Vision-Language Navigation with Energy-Based Policy

Rui Liu[1,2] **Wenguan Wang**[2] **Yi Yang**[1,2,*]

[1]The State Key Lab of Brain-Machine Intelligence, Zhejiang University, Hangzhou, China
[2]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
https://github.com/DefaultRui/VLN-ENP

Abstract

Vision-language navigation (VLN) requires an agent to execute actions following human instructions. Existing VLN models are optimized through expert demonstrations by supervised behavioural cloning or incorporating manual reward engineering. While straightforward, these efforts overlook the accumulation of errors in the Markov decision process, and struggle to match the distribution of the expert policy. Going beyond this, we propose an Energy-based Navigation Policy (ENP) to model the joint state-action distribution using an energy-based model. At each step, low energy values correspond to the state-action pairs that the expert is most likely to perform, and vice versa. Theoretically, the optimization objective is equivalent to minimizing the forward divergence between the occupancy measure of the expert and ours. Consequently, ENP learns to globally align with the expert policy by maximizing the likelihood of the actions and modeling the dynamics of the navigation states in a collaborative manner. With a variety of VLN architectures, ENP achieves promising performances on R2R, REVERIE, RxR, and R2R-CE, unleashing the power of existing VLN models.

1 Introduction

Vision-language navigation (VLN) [1] entails an embodied agent that executes a sequence of actions to traverse complex environments based on natural language instructions. VLN is typically considered as a Partially Observable Markov Decision Process (POMDP) [2], where future states are determined only by the current state and current action without explicit rewards. Therefore, the central challenge for VLN is to learn an effective navigation policy. In this context, existing neural agents [1, 3, 4, 5, 6] are naturally constructed as data-driven policy networks by imitation learning (IL) [7] and bootstrapping from expert demonstrations, *i.e.*, ground-truth trajectories [1].

Most VLN models [1, 3, 8, 4, 5, 6] utilize behavioural cloning (BC), a classic approach for IL, through supervised training. While conceptually simple, BC suffers seriously from quadratic accumulation errors [9, 10] along the trajectory due to covariate shift, especially in partially observable settings. Several efforts introduce 'student-forcing' [1, 11] and 'pseudo interactive demonstrator' [6], which are essentially online versions of DAGGER [12], to alleviate this distribution mismatch. DAGGER assumes interactive access to expert policies (*i.e.*, the shortest path from current state to the goal[2]) and reduces to linear expected errors [12, 10]. Some other studies [13, 3, 8, 4, 5] combine reinforcement learning (RL) [14] and IL algorithms to mitigate the limitations of BC. Though effective, it presents challenges in designing an optimal reward function [15] and demands careful manual reward engineering. Such reward functions may not be robust to changes in the environment dynamics and

*Corresponding author: Yi Yang.

[2]These approaches require a slightly more interactive setting than traditional imitation learning, allowing the learner to query the expert at any navigation state during training. Previous studies [1, 6] have adopted this setting, and this assume is feasible for many real-world imitation learning tasks [12].

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

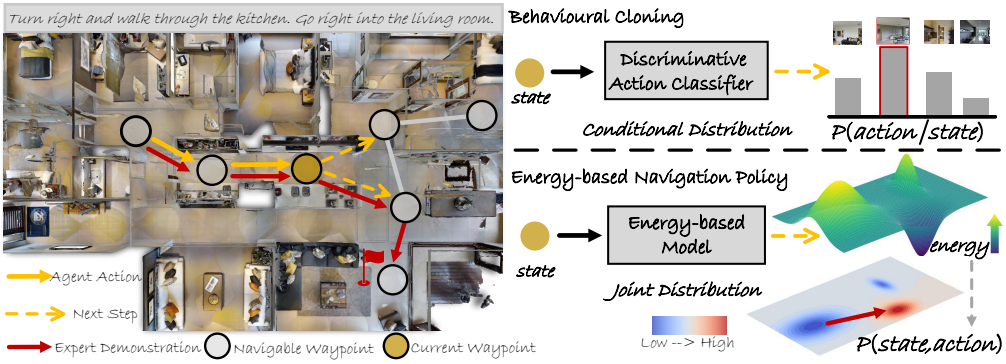

Figure 1: Comparison of behavioural cloning (BC) and ENP for VLN. Previous methods use BC to optimize the conditional action distribution directly. ENP models the joint state-action distribution through an energy-based model. The low energy values correspond to the state-action pairs that the expert is most likely to perform.

struggle to generalize well across diverse scenes [16]. In addition, recent research [6] has revealed that training transformer [17]-based VLN models using RL would introduce instability.

From a probabilistic view, most VLN agents [1, 3, 4, 6] essentially train a discriminative action classifier [8, 3, 4], and model the action distribution conditioned on current navigation state $P(a|s)$ at each time step (Fig. 1). This classifier is supervised by the cross-entropy loss to minimize 1-step deviation error of single-step decision along trajectories [18]. However, the prediction of current action affects future states during the global execution of the learned policy, potentially leading to catastrophic failures [9, 12]. In other words, solely optimizing the conditional action probabilities (*i.e.*, policy) remains unclear for the underlying reasons of the expert behaviour, yet fails to exploit complete information in the state-action distribution $P(s, a)$ induced by the expert [14].

In light of this, we propose an Energy-based Navigation Policy (ENP) framework that addresses the limitations of current policy learning in VLN from an energy perspective (Fig. 1). Energy-based models (EBMs) [19] are flexible and capable of modeling expressive distributions, as they impose minimal restrictions on the tractability of the normalizing constant (partition function) [20]. We reinterpret the VLN model as an energy-based model to learn the expert policy from the expert demonstrations. Specifically, ENP models the joint distribution of state-action pairs, *i.e.*, $P(s, a)$, by energy function, and assigns low energy values to the state-action pairs that the expert mostly performs. For optimization, this energy function is regarded as the unnormalized $\log$-density of joint distribution, which can be decomposed into the standard cross-entropy of action prediction and marginal distribution of the states $P(s)$. During training, ENP introduces persistent contrastive divergence [21, 22] to estimate the expectation of $P(s)$, and samples by Stochastic Gradient Langevin Dynamics [23]. In this way, ENP simulates the expert by maximizing the likelihood of the action and incorporates prior knowledge of VLN by optimizing state marginal distribution in a collaborative manner, thereby leveraging the benefits of both energy-based and discriminative approaches.

Theoretically, the training objective of ENP is to maximize the expected $\log$-likelihood function over the joint distribution $P(s, a)$. This is equivalent to estimating the unnormalized probability density (*i.e.*, energy) of the expert's occupancy measure [15, 24, 25]. Therefore, ENP performs prioritized optimization for entire trajectories rather than the single-time step decisions in BC, and achieves a global alignment with the expert policy. Furthermore, we realize that ENP is closely related to Inverse Reinforcement Learning (IRL) [7, 26, 15]. For the optimized objective, ENP shares similarities with IRL but minimizes the forward KL divergence [27, 28] between the occupancy measures.

For thorough examination (§4), we explore ENP across several representative VLN architectures [3, 4, 6, 29]. Experimental results demonstrate that ENP outperforms the counterparts, *e.g.*, $\mathbf{2}\%$ SR and $\mathbf{1}\%$ SPL gains over R2R [1], $\mathbf{1.22}\%$ RGS on REVERIE [30], $\mathbf{2}\%$ SR on R2R-CE [31], and $\mathbf{1.07}\%$ NDTW on RxR-CE [32], respectively. ENP not only helps release the power of existing VLN models, but also evidences the merits of energy-based approaches in the challenging VLN decision-making.

2 Related Work

Vision-Language Navigation (VLN). Early VLN agents [1, 13, 3, 33] are built upon a recurrent neural policy network within a sequence-to-sequence framework [34, 35] to predict action distribution.

As compressing all history into a hidden vector might be sub-optimal for state tracking across extended trajectories, later attempts [36, 37, 6, 38] incorporate a memory buffer (*e.g.*, topological graph) for long-term planning. Recent efforts [39, 5, 4] are devoted to encode complete episode history of navigation states and actions by transformer [17] and optimize the whole model in end-to-end training. Later, various strategies have been proposed to improve the generalization of policies in both seen and unseen environments [3, 8, 40, 41], such as progress estimation [42], instruction generation [33, 43, 44, 45, 46], backtracking [47], cross-modal matching [48, 49], self-supervised pre-training [50, 51, 52], environmental augmentation [3, 53, 54, 55], visual landmarks [56, 57], exploration [58, 59], map building [60, 61, 62, 63], waypoint prediction [64], and foundation models [65, 66].

Policy Learning in VLN. VLN can be viewed as a *POMDP* [2], providing several expert demonstrations sourced from ground-truth shortest-path trajectories [1, 30] or human demonstrations [32]. Behavioural cloning (BC) [67, 68], a typical approach of imitation learning (IL) [7, 15], is widely used in current VLN agents [1, 33, 4, 6] with supervised policy learning. Nevertheless, it suffers from distribution shifts between training and testing [12, 69]. [1] introduces 'student-forcing' [11] training regime from sequence prediction to mitigate this limitation. Later agents [3, 8, 4, 5] combine both IL and model-free RL [14, 70] for policy learning, where the reward function is manually designed on the distance metric. Instead of directly mapping visual observations into actions or state-action values, [13, 71] investigate model-based RL [72, 73] for look-ahead planning, exploring the potential future states. As reward engineering requires careful manual tuning, [16] proposes to learn reward functions from the expert distribution directly. Although RL is an effective approach in principle, DUET [6] finds it difficult to train large-scale transformers with inaccurate and sparse RL rewards [74] in VLN. Existing studies [61, 62, 63] use an interactive demonstrator (similar to DAGGER [12]) to generate additional pseudo demonstrations and achieve promising performance.

In this paper, ENP learns to align with the expert policy by matching the joint state-action distribution for entire trajectories from the occupancy measure view. Furthermore, ENP reinterprets the optimization of policy into an energy-based model, representing an expressive probability distribution.

Energy-based Model. Energy-based models (EBMs) [19] are non-normalized probabilistic models that represent data using a Boltzmann distribution. In EBMs, each sample is associated with an energy value, where high-density regions correspond to lower energy [75]. A range of MCMC-based [76, 21] and MCMC-free [77, 78] approaches can be adopted to estimate the gradient of the $\log$-likelihood. EBMs have been widely utilized in generation [22], perception [79], and policy learning [80, 81]. The most related work [25, 24, 22] will be discussed below. [25] employs EBM for distribution matching, strictly prohibiting interaction with the environment. Moreover, this approach is applied in an offline setting, specifically for healthcare. [24] considers the expert energy as the reward function and leverages this reward for reinforcement learning. [22] adopts parameterized energy functions to model the conditional action distribution, optimizing for actions that minimize the energy landscape.

In contrast, ENP devotes to learning the joint distribution of state-action pairs from the expert demonstrations online. An external marginal state memory is introduced to store the historical information for initializing the samples for next iterations. ENP demonstrates the potential of EBMs in VLN decision-making, and improves the existing VLN agents across various frameworks.

3 Method

In this section, we first formalize the VLN task and discuss the limitations of existing work from a probabilistic view (§3.1). Then we introduce Energy-based Navigation Policy learning framework – ENP (§3.2). Furthermore, ENP is compared with other imitation learning methods based on divergence minimization (§3.3). Finally, we provide the implementation details (§3.4).

3.1 Problem Formulation

VLN is typically formulated as a POMDP [2, 51] with a finite horizon T, defined as the tuple $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0>$. $\mathcal{S}$ is the navigation state space. $\mathcal{A}$ is the action space. Note that we use panoramic action space [33], which encodes high-level behaviour. The transition distribution $\mathcal{T}(s_{t+1}|s_t, a_t)$ (Markov property) of the environment, the reward function $\mathcal{R}(s_t, a_t)$, and the initial state distribution ρ_0 are unknown in VLN and $\mathcal{T}$ can only be queried through interaction with the environment.

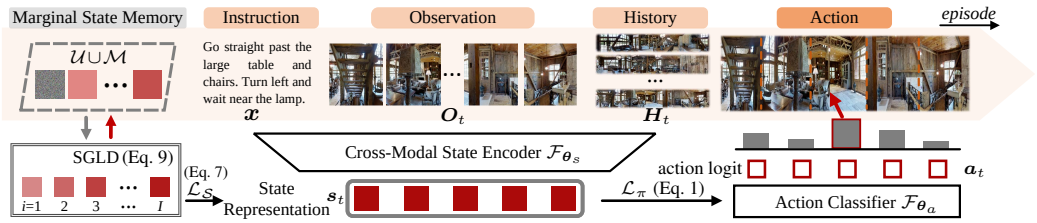

Figure 2: Overview of ENP. At each step t, the agent acquires a series of observations, and predicts the next step based on the instruction and navigation history. ENP optimizes the marginal state matching loss $\mathcal{L}_{\mathcal{S}}$ through SGLD sampling from Marginal State Memory (Eq. 9), and minimizes the cross-entropy loss $\mathcal{L}_\pi$ jointly (Eq. 1).

In the standard VLN setting [1], a training dataset $\mathcal{D} = \{(x, \tau)\}$ consists of pairs of the language instructions x and corresponding expert trajectories τ. At step t, the agent acquires an egocentric observation O_t, and then takes an action $a_t \in \mathcal{A}$ from K_t candidates based on the instruction x. The agent needs to learn a navigation policy $\pi^*(a|s)$ and predict the action distribution $P(a_t|\boldsymbol{s}_t)$.

Recent VLN agents employ Seq2Seq [8, 3] or transformer-based networks [4, 6] as a combination of a cross-modal state encoder $\mathcal{F}_{\boldsymbol{\theta}_s}$ and a conditional action classifier $\mathcal{F}_{\boldsymbol{\theta}_a}$ for action prediction. Hence they are usually built as a composition of $\mathcal{F}_{\boldsymbol{\theta}_s} \diamond \mathcal{F}_{\boldsymbol{\theta}_a}$. There are different state representations in previous work [51, 4], here we define a more general one. The cross-modal state encoder extracts a joint state representation $\boldsymbol{s}_t = \mathcal{F}_{\boldsymbol{\theta}_s}(\boldsymbol{x}, \boldsymbol{O}_t, \boldsymbol{H}_t) \in \mathbb{R}^{K_t \times D}$ by the visual embedding $\boldsymbol{O}_t$, the instruction embedding $\boldsymbol{x}$, and history embedding $\boldsymbol{H}_t$ along the trajectory τ. Concretely, $\boldsymbol{H}_t$ comes from recurrent hidden units of Seq2Seq framework or an external memory module (*e.g.*, topological graph [6]). Then, the action classifier maps the state representation $\boldsymbol{s}_t$ into an action-related vector $\boldsymbol{a}_t = \{\boldsymbol{a}_t[k]\}_{k=1}^{K_t} \in \mathbb{R}^{K_t}$, *i.e.*, $\boldsymbol{a}_t = \mathcal{F}_{\boldsymbol{\theta}_a}(\boldsymbol{s}_t)$, and $\boldsymbol{a}_t[k]$ is termed as the logit for k-th action candidate. Then the probability of $a_{t,k}$ is computed by softmax, *i.e.*, $P(a_{t,k}|\boldsymbol{s}_t) = \frac{\exp(\boldsymbol{a}_t[k])}{\sum_{k=1}^{K} \exp(\boldsymbol{a}_t[k])}$.

The cross-modal state encoder $\mathcal{F}_{\boldsymbol{\theta}_s}$ and action classifier $\mathcal{F}_{\boldsymbol{\theta}_a}$ usually employ a joint end-to-end training. Imitation learning (IL) aims to learn the policy from expert demonstrations $\mathcal{D}$ without any reward signals. Behaviour cloning (BC) [68] is a straightforward approach of IL by maximizing likelihood estimation of $\pi^*(a|s)$, *i.e.*, minimizing the cross-entropy loss in a supervised way:

$$\arg\min_{\boldsymbol{\theta}} -\mathbb{E}[\log \pi^*(a|s)] \rightarrow \mathcal{L}_\pi = \sum\nolimits_{(x,\tau)\sim\mathcal{D}} \sum\nolimits_t -\log P_{\boldsymbol{\theta}}(a_t|\boldsymbol{s}_t), \tag{1}$$

where $\boldsymbol{\theta} = \boldsymbol{\theta}_s \cup \boldsymbol{\theta}_a$. BC suffers from the covariate shift problem for the crucial *i.i.d.* assumption, as the execution of a_t will affect the distribution of future state $\boldsymbol{s}_{t+1}$ [9]. Inspired by DAGGER [12], some agents [1, 6] query the shortest paths (by expert policy π^{e}) during navigation, explore more trajectories $\{\tau^+\}$ by interacting with the environments, and aggregate these trajectories as $\mathcal{D}^+ = \{(x, \tau)\} \cup \{(x, \tau^+)\}$. Then, the policy is updated by mixing iterative learning [12]. However, these traditional approaches ignore the changes in distribution and train a policy that performs well under the distribution of states encountered by the expert. When the agent navigates in the unseen environment and receives different observations than expert demonstrations, it will lead to a compounding of errors, as they provide no way to understand the underlying reasons for the expert behaviour [27].

3.2 Energy-based Expert Policy Learning

Instead of solely training the discriminative action classifier at each step, we want to learn a policy π^* by exploiting the complete information in the state-action distribution from the expert policy π^{e}. To facilitate later analysis, the occupancy measure $\rho^\pi(s, a)$ of the policy π is introduced [14]. $\rho^\pi(s, a)$ denotes the stationary distribution of state-action pairs (s, a) when an agent navigates with policy π in the environment, defined as:

$$\rho^\pi(s, a) \triangleq \pi(a|s) \sum\nolimits_t P(s_t = s|\pi) = \sum\nolimits_t P(s_t = s, a_t = a), \tag{2}$$

and $\rho^\pi(s, a)$ has a one-to-one relationship with policy π: $\rho^\pi = \rho^{\pi^*} \Leftrightarrow \pi = \pi^*$ [15]. Therefore, we induce a policy π^* with $\rho^*(s, a)$ that matches the expert $\rho^{\mathrm{e}}(s, a)$ based on the joint distribution of state-action pairs $P(s, a)$. t is omitted for brevity. The matching loss $\mathcal{L}_\rho$ is designed as:

$$\arg\min -\mathbb{E}_{(s,a)\sim\rho^{\mathrm{e}}(s,a)}[\log \rho^*(s, a)] \propto \mathcal{L}_\rho = -\sum\nolimits_{(x,\tau)\sim\mathcal{D}} \log P_{\boldsymbol{\theta}}(s, a). \tag{3}$$

Then we reformulate it from an energy view, as energy-based models (EBMs) [19, 79] parameterize an unnormalized data density and can represent a more expressive probability distribution. Specifically,

EBMs are first introduced to model $P(s,a)$ of the agent as a Boltzmann distribution:

$$P_{\boldsymbol{\theta}}(s,a) = \frac{\exp(-E_{\boldsymbol{\theta}}(s,a))}{Z_{\boldsymbol{\theta}}}, \tag{4}$$

where $E_{\boldsymbol{\theta}}(s,a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the energy function, and $Z_{\boldsymbol{\theta}}$ is the partition function (normalizing constant). As it is difficult to optimize the joint distribution directly [79], we factorize it as (Fig. 2):

$$-\log P_{\boldsymbol{\theta}}(s,a) = -\log P_{\boldsymbol{\theta}}(a|s) - \log P_{\boldsymbol{\theta}}(s) \rightarrow \mathcal{L}_{\rho} = \mathcal{L}_{\pi} + \mathcal{L}_{\mathcal{S}}. \tag{5}$$

The first term, $\log P(a|s)$, is typically from a discriminative classifier and can be optimized by minimizing the cross-entropy loss $\mathcal{L}_{\pi}$ as in Eq. 1. As stated in [82, 83], the marginal state distribution matching loss $\mathcal{L}_{\mathcal{S}}$ provides an explicit objective for navigation exploration and incorporates prior knowledge from the expert demonstrations. $P(s)$ is computed by marginalizing out a as:

$$P_{\boldsymbol{\theta}}(s) = \sum\nolimits_{k=1}^{K} P_{\boldsymbol{\theta}}(s,a_k) = \frac{\sum_{k=1}^{K} \exp(-E_{\boldsymbol{\theta}}(s,a_k))}{Z_{\boldsymbol{\theta}}}. \tag{6}$$

It is worth noting that $\log P(s)$ is the likelihood of an EBM, which cannot be computed directly due to the intractable partition function $Z_{\boldsymbol{\theta}}$. A common approach is to estimate the gradient of the log-likelihood for likelihood maximization with gradient ascent [75]:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{S}} \rightarrow -\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(s) &= -\nabla_{\boldsymbol{\theta}} \log \sum\nolimits_{k=1}^{K} \exp(-E_{\boldsymbol{\theta}}(s,a_k)) + \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} \\ &= \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(s) - \mathbb{E}_{\hat{s} \sim P_{\boldsymbol{\theta}}(\hat{s})}[\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\hat{s})]. \end{aligned} \tag{7}$$

Considering the low energy value corresponds to the state-action pairs that the expert mostly perform, the action logits $\boldsymbol{a}_t[k]$ of the agent can be used to describe such energy value as $E_{\boldsymbol{\theta}}(s_t, a_{t,k}) \triangleq -a_{t,k}$ formally. Then, the first gradient term, $-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(s)$, is calculated by automatic differentiation in the deep network of the neural agent. Estimating $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$ requires Markov chain Monte Carlo (MCMC) [84] sampling from the Boltzmann distribution within the inner loop of learning. Specifically, a sampler based on Stochastic Gradient Langevin Dynamics [23] (SGLD), a practical method of Langevin MCMC [85]) can effectively draw samples as:

$$\hat{s}^0 \sim \mathcal{U}, \quad \hat{s}^{i+1} = \hat{s}^i - \frac{\epsilon^2}{2} \nabla_{\hat{s}} E_{\boldsymbol{\theta}}(\hat{s}^i) + \xi, \quad \xi \sim \mathcal{N}, \tag{8}$$

where $\mathcal{U}$ denotes the uniform distribution, i is the sampling iteration with step size ϵ, ξ is Gaussian noise, and $\hat{s}^i$ will converge to $P_{\boldsymbol{\theta}}(\hat{s})$ when $\epsilon \rightarrow 0$ and $i \rightarrow \infty$. While training on a new state, this MCMC chain will be reset. In practice, adopting standard MCMC sampling suffers from slow convergence and requires expensive computation. Recent persistent contrastive divergence approaches [76, 21] maintain a single MCMC chain, and use it to initialize a new chain for next sampling iteration [22].

Inspired by this, we introduce a Marginal State Memory $\mathcal{M}$ served as an online replay buffer [22], collecting historical information of the MCMC chain during training (Fig. 2). To initialize the inner loop of SGLD, the state $\hat{s}^0$ is sampled randomly from $\mathcal{U} \cup \mathcal{M}$. This helps continue to refine the previous samples, further accelerating the convergence. After several iterations I, $\hat{s}^I$ is stored in $\mathcal{M}$:

$$\hat{s}^0 \sim \mathcal{U} \cup \mathcal{M}, \quad \hat{s}^{i+1} = \hat{s}^i - \frac{\epsilon^2}{2} \nabla_{\hat{s}} E_{\boldsymbol{\theta}}(\hat{s}^i) + \xi, \quad \hat{s}^I \rightarrow \mathcal{M}, \tag{9}$$

where I is fewer than the number of iterations required for MCMC convergence. The training details are illustrated in Algorithm 1.

3.3 Comparison with Other Imitation Learning Methods

In VLN, ENP aims to learn a policy π^* from expert demonstrations (π^{e}) without access to the explicit reward, which is analogous to the most common approaches of IL, *e.g.*, BC and inverse reinforcement learning (IRL). For in-depth analysis, we explore the relation between ENP and other IL Methods, and provide a general understanding of them from a divergence viewpoint [28].

Standard BC is treated as a supervised learning problem (Eq. 1), and the policy is required to match the conditional distribution of π^{e} under Kullback–Leibler (KL) divergence distance D_{KL} as:

$$\arg\min -\mathbb{E}[\log \pi^*(a|s)] = D_{\mathrm{KL}}(\pi^{\mathrm{e}}(a|s) \| \pi^*(a|s)) - \mathbb{E}(\pi^{\mathrm{e}}(a|s)), \tag{10}$$

where $\mathcal{H}(\pi^{\mathrm{e}}) = \mathbb{E}(\pi^{\mathrm{e}}(a|s))$ is a constant, *a.k.a.* the causal entropy [86]. $\mathcal{H}(\pi^{\mathrm{e}})$ is considered as the expected number of options at each state for $\pi^{\mathrm{e}}(a|s)$ [28, 15].

Algorithm 1 Energy-based Navigation Policy (ENP) Learning Algorithm

Input: The training dataset $\mathcal{D} = \{(x, \tau)\}$, Cross-Modal State Encoder $\mathcal{F}_{\boldsymbol{\theta}_s}$, Action Classifier $\mathcal{F}_{\boldsymbol{\theta}_a}$, Visual embedding $\boldsymbol{O}_t$, Instruction embedding $\boldsymbol{x}$, History embedding $\boldsymbol{H}_t$
Initialize: $\boldsymbol{\theta} = \boldsymbol{\theta}_s \cup \boldsymbol{\theta}_a$, and Marginal State Memory $\mathcal{M} \leftarrow \varnothing$ or $\mathcal{M}_{\text{ft}} \leftarrow \mathcal{M}_{\text{pre}}$
for training step $n = 1$ to N **do**
  Aggragate $\mathcal{D}_n^+ = \{(x, \tau_n^+)\}$ of visited states by π_n^* and actions from π^{e}. Sample (x, τ) from $\mathcal{D}_n^+ \cup \mathcal{D}$
  for navigation step $t = 1$ to T **do**
    $\boldsymbol{s}_t = \mathcal{F}_{\boldsymbol{\theta}_s}(\boldsymbol{x}, \boldsymbol{O}_t, \boldsymbol{H}_t)$, $\boldsymbol{a}_t = \mathcal{F}_{\boldsymbol{\theta}_a}(\boldsymbol{s}_t)$, Sample $\hat{\boldsymbol{s}}_t^0$ from $\mathcal{U} \cup \mathcal{M}$ ▷ Defined in §3.1.
    for SGLD iteration $i = 1$ to I **do**
      $\hat{\boldsymbol{s}}_t^{i+1} = \hat{\boldsymbol{s}}_t^i - \frac{\epsilon^2}{2} \nabla_{\hat{\boldsymbol{s}}} E_\theta(\hat{\boldsymbol{s}}_t^i) + \xi,\ \xi \sim \mathcal{N},\ \hat{\boldsymbol{s}}_t^I \rightarrow \mathcal{M}$ ▷ Defined in Eq. (9).
    $\tilde{\mathcal{L}}_\pi \leftarrow -\log P_{\boldsymbol{\theta}}(a_t | \boldsymbol{s}_t),\ \ \tilde{\mathcal{L}}_{\mathcal{S}} \leftarrow \big(E_{\boldsymbol{\theta}}(\boldsymbol{s}_t) - E_{\boldsymbol{\theta}}(\hat{\boldsymbol{s}}_t^I)\big)$ ▷ Defined in Eq. (1) and (7).
  Backpropagate $\nabla_{\boldsymbol{\theta}} \mathcal{L}_\rho = \nabla_{\boldsymbol{\theta}} \mathcal{L}_\pi + \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{S}}$. Update $\mathcal{F}_{\boldsymbol{\theta}_s}$ and $\mathcal{F}_{\boldsymbol{\theta}_a}$
Return: Optimized Parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_s \cup \boldsymbol{\theta}_a$ of $\mathcal{F}_{\boldsymbol{\theta}_s}$ and $\mathcal{F}_{\boldsymbol{\theta}_a}$

For ENP, the optimization objective in Eq. 3 can be written in a form of KL distance minimization:

$$\arg\min -\mathbb{E}_{(s,a)\sim\rho^{\text{e}}(s,a)}[\log \rho^*(s,a)] = D_{\text{KL}}\big(\rho^{\text{e}}(s,a) || \rho^*(s,a)\big) + \mathcal{H}(\rho^{\text{e}}(s,a)), \tag{11}$$

where $\mathcal{H}(\rho^{\text{e}}(s,a))$ is the entropy of the occupancy measure, which is a constant. IRL aims to infer the reward function of the expert, and learn a policy to optimize this reward. A representative work, AIRL [27], adopts an adversarial reward learning, which is equivalent to matching $\rho^{\text{e}}(s,a)$ [15] as:

$$\arg\min D_{\text{KL}}\big(\rho^*(s,a) || \rho^{\text{e}}(s,a)\big) = -\mathbb{E}_{\rho^*(s,a)}[\log \rho^{\text{e}}(s,a)] - \mathcal{H}(\rho^*(s,a)). \tag{12}$$

Hence, our objective is similar to the forward KL version of AIRL [28, 27] except for the entropy term. Both AIRL and ENP perform prioritized optimization for entire trajectories rather than the single-time step decisions in BC, so the compounding error can be mitigated [15, 27]. As D_{KL} is a non-symmetric metric, the forward KL divergence results in distributions with a mode-covering behaviour, while using the reverse KL results in mode-seeking behaviour [18]. In addition, ENP directly models the joint distribution instead of recovering the reward function. We compare them in the experiment (§4.3) and find ENP is more applicable to VLN (Table 8).

3.4 Implementation Details

Network Architecture. ENP is a general framework that can be built upon existing VLN models. We investigate ENP on several representative architectures, including: i) EnvDrop [3] adopts a Seq2Seq model with a bidirectional LSTM-RNN and an attentive LSTM-RNN. ii) VLN↻BERT [4] is built upon multi-layer transformers with recurrent states. iii) DUET [6] is a dual-scale transformer-based model with a topological map for global action space, enabling long-term decision-making. iv) ETPNav [29] utilizes a transformer-based framework with an online topological map. For continuous environments, both VLN↻BERT [4] and ETPNav [29] employ a waypoint predictor [64]. K_t dimension of the state representations $\boldsymbol{s}_t \in \mathbb{R}^{K_t \times D}$ is different due to the local or global action spaces of these methods. In §4, we follow the same settings of these models, replace their policy learning as ENP, and maintain a Marginal State Memory $\mathcal{M}$ of 100 buffer-size.

Training. For fairness, the hyper-parameters (*e.g.*, batch size, optimizers, maximal iterations, learning rates) of these models are kept the original setup. Back translation is employed to train EnvDrop [3]. The parameters of VLN↻BERT [4] is initialized from PREVALENT [51]. The pre-training and finetuning paradigm is used for DUET [6] and ETPNav [29]. For SGLD, $\hat{s}^0$ is sampled from $\mathcal{M}$ with a probability of 95% (Eq. 9). In EnvDrop [3] and VLN↻BERT [4], the number of SGLD iterations is set as $I = 15$. $I_{\text{pre}} = 20$ and $I_{\text{ft}} = 5$ of the pre-training and finetuning stages are used for DUET [6] and ETPNav [29]. Although the step size ϵ and the Gaussian noise ξ in the original transition kernel of SGLD are related by $\text{Var}(\xi) = \epsilon$ [23], most studies [22, 79] choose to set ϵ and ξ separately in practice. The step size $\epsilon = 1.5$ and $\xi \sim \mathcal{N}(0, 0.01)$ are set in ENP (§4.3).

Inference. During the testing phase, the agent uses the learned $\mathcal{F}_{\boldsymbol{\theta}_s}$ for cross-modal state encoding, and predicts the next step via $\mathcal{F}_{\boldsymbol{\theta}_a}$, until it chooses [STOP] action or reaches the maximum step limit.

Reproducibility. All experiments are conducted on a single NVIDIA 4090 GPU with 24GB memory in PyTorch. Testing is conducted on the same machine.

4 Experiment

We examine the efficacy of ENP for VLN in discrete environments (§4.1), and more challenging continuous environments (§4.2). Then we provide diagnostic analysis on core model design (§4.3).

4.1 VLN in Discrete Environments

Datasets. R2R [1] contains 7,189 shortest-path trajectories captured from 90 real-world building-scale scenes [87]. It consists of 22K step-by-step navigation instructions. REVERIE [30] contains 21,702 high-level instructions, which requires an agent to reach the correct location and localize a remote target object. All these datasets are devided into *train*, *val seen*, *val unseen*, and *test unseen* splits, which mainly focus on the generalization capability in unseen environments. Both R2R [1] and REVERIE [30] are built upon Matterport3D Simulator [1].

Evaluation Metrics. As in previous work [1, 30], Success Rate (SR), Trajectory Length (TL), Oracle Success Rate (OSR), Success rate weighted by Path Length (SPL), and Navigation Error (NE) are used for R2R. In addition, Remote Grounding Success rate (RGS) and Remote Grounding Success weighted by Path Length (RGSPL) are adopted for object grounding in REVERIE.

Performance on R2R. Table 1 demonstrates the numerical results of ENP with different frameworks on R2R. With Seq2Seq-style neural architectures, ENP provides **1**% SR and **1**% SPL gains over EnvDrop [3] on *val unseen*. Adopting ENP in transformer-based models leads to promising performance gains on *test unseen*, *e.g.*, **2**% on SR with VLN↻BERT [4], and **2**% on SR with DUET [6]. The performance is still lower than BEVBert [61] and BSG [62] due to the additional semantic maps. Moreover, we provide the average success rate across different path lengths in Appendix (Fig. 4).

Table 1: Quantitative comparison results on R2R [1] (§4.1).

Models	R2R *val unseen*				R2R *test unseen*			
	TL↓	NE↓	SR↑	SPL↑	TL↓	NE↓	SR↑	SPL↑
Seq2Seq [1] [CVPR2018]	8.39	7.81	22	–	8.13	7.85	20	18
SF [33] [NeurIPS2018]	–	6.62	35	–	14.82	6.62	35	28
AuxRN [48] [CVPR2020]	–	5.28	55	50	–	5.15	55	51
Active [88] [ECCV2020]	20.60	4.36	58	40	21.60	4.33	60	41
HAMT [5] [NeurIPS2021]	11.46	2.29	66	61	12.27	3.93	65	60
SSM [38] [CVPR2021]	20.7	4.32	62	45	20.4	4.57	61	46
HOP [89] [CVPR2022]	12.27	3.80	64	57	12.68	3.83	64	59
TD-STP [90] [MM2022]	–	3.22	70	63	–	3.73	67	61
LANA [91] [ICCV2023]	12.00	–	68	62	12.60	–	65	60
BSG [62] [ICCV2023]	14.90	2.89	74	62	14.86	3.19	73	62
BEVBert [61] [ICCV2023]	14.55	2.81	75	64	–	3.13	73	62
VER [92] [CVPR2024]	14.83	2.80	76	65	15.23	2.74	76	66
EnvDrop [3] [NAACL2019]	10.70	5.22	52	48	11.66	5.23	51	47
ENP–EnvDrop	11.17	**4.69**	**53**	**49**	**10.78**	5.30	**52**	**48**
VLN↻BERT [4] [CVPR2021]	12.01	3.93	63	57	12.35	4.09	63	57
ENP–VLN↻BERT	12.17	**3.81**	**65**	57	13.12	**3.91**	**65**	**59**
DUET [6] [CVPR2022]	13.94	3.31	72	60	14.73	3.65	69	59
ENP–DUET	14.23	**3.00**	**74**	60	**14.27**	**3.39**	**71**	**60**

Performance on REVERIE. Table 2 presents the results of ENP with different transformer-based frameworks on REVERIE. ENP surpasses all the counterparts on both *val unseen* and *test unseen* by large margins, *e.g.*, **1.34**% on SR and **0.64**% on RGS over VLN↻BERT [4], **0.68**% on SR and **1.22**% on RGS over DUET [6], verifying the efficacy of ENP. Inspired by recent efforts [61], potential improvement on object grounding can be achieved by introducing CLIP features [93].

Table 2: Quantitative comparison results on REVERIE [30] (§4.1). '−': unavailable statistics.

Models	REVERIE *val unseen*						REVERIE *test unseen*					
	TL↓	OSR↑	SR↑	SPL↑	RGS↑	RGSPL↑	TL↓	OSR↑	SR↑	SPL↑	RGS↑	RGSPL↑
Seq2Seq [1] [CVPR2018]	11.07	8.07	4.20	2.84	2.16	1.63	10.89	6.88	3.99	3.09	2.00	1.58
RCM [8] [CVPR2019]	11.98	14.23	9.29	6.97	4.89	3.89	10.60	11.68	7.84	6.67	3.67	3.14
INP [30] [CVPR2020]	45.28	28.20	14.40	7.19	7.84	4.67	39.05	30.63	19.88	11.61	11.28	6.08
Airbert [94] [ICCV2021]	18.71	34.51	27.89	21.88	18.23	14.18	17.91	34.20	30.28	23.61	16.83	13.28
HAMT [5] [NeurIPS2021]	14.08	36.84	32.95	30.20	18.92	17.28	13.62	33.41	30.40	26.67	14.88	13.08
HOP [89] [CVPR2022]	16.46	36.24	31.78	26.11	18.85	15.73	16.38	33.06	30.17	24.34	17.69	14.34
TD-STP [90] [MM2022]	–	39.48	34.88	27.32	21.16	16.56	–	40.26	35.89	27.51	19.88	15.40
LANA [91] [ICCV2023]	23.18	52.97	48.31	33.86	32.86	22.77	18.83	57.20	51.72	36.45	32.95	22.85
BSG [62] [ICCV2023]	24.71	58.05	52.12	35.59	35.36	24.24	22.90	62.83	56.45	38.70	33.15	22.34
BEVBert [61] [ICCV2023]	–	56.40	51.78	36.37	34.71	24.44	–	57.26	52.81	36.41	32.06	22.09
VER [92] [CVPR2024]	23.03	61.09	55.98	39.66	33.71	23.70	24.74	62.22	56.82	38.76	33.88	23.19
VLN↻BERT [4] [CVPR2021]	16.78	35.02	30.67	24.90	18.77	15.27	15.86	32.91	29.61	23.99	16.50	13.51
ENP–VLN↻BERT	**16.90**	**35.80**	**31.27**	**25.22**	**18.78**	**15.80**	15.90	**34.00**	**30.95**	**24.46**	**17.14**	**14.12**
DUET [6] [CVPR2022]	22.11	51.07	46.98	33.73	32.15	23.03	21.30	56.91	52.51	36.06	31.88	22.06
ENP–DUET	25.76	**54.70**	**48.90**	**33.78**	**34.74**	**23.39**	22.70	**59.38**	**53.19**	**36.26**	**33.10**	**22.14**

Qualitative Results. In Fig. 3 (a), we illustrate the qualitative comparisons of ENP against DUET [6] on R2R *val unseen*. Based on ENP, our agent yields more accurate predictions than DUET. It verifies that ENP leads to better decision-making when facing challenging scenes. In Fig. 3 (b), our agent may fail due to the serious occlusion, and introducing map prediction [95] may alleviate this problem.

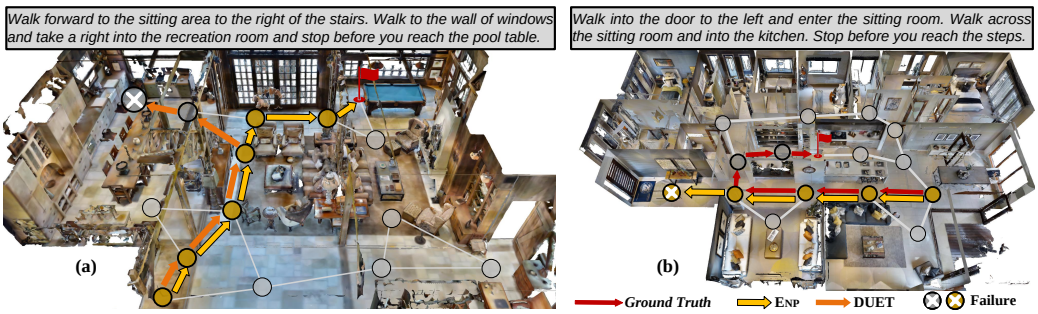

Figure 3: Qualitative results on R2R [1] (§4). (a) DUET [6] arrives in the wrong room instead of 'recreation room' since the scene contains multiple rooms. Our agent reaches the goal successfully, demonstrating better decision-making ability. (b) Failure case: Due to partial observability and occlusion of the environment, it is hard to find 'kitchen' at some positions. Thus our agent goes the wrong way and ends in failure (§4.1).

4.2 VLN in Continuous Environments

Datasets. R2R-CE [31] and RxR-CE [32] are more practical yet challenging, where the agent can navigate to any unobstructed point instead of teleporting between fixed nodes of a pre-defined navigation graph in R2R. R2R-CE comprises a total of $5,611$ shortest-path trajectories with significantly longer time horizons than R2R. RxR-CE presents more instructions with longer annotated paths, including multilingual descriptions. In addition, the agents in RxR-CE are forbidden to slide along obstacles. Both of are R2R-CE and RxR-CE driven by Habitat Simulator [96].

Evaluation Metrics. The metrics of R2R-CE [31] are similar to R2R [1]. For RxR-CE [32], Normalize Dynamic Time Wrapping (NDTW) and NDTW penalized by SR (SDTW) are further used to evaluate the fidelity between the predicted and annotated paths.

Performance on R2R-CE. As shown in Table 3, ENP provides a considerable performance gain against VLN⟳BERT [4], *e.g.*, $\mathbf{1}\%$ SR and $\mathbf{1}\%$ SPL on *val unseen*. Concretely, it outperforms the top-leading ETPNav [29] by $\mathbf{1}\%$ and $\mathbf{1}\%$ in terms of OSR and SR on *test unseen*, respectively. These quantitative results substantiate our motivation of empowering VLN agents with ENP.

Table 3: Quantitative comparison results on R2R-CE [31] (§4.2). '−': unavailable statistics.

Models	R2R-CE *val seen*					R2R-CE *val unseen*					R2R-CE *test unseen*				
	TL↓	NE↓	OSR↑	SR↑	SPL↑	TL↓	NE↓	OSR↑	SR↑	SPL↑	TL↓	NE↓	OSR↑	SR↑	SPL↑
VLN-CE [31] [ECCV2020]	9.26	7.12	46	37	35	8.64	7.37	40	32	30	8.85	7.91	36	28	25
CMTP [37] [CVPR2021]	–	7.10	56	36	31	–	7.90	38	26	23	–	–	–	–	–
HPN [97] [ICCV2021]	8.54	5.48	53	46	43	7.62	6.31	40	36	34	8.02	6.65	37	32	30
SASRA [98] [ICPR2022]	8.89	7.71	–	36	34	7.89	8.32	–	24	22	–	–	–	–	–
CM2 [99] [CVPR2022]	12.05	6.10	51	43	35	11.54	7.02	42	34	28	13.90	7.70	39	31	24
WSMGMap [60] [NeurIPS2022]	10.12	5.65	52	47	43	10.00	6.28	48	39	34	12.30	7.11	45	35	28
Sim2Sim [100] [ECCV2022]	11.18	4.67	61	52	44	10.69	6.07	52	43	36	11.43	6.17	52	44	37
BEVBert [61] [ICCV2023]	–	–	–	–	–	–	4.57	67	59	50	–	4.70	67	59	50
DREAM [71] [ICCV2023]	11.60	4.09	66	59	48	11.30	5.53	59	49	44	11.80	5.48	57	49	44
HNR [101] [CVPR2024]	11.79	3.67	76	69	61	12.64	4.42	67	61	51	13.03	4.81	67	58	50
VLN⟳BERT [4] [CVPR2021]	12.50	5.02	59	50	44	12.23	5.74	53	44	39	13.51	5.89	51	42	36
ENP–VLN⟳BERT	**12.08**	**4.89**	**60**	**51**	44	12.37	5.81	53	**45**	**40**	**13.21**	**5.68**	**52**	**44**	36
ETPNav [29] [TPAMI2024]	11.78	3.95	72	66	59	11.99	4.71	65	57	49	12.87	5.12	63	55	48
ENP–ETPNav	11.82	**3.90**	**73**	**68**	59	**11.45**	**4.69**	65	**58**	**50**	**12.71**	**5.08**	**64**	**56**	48

Performance on RxR-CE. Table 4 reports the results with Marky-mT5 instructions [56] on RxR-CE. ENP achieves competitive results on longer trajectory navigation with multilingual instructions, *e.g.*, $\mathbf{55.27}\%$ *vs.* 54.79% SR and $\mathbf{62.97}\%$ *vs.* 61.90% NDTW for ETPNav [29] on *val unseen*. We attribute this to the global expert policy matching, which promotes the path fidelity. This indicates ENP enables the agent to make long-term plans, resulting in better NDTW and SDTW.

Table 4: Quantitative comparison results on RxR-CE [32] (§4.2). '−': unavailable statistics.

Models	RxR-CE *val seen*					RxR-CE *val unseen*				
	NE↓	SR↑	SPL↑	NDTW↑	SDTW↑	NE↓	SR↑	SPL↑	NDTW↑	SDTW↑
CWP-CMA [64] [CVPR2022]	–	–	–	–	–	8.76	26.59	22.16	47.05	–
VLN⟳BERT [4] [CVPR2021]	–	–	–	–	–	8.98	27.08	22.65	46.71	–
Reborn [29] [CVPR2022]	5.69	52.43	45.46	66.27	44.47	5.98	48.60	42.05	63.35	41.82
HNR [101] [CVPR2024]	4.85	63.72	53.17	68.81	52.78	5.51	56.39	46.73	63.56	47.24
ETPNav [29] [TPAMI2024]	5.03	61.46	50.83	66.41	51.28	5.64	54.79	44.89	61.90	45.33
ENP–ETPNav	5.10	**62.01**	**51.18**	**67.22**	**51.90**	**5.51**	**55.27**	**45.11**	**62.97**	**45.83**

4.3 Diagnostic Experiment

Joint Distribution Matching. We first investigate the joint distribution matching loss $\mathcal{L}_\rho$ (Eq. 3) in ENP, which is factored as the discriminative action loss $\mathcal{L}_\pi$ (Eq. 1) and the marginal state distribution matching loss $\mathcal{L}_\mathcal{S}$ (Eq. 7). Both of $\mathcal{L}_\pi$ and $\mathcal{L}_\mathcal{S}$ are online optimized iteratively. A clear performance drop is observed in Table 5, *e.g.*, SR from $\mathbf{74}\%$ to 72% for DUET [6] on R2R, as only optimizing the conditional action distribution with $\mathcal{L}_\pi$ will degenerate into BC (or DAGGER) with cross-entropy loss. This reveals the appealing effectiveness of the joint distribution matching strategy in ENP.

Table 5: Ablation study of objective loss (§4.3).

Models	Objective	R2R *val*		R2R-CE *val*	
		SR↑	SPL↑	SR↑	SPL↑
ENP–VLN⟳BERT [4]	$\mathcal{L}_\pi$	61	55	44	39
	$\mathcal{L}_\pi + \mathcal{L}_\mathcal{S}$	**65**	**57**	**45**	**40**
ENP–DUET&ETPNav [6, 29]	$\mathcal{L}_\pi$	72	60	57	49
	$\mathcal{L}_\pi + \mathcal{L}_\mathcal{S}$	**74**	60	**58**	**50**

Step Size and Guassian Noise in SGLD. In ENP, SGLD [23] is used to draw samples from Boltzmann distribution (Eq. 9) and further optimize $\mathcal{L}_\mathcal{S}$. Following recent EBM training, we relax the restriction on the relation between the step size ϵ and Gaussian noise ξ, *i.e.*, $\text{Var}(\xi) = \epsilon$. Table 6 shows the results with different step sizes (when $I = 15$), and changing ϵ in a small range has little effects on the performance. In addition, different noise variances $\text{Var}(\xi)$ are also explored. Our ENP is insensitive to parameter changes. We find that $\epsilon = 1.5$ and $\xi \sim \mathcal{N}(0, 0.01)$ for the transition kernel of SGLD works well across a variety of datasets and frameworks in VLN.

Table 6: Ablation study of step size and noise (§4.3).

Models	#SGLD	R2R *val*		R2R-CE *val*	
		SR↑	SPL↑	SR↑	SPL↑
ENP–VLN⟳BERT [4]	$\epsilon = 1,\ \text{Var}(\xi) = 0.01$	64	57	45	39
	$\epsilon = 1.5,\ \text{Var}(\xi) = 0.01$	**65**	**57**	**45**	**40**
	$\epsilon = 2,\ \text{Var}(\xi) = 0.01$	64	56	44	40
	$\epsilon = 1.5,\ \text{Var}(\xi) = 0.1$	63	55	41	38

Number of SGLD Loop per Training Step. For the SGLD sampler, we study the influence of the number of inner loop iterations per training step. In Table 7, we can find that $I = 15$ iterations is enough to sample the recurrent states for EnvDrop [3] and VLN⟳BERT [4]. Slightly more iterations are required for the models [6, 29] with additional topological graph memory due to the global action space. This can be mitigated by pre-storing the samples in the Marginal State Memory $\mathcal{M}$ during pre-training. As these frameworks [6, 29] employ pre-training and finetuning paradigm for VLN, we design a different number of iterations, *i.e.*, $I_{\text{pre}} = 20$ and $I_{\text{ft}} = 5$. It is worth noting that the state memory $\mathcal{M}_{\text{pre}}$ can acquire the samples while performing single-step action prediction in the VLN pre-training. Thus, the SGLD iterations in finetuning based on $\mathcal{M}_{\text{ft}} \leftarrow \mathcal{M}_{\text{pre}}$ can be further reduced.

Table 7: Ablation study of SGLD iterations (§4.3).

Models	#Loop	R2R *val*		R2R-CE *val*	
		SR↑	SPL↑	SR↑	SPL↑
ENP–VLN⟳BERT [4]	$I = 5$	64	56	44	39
	$I = 15$	**65**	**57**	**45**	**40**
	$I = 20$	65	57	45	40
ENP–DUET [6] &ENP–ETPNav [29]	$I_{\text{pre}} = 10,\ I_{\text{ft}} = 5$	74	58	56	49
	$I_{\text{pre}} = 20,\ I_{\text{ft}} = 5$	**74**	**60**	**58**	**50**
	$I_{\text{pre}} = 20,\ I_{\text{ft}} = 10$	73	60	58	49

ENP *vs.* AIRL. In Table 8, we compare ENP against AIRL [27], which is a representative work of adversarial imitation learning. The objective function for ENP is equivalent to the forward KL divergence minimization, where AIRL [27] makes use of the backward KL divergence. By modifying the original policy learning in VLN⟳BERT [4] (a mixture of RL and IL) as an adversarial reward learning formulation, AIRL achieves the comparable performance. ENP yields better scores on R2R-CE with $\mathbf{45}\%$ SR and $\mathbf{40}\%$ SPL. Meanwhile, ENP avoids the potential influence from the adversarial training.

Table 8: ENP *vs.* AIRL (§4.3).

Models	Policy	R2R *val*		R2R-CE *val*	
		SR↑	SPL↑	SR↑	SPL↑
VLN⟳BERT [4]	AIRL[27]	63	57	43	39
	ENP	**65**	57	**45**	**40**

Runtime Analysis. The additional computational complexity of ENP is from the iterations of SGLD inner loop. Towards this, we make some specific designs, such as Marginal State Memory $\mathcal{M}$ and pre-storing in pre-training. During inference, there is no additional computational consumption in executing the strategy compared to existing VLN agents.

5 Conclusion

In this paper, we propose ENP, an Energy-based Navigation Policy learning framework. By explicitly modeling the joint state-action distribution using an energy-based model, ENP shows promise to solve the intrinsic limitations of supervised behavioural cloning. Through maximizing the likelihood of the action prediction and modeling the dynamics of the navigation states jointly, ENP learns to minimize the divergence distance from the expert policy in a collaborative manner. Experimental results on gold-standard VLN datasets in both discrete (R2R and REVERIE) and continuous environments (R2R-CE and RxR-CE) demonstrate the superiority against existing methods. In the future, we will explore more tasks and frameworks with the energy-based policy.

Acknowledgment. This work was supported by the National Science and Technology Major Project (No. 2023ZD0121300), the National Natural Science Foundation of China (No. 62372405), the Fundamental Research Funds for the Central Universities 226-2024-00058, the National Key Laboratory of Human-Machine Hybrid Augmented Intelligence, Xi'an Jiaotong University (No. HMHAI-202403), and CIPSC-SMP-Zhipu Large Model Cross-Disciplinary Fund.

References

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

[2] Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

[3] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL*, 2019.

[4] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln bert: A recurrent vision-and-language bert for navigation. In *CVPR*, 2021.

[5] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. In *NeurIPS*, 2021.

[6] 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 *CVPR*, 2022.

[7] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[8] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019.

[9] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

[10] Tian Xu, Ziniu Li, and Yang Yu. Error bounds of imitating policies and environments. In *NeurIPS*, 2020.

[11] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, 2015.

[12] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[13] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *ECCV*, 2018.

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

[15] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.

[16] Hu Wang, Qi Wu, and Chunhua Shen. Soft expert reward learning for vision-and-language navigation. In *ECCV*, 2020.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[18] Christopher M Bishop. Pattern recognition and machine learning. *Springer*, 2:645–678, 2006.

[19] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[20] Fredrik K Gustafsson, Martin Danelljan, Radu Timofte, and Thomas B Schön. How to train your energy-based model for regression. In *BMVC*, 2020.

[21] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *ICML*, 2008.

[22] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *NeurIPS*, 2019.

[23] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[24] Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. In *AAMAS*, 2021.

[25] Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Strictly batch imitation learning by energy-based distribution matching. In *NeurIPS*, 2020.

[26] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

[27] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *ICLR*, 2018.

[28] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *CoRL*, 2020.

[29] Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE TPAMI*, 2024.

[30] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *CVPR*, 2020.

[31] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *ECCV*, 2020.

[32] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *EMNLP*, 2020.

[33] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018.

[34] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.

[35] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[36] Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. In *NeurIPS*, 2020.

[37] Kevin Chen, Junshen K Chen, Jo Chuang, Marynel Vázquez, and Silvio Savarese. Topological planning with transformers for vision-and-language navigation. In *CVPR*, 2021.

[38] Hanqing Wang, Wenguan Wang, Wei Liang, Caiming Xiong, and Jianbing Shen. Structured scene memory for vision-language navigation. In *CVPR*, 2021.

[39] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In *ICCV*, 2021.

[40] Aming Wu, Rui Liu, Yahong Han, Linchao Zhu, and Yi Yang. Vector-decomposed disentanglement for domain-invariant object detection. In *ICCV*, 2021.

[41] Yi Yang, Yueting Zhuang, and Yunhe Pan. Multiple knowledge representation for big data artificial intelligence: framework, applications, and case studies. *Frontiers of Information Technology & Electronic Engineering*, 22(12):1551–1558, 2021.

[42] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019.

[43] Hanqing Wang, Wei Liang, Jianbing Shen, Luc Van Gool, and Wenguan Wang. Counterfactual cycle-consistent learning for instruction following and generation in vision-language navigation. In *CVPR*, 2022.

[44] Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. Learning to follow and generate instructions for language-capable navigation. *IEEE Transactions on PAMI*, 2023.

[45] Xianghao Kong, Jinyu Chen, Wenguan Wang, Hang Su, Xiaolin Hu, Yi Yang, and Si Liu. Controllable navigation instruction generation with chain of thought prompting. In *ECCV*, 2024.

[46] Sheng Fan, Rui Liu, Wenguan Wang, and Yi Yang. Navigation instruction generation with bev perception and large language models. In *ECCV*, 2024.

[47] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *CVPR*, 2019.

[48] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020.

[49] Hanqing Wang, Wei Liang, Luc V Gool, and Wenguan Wang. Towards versatile embodied navigation. In *NeurIPS*, 2022.

[50] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *ECCV*, 2020.

[51] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *CVPR*, 2020.

[52] Kunyang Lin, Peihao Chen, Diwei Huang, Thomas H Li, Mingkui Tan, and Chuang Gan. Learning vision-and-language navigation from youtube videos. In *ICCV*, 2023.

[53] Chong Liu, Fengda Zhu, Xiaojun Chang, Xiaodan Liang, Zongyuan Ge, and Yi-Dong Shen. Vision-language navigation with random environmental mixup. In *ICCV*, 2021.

[54] Jialu Li, Hao Tan, and Mohit Bansal. Envedit: Environment editing for vision-and-language navigation. In *CVPR*, 2022.

[55] 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 *ICCV*, 2023.

[56] Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. Less is more: Generating grounded navigation instructions from landmarks. In *CVPR*, 2022.

[57] Yibo Cui, Liang Xie, Yakun Zhang, Meishan Zhang, Ye Yan, and Erwei Yin. Grounded entity-landmark adaptive pre-training for vision-and-language navigation. In *ICCV*, 2023.

[58] Hanqing Wang, Wenguan Wang, Wei Liang, Steven CH Hoi, Jianbing Shen, and Luc Van Gool. Active perception for visual-language navigation. *IJCV*, 131(3):607–625, 2023.

[59] Jinyu Chen, Wenguan Wang, Si Liu, Hongsheng Li, and Yi Yang. Omnidirectional information gathering for knowledge transfer-based audio-visual navigation. In *ICCV*, 2023.

[60] Peihao Chen, Dongyu Ji, Kunyang Lin, Runhao Zeng, Thomas Li, Mingkui Tan, and Chuang Gan. Weakly-supervised multi-granularity map learning for vision-and-language navigation. In *NeurIPS*, 2022.

[61] 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 *ICCV*, 2023.

[62] Rui Liu, Xiaohan Wang, Wenguan Wang, and Yi Yang. Bird's-eye-view scene graph for vision-language navigation. In *ICCV*, 2023.

[63] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Gridmm: Grid memory map for vision-and-language navigation. In *ICCV*, 2023.

[64] Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *CVPR*, 2022.

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

[66] Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. Towards learning a generalist model for embodied navigation. In *CVPR*, 2024.

[67] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1), 1991.

[68] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[69] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *CoRL*, 2022.

[70] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[71] Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. Dreamwalker: Mental planning for continuous vision-language navigation. In *ICCV*, 2023.

[72] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *ICLR*, 2019.

[73] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NeurIPS*, 2015.

[74] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *ICML*, 2020.

[75] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[76] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[77] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

[78] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

[79] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *ICLR*, 2019.

[80] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[81] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.

[82] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *ICML*, 2019.

[83] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[84] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[85] Giorgio Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

[86] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

[87] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017.

[88] Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. Active visual information gathering for vision-language navigation. In *ECCV*, 2020.

[89] Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. Hop: history-and-order aware pre-training for vision-and-language navigation. In *CVPR*, 2022.

[90] Yusheng Zhao, Jinyu Chen, Chen Gao, Wenguan Wang, Lirong Yang, Haibing Ren, Huaxia Xia, and Si Liu. Target-driven structured transformer planner for vision-language navigation. In *ACM MM*, 2022.

[91] Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. Lana: A language-capable navigator for instruction following and generation. In *CVPR*, 2023.

[92] Rui Liu, Wenguan Wang, and Yi Yang. Volumetric environment representation for vision-language navigation. In *CVPR*, 2024.

[93] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[94] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *ICCV*, 2021.

[95] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2019.

[96] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019.

[97] Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *ICCV*, 2021.

[98] Muhammad Zubair Irshad, Niluthpol Chowdhury Mithun, Zachary Seymour, Han-Pang Chiu, Supun Samarasekera, and Rakesh Kumar. Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. In *ICPR*, 2022.

[99] Georgios Georgakis, Karl Schmeckpeper, Karan Wanchoo, Soham Dan, Eleni Miltsakaki, Dan Roth, and Kostas Daniilidis. Cross-modal map learning for vision and language navigation. In *CVPR*, 2022.

[100] Jacob Krantz and Stefan Lee. Sim-2-sim transfer for vision-and-language navigation in continuous environments. In *ECCV*, 2022.

[101] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, Junjie Hu, Ming Jiang, and Shuqiang Jiang. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In *CVPR*, 2024.

[102] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

[103] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *EMNLP*, 2019.

[104] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018.

[105] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123:32–73, 2017.

[106] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[107] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[108] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[109] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, 2020.

[110] Weijian Luo, Boya Zhang, and Zhihua Zhang. Entropy-based training methods for scalable neural implicit samplers. In *NeurIPS*, 2023.

A Appendix

A.1 Model Details

EnvDrop [3] adopts an encoder-decoder model. The encoder is a bidirectional LSTM-RNN with an embedding layer. The decoder is an attentive LSTM-RNN. EnvDrop uses the 2048-dimensional image features from ResNet-152 pretrained on ImageNet. The model is first trained with supervised learning via the mixture of imitation and reinforcement learning. Then, it is finetuned by back translation with environmental dropout. The word embedding size is 256 and the dimension of the action embedding is 64. The size of the LSTM units is set to 512 while 256 is set for the bidirectional LSTM. RMSprop [102] is employed to optimize the loss with a learning rate 1×10^{-4} and the batch size is set as 64.

VLN↻BERT [4] is built upon LXMERT-like transformer [103]. It is trained directly on the mixture of the original training data and the augmented data from PREVALENT [51] for R2R [1]. AdamW optimizer [104] is used and the batch size is set to 16. For REVERIE [30], the image features are extracted by ResNet-152 pre-trained on Places365 [48], and the object features are encoded by Faster-RCNN pre-trained on Visual Genome [105].The batch size is set as 8.

VLN↻BERT for CE [64] proposes a waypoint predictor to generate a set of candidate waypoints during navigation. The training objective is the cross-entropy loss between the ground-truth and action predictions. It is minimized by AdamW optimizer [104] with the learning rate 1×10^{-5} during training. The decay ratio is set as 0.50 for R2R-CE [31] and 0.75 for RxR-CE [32].

DUET [6] uses a dual-scale graph transformer initialized from LXMERT [103]. The hidden layer size is 768. The image features are extracted by ViT-B/16 [106] pretrained on ImageNet [107]. On REVERIE [30], DUET is first pre-trained with the batch size of 32. Some auxiliary tasks are employed, such as single-step action prediction, object grounding, masked language modeling, and masked region classification [5]. Then it is finetuned with the batch size of 8.

ETPNav [29] contains a topological mapping module and a waypoint prediction module [64]. It is pre-trained using proxy tasks following existing transformer-based VLN models [4, 6] with a batch size of 64 and a learning rate of 5×10^{-5}. During finetuning, the AdamW optimizer [104] is utilized. The batch size is set as 16 with a learning rate of 1×10^{-5}. The decay ratio is 0.75.

A.2 Comparison Results across Different Path Lengths

In Fig. 4, ENP demonstrates notable performance improvements over BC [6] across both *seen* and *unseen* splits of R2R [1]. When the length of ground-truth paths exceeds 16 meters, our ENP outperforms BC by a significant margin in both the *val seen* and *val unseen* splits. In addition, the performance gains are relatively more significant in the *unseen* split compared to the *seen* split. For example, ENP achieves approximately **5**% improvement of SR on the *val seen* and **6**% improvement on the *val unseen* for navigation paths longer than 16 meters.

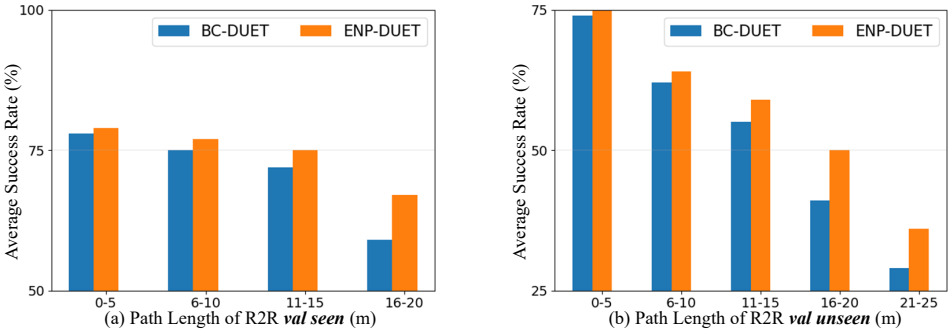

Figure 4: The average success rate of ENP and BC [6] across different trajectory lengths.

A.3 Discussion

Terms of use, Privacy, and License. R2R [1], REVERIE [30], R2R-CE [31], and RxR-CE [32] datasets utilized in our work contain a large number of indoor photos from Matterport3D [87],

which are freely available for non-commercial research purposes. No identifiable individuals are present in any of the photographs. All experiments are conducted on the Matterport3D and Habitat Simulators [1, 96], which are released under the MIT license.

Limitations. Existing studies in VLN mainly focus on improving the success rate of navigation, with limited attention given to collision avoidance and navigation safety. In addition, current agents are typically designed for static environments within simulators. To mitigate the risk of collisions, it is essential for robots to operate under specific security constraints. Therefore, further advancements are necessary to ensure safe deployment in real-world dynamic environments.

Future Direction. i) We explore ENP on several representative VLN architectures [3, 4, 6, 29], and the generalization of ENP to other architectures [63, 61] and more tasks [108] is not clear. We will verify it in future work. ii) Efficiently sampling from un-normalized distributions for high-dimensional data is a fundamental challenge in machine learning. Utilizing MCMC-free approaches [75, 109] and neural implicit samplers [110] may offer valuable insights to enhance our approach. One limitation of ENP is that the optimization through the SGLD inner loop at each training iteration (Algorithm 1), which would reduce the training efficiency (about 9% delay for 5 SGLD iterations). In practice, ENP balances performance and training time with a limited number of iterations. iii) In the future, agents should be able to actively acquire the skills by watching videos similar to humans [52], thereby reducing high training data (expert demonstrations) requirements.

Broader Impacts. We introduce an energy-based policy learning for VLN. Equipped with ENP, existing agents are able to perform comprehensive decision-making, showcasing promising improvements across various VLN benchmarks. In real-world deployment, the agent holds potential to significantly benefit society by providing assistance to individuals with specific needs, *e.g.*, blind and disabled, enabling them to accomplish daily tasks more independently. In addition, we encourage more efforts devoted to policy learning for future research in the community.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See §1. We include the paper's contributions and scope in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Appendix A.3, we discuss the limitations of the work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: In §3.2, we provide the details of our method.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Algorithm 1 and §3.4, we provide the implementation details of our method.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: As we promised, the code will be released upon the publication of our paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In §4 and Appendix A.1, we provide the training and test details for the experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Table 1, 2, 3, and 4, we report error bars on VLN benchmarks.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In §3.4, we provide the details of the computer resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We acknowledge the NeurIPS Code of Ethics and obey them in our paper.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In Appendix A.3, we discuss the broader impacts of our work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper is based on publicly available datasets and models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: In Appendix A.3, we provide the details of the license and terms of use. We also use existing models as components and cite appropriately.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.