researchers [31] proposed optimizing the policy network in a self-supervised manner in the test environment. The commonly used VLN dataset R2R [2] only contains 21,567 human-annotated navigation instructions, making data scarcity a challenge for cross-modal matching and limiting VLN performance. To address this, researchers have enhanced training data by automatically generating navigation instructions [4], navigation paths [32], and environments [5], [33]. Specifically, Fried et al. [4] proposed a "speaker" model to describe any randomly sampled navigation path, extending the navigation instructions; Fu et al. [32] adopted an adversarial path sampling strategy to automatically sample navigation paths that are more challenging for the current agent, enhancing the agent's navigation ability in complex environments; Tan et al. [5] randomly dropped some information at the environment feature level and the environment object level to generate new navigation environments. These data-driven approaches overcame the limitation of sparse training data by automatically generating data, improving the model's generalization ability and performance.

Test-Time Adaptation. TTA [8] method aims to solve the problem of distribution shift through leveraging unlabeled test samples to adjust the model parameters so as to improve the performance at test time. According to signals and parameters for adaptation, we summarize existing methods broadly as follows. For the adaptation signals, existing methods include entropy minimization [8], pseudo-label generation [34], [35], and consistency maximization [10]. For adaptation parameters, existing methods include batch norm statistics adaptation [36], classifier adjustment [37], all parameter adjustment [11]. Recently, TTA has been widely used in various domains, such as visual question answering [38], image classification [39]–[41], semantic segmentation [42]–[44], object detection [45]–[48], person re-identification [49]. In VLN task, the problem of distribution shift also exists [3]. How to apply TTA to VLN is an open question. The existing method [1] makes a preliminary attempt, but its setting does not satisfy the real application of VLN. In this paper, we consider more practical setting.

Slow vs. Fast Learning. Slow-fast learning has emerged as a significant concept in machine learning, addressing the need to balance learning from data at different temporal scales or different domains. For data at different temporal scales, several studies have explored slow-fast learning in temporal modeling [50], [51] and reinforcement learning [52], where the slow component captures long-term dependencies and trends, while the fast component adapts to more immediate changes and fine-grained details. This approach has shown improvements in tasks such as time-series forecasting and decision-making efficiency. While for data at different domains, incremental learning has obtained a considerable development, which aims to enable a model acquire new knowledge from new data while preserve old knowledge [53]. However, it faces a dilemma between slow forgetting of old knowledge and fast adaptation to new knowledge. The slow forgetting will cause an underfitting on new data, while the fast adaptation will lead to a catastrophic forgetting. To solve the problem above, slow vs. fast learning is proposed and attempts to keep a trade-off between the slow forgetting and fast adaptation [54], [55]. Motivated by it, we design a two-branch structure that contains an original model and auxiliary model. We freeze the parameters of the original model to preserve old knowledge and adapt the parameters of the auxiliary model to learn new knowledge. Hence, our model learns to balance the preservation of old knowledge and the adaptation of new knowledge.

III. Approach

A. Problem Formulation

Given an existing vision-and-language navigation model denoted as $f(\mathbf{x};\theta)$, which has been trained using the labeled training data $(\mathcal{X}^S, \mathcal{Y}^S)$, our objective is to fine-tune this model to adapt to the unlabeled testing data $\mathcal{X}^T$. Traditional methods propose a self-supervised module that provides supervised objectives in the absence of ground-truth labels $\mathcal{Y}^T$ during the testing stage. However, these methods require retraining the model parameters θ using the training data and update the model using all the testing data prior to the final inference.

In contrast, our approach tackles scenarios where we lack access to the training data, and the testing data is sequentially provided. Specifically, at each time step t, the model takes the current test sample $\mathbf{x}_t$ as input, updating the parameters

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2025.3535356

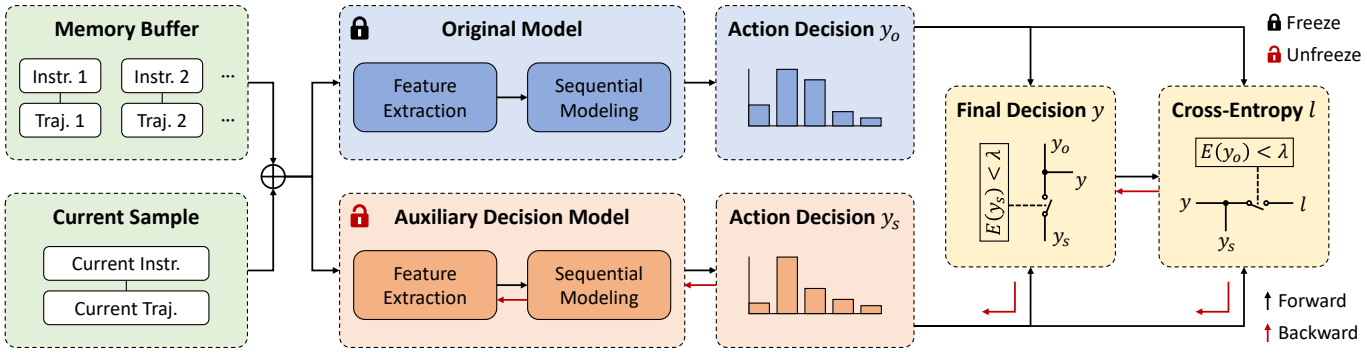

Fig. 2: General scheme of the proposed elastic adaptation model (EAM) for VLN task. We sample historical test samples from memory buffer and combine them with current test sample to form a mini-batch. Besides, we construct an auxiliary decision model to cooperate with the original model. The final decision is determined by both. We generate corresponding pseudo-label and calculate cross-entropy loss to update the auxiliary decision model.

θ to adapt the model to the current scene, and then predicts navigation action using updated parameters.

Nonetheless, adapting the model in our setting poses several challenges. Directly adapting the model to new scenes can lead to catastrophic forgetting, where previous knowledge is completely erased. Besides, using only a single test sample for updating the model may result in unstable updates. Therefore, we propose an auxiliary decision model (Section III-D) and a sample replay mechanism (Section III-E) to overcome these challenges.

B. Overview of Elastic Adaptation Model

Our test-time adaptation approach comprises two main components, namely an **auxiliary decision model** and a **sample replay mechanism**, which can be incorporated into most of the existing VLN models. The goal of our approach is to enhance the performance of the original model by incorporating knowledge from test samples. To achieve this, we introduce the auxiliary decision model, which is combined with the original model to form a two-branch structure. The original model is frozen so that the learned knowledge in the training phase will not be forgotten. The auxiliary decision model helps the original model to adapt to new scenes. These two models mutually support each other to make the final action decision together. To make the adaptation process more stable, we propose a sample replay mechanism that stores the historical test samples in a memory buffer. When adapting the model to a new scene, the historical test samples and current sample build a mini-batch, providing richer information for updating the model in the testing phase.

The overall framework of our proposed method, which we refer to as EAM, is illustrated in Figure 2. In the following subsections, we will first revisit the existing methods for vision-and-language navigation, followed by the introduction of our EAM that adapts the existing methods to new scenes in the testing phase.

C. A Revisit of Existing VLN Methods

We first introduce the basic process of existing methods for vision-and-language navigation, which mainly involves feature extraction, sequential modeling, and action decision. During the navigation process, the agent receives navigation instructions and visual observations from the environment. A text encoder and a visual encoder are used to extract features from these two inputs, respectively. Then a sequential model (*e.g.*, transformer [14], [24] or LSTM [2]) is used to combine these features and the historical observation to figure out an agent stage features. Finally, an action decision model is utilized to select an executable action from a set of candidate actions. The action decision is formulated as a classification problem, taking the agent state features as input and calculating the possibility distribution over all candidate actions.

Existing methods leverage a combination of imitation learning (IL) and reinforcement learning (RL) techniques to train the model [14], [56]. In the case of imitation learning, the model is supervised using ground-truth action labels. On the other hand, reinforcement learning provides feedback to the model in the form of corresponding rewards.

D. Auxiliary Decision Model

Due to the scene distribution shift, the trained models introduced in the last section struggle to perform navigation tasks in the new scenes. A possible solution is to adjust the model parameter according to the testing sample in an unsupervised manner [8]. Since the ground-truth labels of the testing samples are not available, some test-time adaptation methods use the trained model to predict pseudo-labels and use these pseudo-labels to adjust the models to the new scenes. However, if we adapt the original model directly, it will inevitably forget the old knowledge learned from the training stage, resulting in a degradation of the generated pseudo-label quality. The noisy pseudo-labels further lead to error accumulation and catastrophic forgetting [11], which exacerbates the degradation of the pseudo-label. It forms a vicious circle and affects the model adaptation, causing a decline in the performance. When the model starts to deteriorate, it may not recover again [57]. To avoid forgetting, an intuitive method is to reduce the learning rate to slow down the update of the model. Though it relieves the forgetting of old knowledge, it

also restricts the model to learn new knowledge. We need to keep a balance between avoiding the forgetting of old knowledge and promoting the learning of new knowledge.

In response to the above problem, we propose to design an auxiliary decision model, which has the same network architecture as the original model. We combine the auxiliary decision model with the original model to form a two-branch structure. The final decision is determined by the two model jointly. It avoids the forgetting of old knowledge by keeping the original model frozen, and promotes the learning of new knowledge by updating the auxiliary decision model. We initialize the above two model through the trained parameters at the beginning of inference.

Action Decision. In the testing stage, the original model and auxiliary decision model use the same input for forward propagation to obtain the corresponding action decision, respectively. The final action decision of the two-branch structure is defined as the sum of the two above. The process can be formulated as:

$$y_o = f_o(\mathbf{x}; \theta_o), \quad y_s = f_s(\mathbf{x}; \theta_s), \tag{1}$$

$$y = y_o + y_s, \tag{2}$$

where, y_o and θ_o are the action decision and model parameters for original model; y_s and θ_s are the corresponding elements for auxiliary decision model; $\mathbf{x}$ is the model input, including text instructions and visual images.

Cross-Entropy Minimization. We consider to freeze the original model and only update the auxiliary decision model during back propagation. To update the model parameters, we generate pseudo-label according to the final action decision and optimize the cross-entropy loss through gradient descent algorithm. The process can be formulated as:

$$l = \mathrm{CrossEntropy}(y, y_s), \tag{3}$$

$$\theta_s = \theta_s + \alpha \nabla_{\theta_s} l, \tag{4}$$

where, α is the learning rate. The original model and auxiliary decision model complement each other. The former remains unchanged to relieve the forgetting problem. The latter adjusts dynamically to adapt to the new environment. It achieves a balance between avoiding the forgetting of old knowledge and promoting the learning of new knowledge.

Sample Selection. The samples used to adjust the model parameters should be reliable. If the predicted action of a sample (*i.e.*, a probability distribution over all possible actions) has very high entropy, it indicates that the model is uncertain about this particular sample. This uncertainty often arises from samples with limited information, like an RGB image of a plain white wall or an unclear instruction. Using these samples for updating the model in the test phrase may hurt the performance due to their biased and unreliable gradients [9]. Consequently, we set a confidence threshold to select the test samples for updating the auxiliary decision model. Specifically, when the entropy of the original model's action decision is less than the threshold, the corresponding sample is used to update the model. Otherwise, we do not calculate the loss to avoid the disturbance of noise samples. Equation (3) can be rewritten as:

$$l = \mathbb{I}(\mathrm{Entropy}(y_o) < \lambda) \cdot \mathrm{CrossEntropy}(y, y_s), \tag{5}$$

$$\lambda = a \times \ln C, \tag{6}$$

where, λ is the confidence threshold; a is the confidence coefficient; C is the number of possible actions. In addition, to maintain the stability of the final action decision, the action decision of the auxiliary decision model should also have a high confidence before it is combined with the action decision of the original model. Equation (2) can be rewritten as:

$$y = y_o + \mathbb{I}(\mathrm{Entropy}(y_s) < \lambda) \cdot y_s. \tag{7}$$

Algorithm 1 The update of memory buffer.

Require: Current sample $\mathbf{x}$, current index n, memory buffer $\mathcal{M}$, memory capacity M.

$m = |\mathcal{M}|$. *// Calculate the number of samples*
if $m < M$ **then**
  $\mathcal{M}.\mathrm{append}(\mathbf{x})$. *// Append the current sample*
else
  $i = \mathrm{randint}(0, n)$.
  if $i < M$ **then**
    $\mathcal{M}[i] = \mathbf{x}$. *// Replace the i^{th} sample*
  end if
end if

Ensure: Updated memory buffer $\mathcal{M}$.

Algorithm 2 The replay of history sample.

Require: Current sample $\mathbf{x}$, memory buffer $\mathcal{M}$, batch size K.

$m = |\mathcal{M}|$. *// Calculate the number of samples*
if $m < K$ **then**
  $B = \mathbf{x}$. *// Return the current sample*
else
  $B_h \sim \mathcal{M}$. *// Select $K-1$ history samples*
  $B = B_h \cup \mathbf{x}$. *// Form the mini-batch*
end if

Ensure: Mini-batch B.

E. Sample Replay Mechanism

In our setting, the test samples are in the form of data streams. The model can only touch a single test sample each time. It is hard for a single test sample to represent the target data distribution accurately. If we only use it to adjust the model parameters, it will optimize the model to different local optima each time, which leads to an unstable update. To solve the above problem, a possible way is to reuse the historical test samples. However, existing methods generally do not have a retrieval operation for the test samples, *i.e.*, the historical test samples can not be obtained again.

To this end, we propose a sample replay mechanism to fully utilize historical test samples. We design a memory buffer to store test samples each time. We random select the historical test samples from the memory buffer, and combine them with the current test sample to form a mini-batch. We adapt the

Algorithm 3 Elastic Adaptation Model (EAM) Workflow

Require: Current sample x, current index n, memory buffer $\mathcal{M}$, memory capacity M, batch size K

Memory Buffer Update:
$m = |\mathcal{M}|$ *// Calculate the number of samples in the memory buffer*
if $m < M$ **then**
  $\mathcal{M}$.append(x) *// Append the current sample to the memory buffer*
else
  $i = \text{randint}(0, n)$
  if $i < M$ **then**
    $\mathcal{M}[i] = x$ *// Replace the ith sample in the memory buffer*
  end if
end if
Replay of Historical Samples:
$m = |\mathcal{M}|$ *// Recalculate the number of samples in the memory buffer*
if $m < K$ **then**
  $B = x$ *// Return the current sample as the mini-batch*
else
  $B_h \sim \mathcal{M}$ *// Select $K-1$ historical samples from the memory buffer*
  $B = B_h \cup x$ *// Form the mini-batch with the current sample*
end if
Model Inference:
Use the mini-batch B for inference with the original model and the auxiliary decision model
Obtain action decisions y_o and y_s from the original and auxiliary models respectively
Final Decision:
Combine y_o and y_s to make the final decision y
Model Update:
Generate corresponding pseudo-label
Calculate cross-entropy loss to update the auxiliary decision model

Ensure: Final decision y and updated memory buffer $\mathcal{M}$

model through the mini-batch to avoid the problem of unstable update caused by a single test sample.

Note that the memory buffer will store the observations and action decisions at each time step during the navigation process. The model only interacts with the environment corresponding to the current test sample. With regard to the historical test samples, the model directly loads the stored observations for forward propagation and executes the stored actions without interacting with the environments again. Therefore, it still satisfies our setting mentioned in section I.

Memory Buffer Update. To avoid the unbearable storage costs, we set the memory buffer as a data queue with a fixed capacity of M. We update the memory buffer through reservoir sampling [58] so that each stored sample has the same probability to be retained or replaced. Specifically, for the n^{th} sample, if the number of the samples in the memory buffer is less than the memory capacity M, the current sample is appended to the memory buffer directly. Otherwise, we random sample a value i from 0 to n. If the value i is less than the memory capacity M, we replace the i^{th} sample in the memory buffer with the current sample. The pseudo-code is shown in Algorithm 1.

TABLE I: The number of instructions and trajectories in VLN benchmark datasets.

	Val-Seen		Val-Unseen	
	Instr.	Traj.	Instr.	Traj.
R2R [2]	1,020	340	2,349	783
RxR [12]	8,813	1,244	13,652	1,517
REVERIE [13]	1,423	515	3,521	1,328

History Sample Replay. To fully utilize the historical test samples, we random select stored samples from the memory buffer, and combine them with the current sample to form a mini-batch. We adjust the model parameters through the mini-batch to avoid the problem of unstable update by a single test sample. Note that we can not select enough historical test samples to form a mini-batch when the number of the samples in the memory buffer is less than the batch size. At this moment, we choose to inference the current test sample directly and do not use it to adjust the model parameters. The pseudo-code is shown in Algorithm 2.

F. Elastic Model Adaptation and Inference

We summarize the whole inference process of our proposed method. In the testing stage, the current test sample is obtained in the form of data streams. And the historical test samples are sampled from the memory buffer through the sample replay mechanism. We form a mini-batch with the current and historical test samples. The original model and auxiliary model use the mini-batch for forward propagation and obtain corresponding action decisions. We combine them to get the final action decision and calculate the cross-entropy loss for back propagation to update the auxiliary model. Then we update the current test sample to the memory buffer. The above process is repeated until all test samples are inferred.

It is worth noting that when the length of mini-batch is equal to 1, it indicates that the total number of test samples in the memory buffer is less than the batch size. We can not obtain sufficient test samples from the memory buffer to from a mini-batch. At this time, we only perform forward propagation and do not use that it to adjust the model parameters.

IV. Experiments

A. Datasets

We conduct our experiments on multiple VLN benchmark datasets, *i.e.*, R2R [2], RxR [12] and REVERIE [13]. Note that we do not need to access the training data in these datasets. Table I provides the statistical information about these datasets.

R2R. It contains 1,123 trajectories from 64 indoor scenes for validation. Each trajectory is associated with three different

TABLE II: Comparison between our method and existing methods on R2R dataset. For RecBERT, HAMT, DUET, we report the reproduced results using official checkpoints. The relative improvement brought by the incorporation of the existing test-time adaption method DAVIS and our EAM are shown in parentheses.

	Val-Seen				Val-Unseen			
	TL ↓	NE ↓	SR ↑	SPL ↑	TL ↓	NE ↓	SR ↑	SPL ↑
Seq2Seq [2]	11.33	6.01	39	-	**8.39**	7.81	22	-
SF [4]	-	3.36	66	-	-	6.62	35	-
SMNA [56]	-	3.22	67	58	-	5.52	45	32
RCM [59]	10.65	3.53	67	-	11.46	6.09	43	-
PRESS [60]	10.57	4.39	58	55	10.36	5.28	49	45
EnvDrop [5]	11.00	3.99	62	59	10.70	5.22	52	48
AuxRN [61]	-	3.33	70	67	-	5.28	55	50
PREVALENT [24]	10.32	3.67	69	65	10.19	4.71	58	53
RelGraph [62]	**10.13**	3.47	67	65	9.99	4.73	57	53
AirBERT [7]	11.09	2.68	75	70	11.78	4.01	62	56
HOP [63]	11.26	2.72	75	70	12.27	3.80	64	57
RecBERT [14]	11.13	2.90	72.18	67.72	12.01	3.93	62.75	56.84
+ DAVIS* [1]	10.84 (+2.6%)	2.83 (+2.4%)	71.11 (-1.5%)	66.74 (-1.4%)	11.61 (+3.3%)	3.91 (+0.5%)	63.26 (+0.8%)	57.34 (+0.9%)
+ EAM (Ours)	10.71 (+3.8%)	2.73 (+5.8%)	73.46 (+1.7%)	69.34 (+2.4%)	11.47 (+4.5%)	3.93 (+0.0%)	64.20 (+2.3%)	58.51 (+3.1%)
HAMT [15]	11.15	2.51	75.61	72.18	11.46	3.62	66.24	61.51
+ DAVIS* [1]	11.36 (-1.9%)	2.99 (-19.1%)	70.23 (-7.1%)	66.83 (-7.4%)	11.56 (-0.9%)	3.79 (-4.7%)	64.92 (-2.0%)	59.66 (-3.0%)
+ EAM (Ours)	11.07 (+0.7%)	2.41 (+4.0%)	76.98 (+1.8%)	**73.64 (+2.0%)**	11.33 (+1.1%)	3.49 (+3.6%)	68.20 (+3.0%)	**63.37 (+3.0%)**
DUET [16]	12.33	2.29	78.66	72.74	13.94	3.31	71.52	60.41
+ DAVIS* [1]	11.53 (+6.4%)	2.39 (-4.3%)	77.57 (-1.3%)	72.50 (-0.3%)	12.00 (+14.0%)	3.77 (-13.9%)	65.99 (-7.7%)	57.67 (-4.5%)
+ EAM (Ours)	12.05 (+2.3%)	**2.26 (+1.3%)**	**78.86 (+0.3%)**	73.39 (+0.8%)	13.28 (+4.7%)	**3.23 (+2.4%)**	**72.33 (+1.1%)**	61.35 (+1.4%)

instructions. All of samples are split into validation seen and validation unseen sets with 56 and 18 scenes, respectively.

RxR. Its data scale is much larger than R2R and it solves the problem of path biases existed in R2R. Besides, the instructions in RxR involve different kinds of languages, including English, Hindi and Telugu.

REVERIE. The format of instructions in REVERIE is quite different from the above. Instead of giving detailed instructions, it only provides high-level instructions that describe the target position and object without step-by-step guidance.

B. Evaluation Metrics

We follow existing methods [14], [15] to evaluate the navigation performance using trajectory length (TL), navigation error (NE), success rate (SR) and success rate weighted by path length (SPL). For fair comparison, we further employ other evaluation metrics, such as normalized dynamic time warping (nDTW) and success weighted by nDTW (sDTW) for RxR, remote grounding success rate (RGS) and RGS weighted by path length (RGSPL) for REVERIE. The details of each metric are described as follows:

TL & NE. TL measures the agent's final trajectory length in meters. NE measures the geodesic distance from agent's final position to goal position in meters.

SR & SPL. SR measures the ratio of the agent executing STOP action within 3 meters from the goal position. SPL is SR multiplied by the ratio between the length of the shortest path and the predicted path.

nDTW & sDTW. nDTW evaluates how well the predicted path matches the ground-truth path. sDTW is based on nDTW and only calculates the successful trajectories.

RGS & RGSPL. RGS is the ratio of the agent grounding the target object successfully. And RGSPL is RGS weighted by path length, which is similar with SPL. These two metrics are used in the REVERIE dataset.

C. Implementation Details

We implement our method based on PyTorch framework and Matterport3D simulator [64]. We focus on the discrete environment where the agent navigates between pre-defined viewpoints. Our proposed method EAM is based on multiply existing mothods including RecBERT [14], HAMT [15] and DUET [16]. The navigation model varies depending on the baseline we use. We directly load the trained model parameters, avoiding any interference with the original training process. We only retrain the model when the checkpoints are not provided. We adapt our method on a single NVIDIA Titan XP GPU. We use an Adam optimizer with a learning rate of 1e-5. The confidence coefficient a is set to 0.4. The memory capacity M is set to 32. The batch size K is set to 8.

D. Benchmark Methods

Our method can be applied to various existing methods in the testing stage without altering the training process. To verify the effectiveness of our method, we choose the following existing methods as the benchmark.

RecBERT. This method [14] is based on the V&L BERT model [65]. It uses the [CLS] token in the transformer as a internal recurrent unit to encode histories, so that it does not need to apply any external recurrent modules.

TABLE III: Comparison between our method and existing methods on RxR dataset. The relative improvement brought by the incorporation of our EAM is shown in parentheses.

	Val-Seen				Val-Unseen			
	SR ↑	SPL ↑	nDTW ↑	sDTW ↑	SR ↑	SPL ↑	nDTW ↑	sDTW ↑
Mono [12]	28.8	-	46.8	23.8	28.5	-	44.5	23.1
EnvDrop [5]	48.1	44.0	57.0	40.0	38.5	34.0	51.0	32.0
Syntax [66]	48.1	44.0	58.0	40.0	39.2	35.0	52.0	32.0
HOP [63]	49.4	45.0	58.0	40.0	42.3	36.0	56.6	33.0
ADAPT [67]	52.7	47.0	61.3	**58.5**	46.7	40.3	53.5	37.3
HAMT [15]	59.37	55.65	65.41	50.84	56.50	52.72	63.33	48.37
+ EAM (Ours)	**60.14**$_{(+1.3\%)}$	**56.54**$_{(+1.6\%)}$	**65.98**$_{(+0.9\%)}$	51.46$_{(+1.2\%)}$	**57.45**$_{(+1.7\%)}$	**53.74**$_{(+1.9\%)}$	**64.19**$_{(+1.4\%)}$	**49.08**$_{(+1.5\%)}$

TABLE IV: Comparison between our method and existing methods on REVERIE dataset. The relative improvement brought by the incorporation of our EAM is shown in parentheses.

	Val-Seen				Val-Unseen			
	Navigation		Grounding		Navigation		Grounding	
	SR ↑	SPL ↑	RGS ↑	RGSPL ↑	SR ↑	SPL ↑	RGS ↑	RGSPL ↑
Seq2Seq [2]	29.59	24.01	18.97	14.96	4.20	2.84	2.16	1.63
SMNA [56]	41.25	39.61	30.07	28.98	8.15	6.44	4.54	3.61
RCM [59]	23.33	21.82	16.23	15.36	9.29	6.97	4.89	3.89
FAST-MATTN [13]	50.53	45.50	31.97	29.66	14.40	7.19	7.84	4.67
ORIST [68]	45.19	42.21	29.87	27.77	16.84	15.14	8.52	7.58
SIA [69]	61.91	57.08	45.96	42.65	31.53	16.28	22.41	11.56
AirBERT [7]	47.01	42.34	32.75	30.01	27.89	21.88	18.23	14.18
RecBERT [14]	51.79	47.96	38.23	35.61	30.67	24.90	18.77	15.27
HOP [63]	53.76	47.19	38.65	33.85	31.78	26.11	18.85	15.73
HAMT [15]	44.85	41.67	28.16	26.15	32.95	30.20	18.92	17.28
+ EAM (Ours)	46.10$_{(+2.8\%)}$	43.40$_{(+4.2\%)}$	28.82$_{(+2.3\%)}$	27.13$_{(+3.7\%)}$	34.59$_{(+5.0\%)}$	31.98$_{(+6.0\%)}$	19.91$_{(+5.2\%)}$	18.30$_{(+6.0\%)}$
DUET [16]	71.75	63.94	57.41	51.14	46.98	33.73	32.15	23.03
+ EAM (Ours)	**72.59**$_{(+1.2\%)}$	**64.96**$_{(+1.6\%)}$	**57.83**$_{(+0.7\%)}$	**51.80**$_{(+1.3\%)}$	**48.00**$_{(+2.2\%)}$	**35.65**$_{(+6.0\%)}$	**32.72**$_{(+1.8\%)}$	**24.22**$_{(+5.2\%)}$

HAMT. This method [15] is also based on the transformer model. It explicitly encodes the historical information as a sequence of previous observations and proposes a hierarchical module to reduce the computational complexity.

DUET. This method [16] constructs a topological map to extend the action space for efficient exploration. It proposes a dual-scale graph transformer to encode fine-scale information of local observations and coarse-scale information on a global map simultaneously.

E. Performance Comparison

Test-time adaptation performance on R2R dataset. We compare our method with existing methods on both validation seen and unseen sets of R2R dataset. The results are shown in Table II. For HAMT [15], our method gets better performance in terms of multiple metrics, *e.g.*, NE, SR and SPL. In detail, we achieve 1.37% and 1.96% SR absolute improvement and 1.46% and 1.86% SPL absolute improvement on validation seen and unseen sets, respectively, which demonstrates the effectiveness of our method. Besides, our method also achieves stable improvement based on RecBERT [14]. Specifically, we increase SR from 62.75% to 64.20% and SPL from 56.84% to 58.51% on validation unseen set. When we apply our method to the state-of-the-art method DUET [16], we still acquire a considerable performance improvement on SR, increasing from 71.52% to 72.33%. The experimental results on multiply benchmark show that our method has strong general applicability. It can be applied to most existing methods flexibly without relying on specific models or algorithms.

For validation seen set, the scenes in this set is the same as the training. Our model achieves comparable or even better results than benchmark methods, which shows that our method avoids forgetting old knowledge. For validation unseen set, the agent has never seen the scenes in this set during training. Our method still outperforms all benchmark methods in term of multiple metrics, which shows that our method adapts to the new environments effectively.

We also compare our method with the existing test-time adaptation method DAVIS [1]. For a fair comparison, we reimplement DAVIS on our testing setting, *i.e.*, the training data is not accessed in the testing phase and the testing sample comes one by one. We name the reimplemented version as DAVIS*. Under all three benchmark methods, DAVIS brings limited and even negative improvement. We speculate that the design of DAVIS is tailored for offline settings. In our setting, when the model cannot access all test samples in advance, its

TABLE V: More experiment results on navigation benchmarks

Method	R4R		R2R-Last		CVDN		R2R-Back	
	SR	SPL	SR	SPL	SR	SPL	SR	SPL
HAMT	44.15	40.95	45.34	40.89	16.98	**11.05**	52.23	49.30
+ EAM(ours)	**44.73**	**41.99**	**46.32**	**41.73**	**17.97**	10.88	**54.11**	**51.10**

TABLE VI: Ablation studies on each component in our proposed method EAM including the auxiliary decision model and sample replay mechanism. Results are evaluated on validation unseen set of R2R dataset.

	Val-Unseen			
	TL ↓	NE ↓	SR ↑	SPL ↑
HAMT [15]	11.46	3.62	66.24	61.51
+ EAM w/o ADM	11.37	4.73	54.79	48.23
+ EAM w/o SRM	11.53	3.66	66.96	62.08
+ EAM (Ours)	**11.33**	**3.49**	**68.20**	**63.37**

updates become unstable and are susceptible to the problem of catastrophic forgetting, leading to performance degradation.

Test-time adaptation performance on RxR and REVERIE datasets. We also conduct experiments on datasets RxR and REVERIE. The experimental results are shown in Table III and Table IV. We apply our method to benchmark method HAMT to adjust the model parameters. For dataset RxR, we increase SR from 56.50% to 57.45% and SPL from 52.72% to 53.74% on the validation unseen set. For dataset REVERIE, we achieve 1.64% SR improvement and 1.78% SPL improvement on the validation unseen set, respectively. Furthermore, we outperform the state-of-the-art method DUET by 0.84% and 1.02% in terms of SR on validation seen and unseen set of REVERIE dataset, respectively. The experimental results show that our proposed method achieves stable performance improvement over benchmark methods on different datasets, which further demonstrates the general applicability of our method.

Test-time adaptation performance on more navigation benchmarks. As shown in Tab. V, we tested more navigation benchmarks, and the experimental results show that our method has significant improvements over the baseline on most metrics. R4R extends R2R by concatenating two adjacent tail-to-head trajectories, and R2R-Last, similar to REVERIE, uses only the last sentence of the original R2R instructions to describe the final destination, both emphasizing long horizon navigation; CVDN defines a task where the agent navigates based on multi-turn question-answering dialogs, requiring understanding of human conversations with often ambiguous and under-specified instructions; and R2R-Back introduces a new VLN setup where the agent must return to its start location, requiring memory of navigation histories and handling of vague commands(e.g., "Go to the nightstand in the bedroom and return to the start point."). Experimental results demonstrate that our method can effectively handle various long-distance or instruction-ambiguous navigation tasks, further highlighting the effectiveness of our approach.

F. Ablation Studies

In this section, we evaluate the effectiveness of each component within our method and explore the influence of hyper-parameters on the performance. We conduct experiments on R2R and use HAMT as benchmark method.

Effectiveness of Auxiliary Decision Model. We propose an auxiliary decision model and combine it with the original model. At test time, we only update the auxiliary decision model and freeze the original model. To verify its effectiveness, we design a variant that removes the auxiliary decision model and updates the original model directly. We name the variant as EAM w/o ADM. In Table VI, the performance of the variant is severely degraded, even much lower than the benchmark method, decreasing SR from 66.24% to 54.79% and SPL from 61.51% to 48.23% significantly. It indicates the importance of the auxiliary decision model. When directly updating the original model, it is hard to preserve old knowledge, leading to the problem of catastrophic forgetting and noisy pseudo-labels. The model is difficult to adapt to the new environments through the noisy pseudo-labels, resulting in the deterioration of the model performance.

Effectiveness of Sample Replay Mechanism. We propose a sample replay mechanism to fully utilize the historical test samples to adjust the model parameters. To verify its effectiveness, we design a variant that removes the sample replay mechanism and only uses the current test sample to adjust the model parameters. We name the variant as EAM w/o SRM. In Table VI, compared with the benchmark method, the variant only gets slightly improvement. While our method outperforms the variant, increasing SR from 66.96% to 68.20% and SPL from 62.08% to 63.37%. These experimental results show the significance of the sample replay mechanism. It avoids the bias of a single test sample and represents the target data distribution more accurately through historical test samples stored in the memory buffer.

Influence of Hyper-parameters. We conduct experiments on different values of hyper-parameters to evaluate their effect on the model performance. We focus on confidence coefficient a, memory capacity M and batch size K. For the confidence coefficient, we select its value from $\{0.2, 0.3, 0.4, 0.5\}$ as shown in Figure 3 (a). This coefficient determines whether the test samples are used to adjust the model parameters. If the value is too small, most test samples will be screened out and the model is hard to learn enough new knowledge from the remaining limited test samples to adapt to the new environment. If the value is too large, it will introduce some high entropy samples which may hurt the performance. We choose the value with the best performance ($a = 0.4$) as the default setting. For the memory capacity, each unit of the memory buffer only needs to store annotation information of a certain episode, such as the scene name, episode ID, and the path executed in that episode (represented by unique waypoint IDs). One storage unit occupies approximately 0.36KB of memory and does not require storing observation information for each step within the episode. If our method is to be used in real-world scenarios, such as on a mobile robot, each memory unit will need to additionally store the observation

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2025.3535356

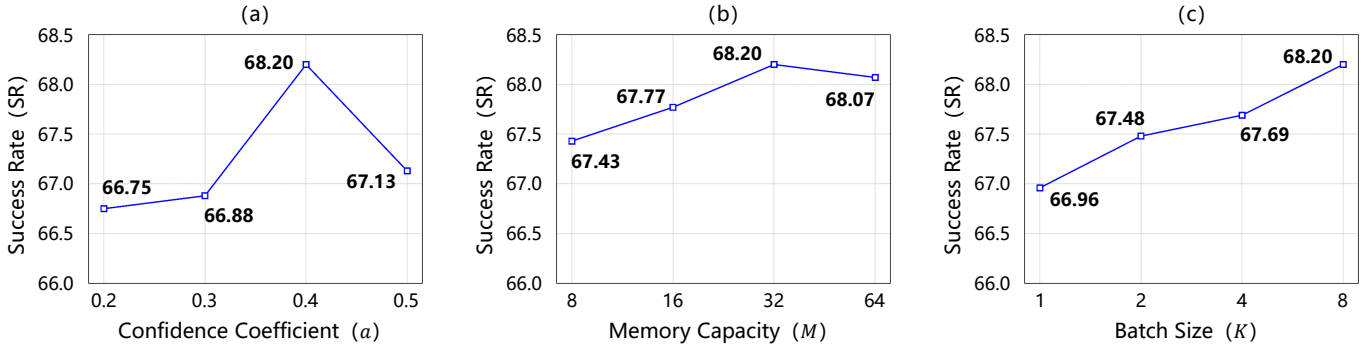

Fig. 3: Ablation studies on hyper-parameters including confidence coefficient a, memory capacity M and batch size K. Results are evaluated on validation unseen set of R2R dataset. We show the curve of success rate (SR) metric.

Fig. 4: Visualization of navigation examples using our method and benchmark method, respectively. The visualization results are from validation unseen set of R2R datasets.

images at each step. Assuming a panoramic image size of (1080, 256, 3) and that each episode lasts up to 15 steps, one memory unit will require an additional 11.86 MB of storage. This overhead is entirely acceptable for a mobile robot. We select its value from $\{8, 16, 32, 64\}$. As shown in Figure 3 (b), the performance gradually improves with the memory capacity increasing. To comprehensively consider the model performance and storage cost, we choose $M = 32$ as the default setting. For the batch size, we select its value from $\{1, 2, 4, 8\}$ as shown in Figure 3 (c). The batch size reflects the number of samples that the model touches during each inference. We sample more historical test samples from the memory buffer to represent the target data distribution as the batch size increasing. When the batch size is set to 1, the effect is equal to removing the sample replay mechanism. Due to the limitation of GPU memory, we do not further increase the batch size to explore its influence and choose $K = 8$ as the default setting.

G. Ablation Study of the Size of Testing Batch.

In our experiments, we assume that only one mobile robot exists in the new scene. For each time step, this robot captures the most recent observation x, along with 7 historical trajectories (*i.e.,* batch memory buffers) to build a batch for test-time adaptation. As for the situation where multiple robots are in the new scene, these robots capture the most recent observation x individually (number of $x > 1$). We can use all these observations, together with batch memory buffers, to update the model using our proposed elastic model adaptation technique. Experimental results are shown in Table VII. Our

elastic model adaptation is suitable for different numbers of x. Using different robots' current and historical observations for model adaption improves performance.

TABLE VII: Ablation study of different numbers of x.

Testing Bstch Size	Elastic Adaptation Model	TL ↓	NE ↓	SR↑	SPL↑
1	x	11.46	3.62	66.24	61.51
1	✓	11.33	**3.49**	68.20	63.37
3	✓	**11.33**	3.61	67.97	63.43
7	✓	11.47	3.58	**68.86**	**64.39**

Experimental results are shown in Table VII below. Our elastic model adaptation is suitable for different numbers of x. Using different robots' current and historical observations for model adaption improves performance.

H. Visualization Results

We visualize navigation trajectories obtained by our method and compare them with the results of the benchmark method HAMT in Figure 4. We show instructions and the panoramic images observed by the agent at the start, intermediate, and stop viewpoints. The red arrow indicates the direction decided by the agent at each moment. The green tick on the final image represents a successful navigation. The red cross means that the navigation process is a failure. The corresponding instruction that describes the navigation process is shown in the green rounded rectangle on the top.

Given the same instruction and start viewpoint, the benchmark method fails to follow the instruction, while our method navigates to the target position successfully. As shown in Figure 4, the benchmark method fails to identify the orientation of the kitchen sink and selects the wrong direction at the beginning, causing the subsequent process to gradually deviate from the target and fail the navigation task. Our method follows the instructions, chooses to walk past the two kitchen sinks, and successfully stops in the hallway indicated by the instruction.

V. Conclusion

To improve the generalization ability, we consider a more practical setting to apply TTA method to VLN task. We propose an elastic adaptation model (EAM) to adapt the model to new environments. We design an auxiliary decision model and combine it with the original model to avoid the forgetting of old knowledge and promote the learning of new knowledge. Moreover, we design a sample replay mechanism to fully utilize the historical test samples for model parameter adjustment. Experimental results show that our method can be applied to most existing methods and achieves better performance in multiple VLN tasks.

References

[1] Y. Lu, H. Zhang, P. Nie, W. Feng, W. Xu, X. E. Wang, and W. Y. Wang, "Anticipating the unseen discrepancy for vision and language navigation," *arXiv*, 2022.

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *CVPR*, 2018, pp. 3674–3683.

[3] V. S. Dorbala, G. A. Sigurdsson, R. Piramuthu, J. Thomason, and G. S. Sukhatme, "Clip-nav: Using CLIP for zero-shot vision-and-language navigation," in *CoRL*, 2022.

[4] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," in *NeurIPS*, 2018, pp. 3318–3329.

[5] H. Tan, L. Yu, and M. Bansal, "Learning to navigate unseen environments: Back translation with environmental dropout," in *NAACL-HLT*, 2019, pp. 2610–2621.

[6] C. Liu, F. Zhu, X. Chang, X. Liang, Z. Ge, and Y. Shen, "Vision-language navigation with random environmental mixup," in *ICCV*, 2021, pp. 1624–1634.

[7] P. Guhur, M. Tapaswi, S. Chen, I. Laptev, and C. Schmid, "Airbert: In-domain pretraining for vision-and-language navigation," in *ICCV*, 2021, pp. 1614–1623.

[8] D. Wang, E. Shelhamer, S. Liu, B. A. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *ICLR*, 2021.

[9] S. Niu, J. Wu, Y. Zhang, Y. Chen, S. Zheng, P. Zhao, and M. Tan, "Efficient test-time model adaptation without forgetting," in *ICML*, 2022, pp. 16 888–16 905.

[10] M. Zhang, S. Levine, and C. Finn, "MEMO: test time robustness via adaptation and augmentation," in *NeurIPS*, 2022, pp. 38 629–38 642.

[11] Q. Wang, O. Fink, L. V. Gool, and D. Dai, "Continual test-time domain adaptation," in *CVPR*, 2022, pp. 7191–7201.

[12] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge, "Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding," in *EMNLP*, 2020, pp. 4392–4412.

[13] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen, and A. van den Hengel, "REVERIE: remote embodied visual referring expression in real indoor environments," in *CVPR*, 2020, pp. 9979–9988.

[14] Y. Hong, Q. Wu, Y. Qi, C. R. Opazo, and S. Gould, "VLN BERT: A recurrent vision-and-language BERT for navigation," in *CVPR*, 2021, pp. 1643–1653.

[15] S. Chen, P. Guhur, C. Schmid, and I. Laptev, "History aware multimodal transformer for vision-and-language navigation," in *NeurIPS*, 2021, pp. 5834–5847.

[16] S. Chen, P. Guhur, M. Tapaswi, C. Schmid, and I. Laptev, "Think global, act local: Dual-scale graph transformer for vision-and- language navigation," in *CVPR*, 2022, pp. 16 516–16 526.

[17] S. Wang, Z. Wu, X. Hu, Y. Lin, and K. Lv, "Skill-based hierarchical reinforcement learning for target visual navigation," *IEEE Transactions on Multimedia*, 2023.

[18] 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," *arXiv preprint arXiv:1911.00357*, 2019.

[19] M. Engelcke, A. R. Kosiorek, O. P. Jones, and I. Posner, "Genesis: Generative scene inference and sampling with object-centric latent representations," *arXiv preprint arXiv:1907.13052*, 2019.

[20] R. Schumann and S. Riezler, "Analyzing generalization of vision and language navigation to unseen outdoor areas," *arXiv preprint arXiv:2203.13838*, 2022.

[21] J. Thomason, D. Gordon, and Y. Bisk, "Shifting the baseline: Single modality performance on visual navigation & qa," *arXiv preprint arXiv:1811.00613*, 2018.

[22] C. Gao, J. Chen, S. Liu, L. Wang, Q. Zhang, and Q. Wu, "Room-and-object aware knowledge reasoning for remote embodied referring expression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3064–3073.

[23] F. Landi, L. Baraldi, M. Corsini, and R. Cucchiara, "Embodied vision-and-language navigation with dynamic convolutional filters," *arXiv preprint arXiv:1907.02985*, 2019.

[24] W. Hao, C. Li, X. Li, L. Carin, and J. Gao, "Towards learning a generic agent for vision-and-language navigation via pre-training," in *CVPR*, 2020, pp. 13 134–13 143.

[25] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6629–6638.

[26] V. Jain, G. Magalhaes, A. Ku, A. Vaswani, E. Ie, and J. Baldridge, "Stay on the path: Instruction fidelity in vision-and-language navigation," *arXiv preprint arXiv:1905.12255*, 2019.

[27] G. Ilharco, V. Jain, A. Ku, E. Ie, and J. Baldridge, "General evaluation for instruction conditioned navigation using dynamic time warping," *arXiv preprint arXiv:1907.05446*, 2019.

[28] F. Landi, L. Baraldi, M. Cornia, M. Corsini, and R. Cucchiara, "Multi-modal attention networks for low-level vision-and-language navigation," *Computer vision and image understanding*, vol. 210, p. 103255, 2021.

[29] W. Zhang, C. Ma, Q. Wu, and X. Yang, "Language-guided navigation via cross-modal grounding and alternate adversarial learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 9, pp. 3469–3481, 2020.

[30] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee, "Beyond the nav-graph: Vision-and-language navigation in continuous environments," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer, 2020, pp. 104–120.

[31] Y. Lu, H. Zhang, P. Nie, W. Feng, W. Xu, X. E. Wang, and W. Y. Wang, "Anticipating the unseen discrepancy for vision and language navigation," *arXiv preprint arXiv:2209.04725*, 2022.

[32] T.-J. Fu, X. E. Wang, M. F. Peterson, S. T. Grafton, M. P. Eckstein, and W. Y. Wang, "Counterfactual vision-and-language navigation via adversarial path sampler," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. Springer, 2020, pp. 71–86.

[33] D. An, Y. Qi, Y. Huang, Q. Wu, L. Wang, and T. Tan, "Neighbor-view enhanced model for vision and language navigation," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5101–5109.

[34] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, "Contrastive test-time adaptation," in *CVPR*, 2022, pp. 295–305.

[35] P. Wang, Y. Yang, Y. Xia, K. Wang, X. Zhang, and S. Wang, "Information maximizing adaptation network with label distribution priors for unsupervised domain adaptation," *IEEE Transactions on Multimedia*, 2022.

[36] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," in *NeurIPS*, 2020, pp. 11 539–11 551.

[37] Y. Iwasawa and Y. Matsuo, "Test-time classifier adjustment module for model-agnostic domain generalization," in *NeurIPS*, 2021, pp. 2427–2440.

[38] Z. Wen, S. Niu, G. Li, Q. Wu, M. Tan, and Q. Wu, "Test-time model adaptation for visual question answering with debiased self-supervisions," *IEEE Transactions on Multimedia*, 2023.

[39] A. Dubey, V. Ramanathan, A. Pentland, and D. Mahajan, "Adaptive methods for real-world domain generalization," in *CVPR*, 2021, pp. 14 340–14 349.

[40] H. Huang, X. Gu, H. Wang, C. Xiao, H. Liu, and Y. Wang, "Extrapolative continuous-time bayesian neural network for fast training- free test-time adaptation," in *NeurIPS*, 2022, pp. 36 000–36 013.

[41] L. Zuo, B. Wang, L. Zhang, J. Xu, and X. Zhen, "Variational neuron shifting for few-shot image classification across domains," *IEEE Transactions on Multimedia*, 2023.

[42] Y. Liu, W. Zhang, and J. Wang, "Source-free domain adaptation for semantic segmentation," in *CVPR*, 2021, pp. 1215–1224.

[43] P. T. Sivaprasad and F. Fleuret, "Uncertainty reduction for model adaptation in semantic segmentation," in *CVPR*, 2021, pp. 9613–9623.

[44] I. Shin, Y. Tsai, B. Zhuang, S. Schulter, B. Liu, S. Garg, I. S. Kweon, and K. Yoon, "MM-TTA: multi-modal test-time adaptation for 3d semantic segmentation," in *CVPR*, 2022, pp. 16 907–16 916.

[45] J. Kim, I. Hwang, and Y. M. Kim, "Ev-tta: Test-time adaptation for event-based object recognition," in *CVPR*, 2022, pp. 17 724–17 733.

[46] K. Kotar and R. Mottaghi, "Interactron: Embodied adaptive object detection," in *CVPR*, 2022, pp. 14 840–14 849.

[47] V. VS, P. Oza, and V. M. Patel, "Towards online domain adaptive object detection," in *WACV*, 2023, pp. 478–488.

[48] C. Zhang, Z. Li, J. Liu, P. Peng, Q. Ye, S. Lu, T. Huang, and Y. Tian, "Self-guided adaptation: Progressive representation alignment for domain adaptive object detection," *IEEE Transactions on Multimedia*, vol. 24, pp. 2246–2258, 2021.

[49] F. Yang, K. Yan, S. Lu, H. Jia, D. Xie, Z. Yu, X. Guo, F. Huang, and W. Gao, "Part-aware progressive unsupervised domain adaptation for person re-identification," *IEEE Transactions on Multimedia*, vol. 23, pp. 1681–1695, 2020.

[50] M. Xu, M. Gao, Z. Gan, H.-Y. Chen, Z. Lai, H. Gang, K. Kang, and A. Dehghan, "Slowfast-llava: A strong training-free baseline for video large language models," *arXiv preprint arXiv:2407.15841*, 2024.

[51] M. Kim, T. Kim, and D. Kim, "Spatio-temporal slowfast self-attention network for action recognition," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 2206–2210.

[52] X. Liu, C. Yang, B. Luo, and W. Dai, "Suboptimal control for nonlinear slow-fast coupled systems using reinforcement learning and takagi–sugeno fuzzy methods," *International Journal of Adaptive Control and Signal Processing*, vol. 35, no. 6, pp. 1017–1038, 2021.

[53] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *TPAMI*, pp. 651–663, 2020.

[54] H. Zhao, Y. Fu, M. Kang, Q. Tian, F. Wu, and X. Li, "Mgsvf: Multi-grained slow vs. fast framework for few-shot class- incremental learning," *TPAMI*, 2021.

[55] J. Gao, X. Yao, and C. Xu, "Fast-slow test-time adaptation for online vision-and-language navigation," in *Forty-first International Conference on Machine Learning*.

[56] C. Ma, J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher, and C. Xiong, "Self-monitoring navigation agent via auxiliary progress estimation," in *ICLR*, 2019.

[57] H. Ye, Y. Ding, J. Li, and H. T. Ng, "Robust question answering against distribution shifts with test-time adaptation: An empirical study," in *EMNLP*, 2023.

[58] J. S. Vitter, "Random sampling with a reservoir," *ACM TOMS*, pp. 37–57, 1985.

[59] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *CVPR*, 2019, pp. 6629–6638.

[60] X. Li, C. Li, Q. Xia, Y. Bisk, A. Celikyilmaz, J. Gao, N. A. Smith, and Y. Choi, "Robust navigation with language pretraining and stochastic sampling," in *EMNLP-IJCNLP*, 2019, pp. 1494–1499.

[61] F. Zhu, Y. Zhu, X. Chang, and X. Liang, "Vision-language navigation with self-supervised auxiliary reasoning tasks," in *CVPR*, 2020, pp. 10 009–10 019.

[62] Y. Hong, C. R. Opazo, Y. Qi, Q. Wu, and S. Gould, "Language and visual entity relationship graph for agent navigation," in *NeurIPS*, 2020, pp. 7685–7696.

[63] Y. Qiao, Y. Qi, Y. Hong, Z. Yu, P. Wang, and Q. Wu, "HOP: history-and-order aware pre-training for vision-and-language navigation," in *CVPR*, 2022, pp. 15 418–15 427.

[64] A. X. Chang, A. Dai, T. A. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from RGB-D data in indoor environments," in *3DV*, 2017, pp. 667–676.

[65] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, Y. Choi, and J. Gao, "Oscar: Object-semantics aligned pre-training for vision-language tasks," in *ECCV*, 2020, pp. 121–137.

[66] J. Li, H. Tan, and M. Bansal, "Improving cross-modal alignment in vision language navigation via syntactic information," in *NAACL-HLT*, 2021, pp. 1041–1050.

[67] B. Lin, Y. Zhu, Z. Chen, X. Liang, J. Liu, and X. Liang, "ADAPT: vision-language navigation with modality-aligned action prompts," in *CVPR*, 2022, pp. 15 375–15 385.

[68] Y. Qi, Z. Pan, Y. Hong, M. Yang, A. van den Hengel, and Q. Wu, "The road to know-where: An object-and-room informed sequential BERT for indoor vision-language navigation," in *ICCV*, 2021, pp. 1635–1644.

[69] X. Lin, G. Li, and Y. Yu, "Scene-intuitive agent for remote embodied visual grounding," in *CVPR*, 2021, pp. 7036–7045.

Mingkui Tan is currently a professor with the School of Software Engineering at South China University of Technology. He received his Bachelor Degree in Environmental Science and Engineering in 2006 and Master degree in Control Science and Engineering in 2009, both from Hunan University in Changsha, China. He received the Ph.D. degree in Computer Science from Nanyang Technological University, Singapore, in 2014. From 2014-2016, he worked as a Senior Research Associate on computer vision in the School of Computer Science, University of Adelaide, Australia. His research interests include machine learning, sparse analysis, deep learning and large-scale optimization.

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2025.3535356

Peihao Chen received the B.E. degree in Automation Science and Engineering from South China University of Technology, China, in 2018. He is working toward the PhD degree in the School of Software Engineering, South China University of Technology, China. His research interests include embodied AI and multi-modal video understanding.

Hongyan Zhi received the B.E. degree inMechanical Manufacture and Automation from South ChinaUniversity of Technology, China, in 2023. He is working toward the M.E. degree in the School of Software Engineering, South China University ofTechnology, China. His research interests include Deep Learning in Embodied AI.

Jiajie Mai received his B.S. degree from the Beijing University of Posts and Telecommunications in 2020 and his Master's degree from King's College London in 2021. His research interests are non-convex optimization and Neural Networks.

Benjamin Rosman obtained the PhD. degree from the University of Edinburgh. His research interests include robotics, artificial intelligence, decision theory, and machine learning.

Dongyu Ji obtained the B.E. degree in the School of Automation Scienceand Engineering from South China University of Technology in 2020. He is currently working toward the M.E. degree in the School of Software Engineering from SouthChina University of Technology. His research interests include Deep Learning in Visual-and-Language Navigation.

Runhao Zeng received the PhD degree in software engineering from South China University of Technology, in 2021. He is currently an associate professor at the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University. He has authored or coauthored several peer-reviewed papers on computer vision, machine learning on top-tier conferences and journals, including the Proceedings of NeurIPS, CVPR, ICCV, and TPAMI. His current research interests include machine learning, computer vision, with particular focus on video analysis.