Hijacking Vision-and-Language Navigation Agents with Adversarial Environmental Attacks

Zijiao Yang, Xiangxi Shi, Eric Slyman, Stefan Lee
Oregon State University
{yangziji,shixia,slymane,leestef}@oregonstate.edu

2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) | 979-8-3315-1083-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/WACV61041.2025.00594

Abstract

Assistive embodied agents that can be instructed in natural language to perform tasks in open-world environments have the potential to significantly impact labor tasks like manufacturing or in-home care – benefiting the lives of those who come to depend on them. In this work, we consider how this benefit might be hijacked by local modifications in the appearance of the agent's operating environment. Specifically, we take the popular Vision-and-Language Navigation (VLN) task as a representative setting and develop a whitebox adversarial attack that optimizes a 3D attack object*'s appearance to induce desired behaviors in pretrained VLN agents that observe it in the environment. We demonstrate that the proposed attack can cause VLN agents to ignore their instructions and execute alternative actions after encountering the attack object – even for instructions and agent paths* not *considered when optimizing the attack. For these novel settings, we find our attacks can induce early-termination behaviors or divert an agent along an attacker-defined multi-step trajectory. Under both conditions, environmental attacks significantly reduce agent capabilities to successfully follow user instructions.*

1. Introduction

Developing assistants that follow natural language instructions to execute complex tasks in open-world environments is a compelling task that enables applications like household robotics [1] and automated warehouse management [37]. To study computational mechanisms underpinning the multimodal reasoning skills necessary to support such applications, the community has rallied around benchmark tasks like Vision-and-Language Navigation (VLN) that require agents to traverse an environment by grounding a natural language navigation instruction to visual observations [3]. A key focus in VLN is that agents must generalize to new natural language instructions in novel environments. However, deploying a learning-based system in uncontrolled environments opens the possibility of malicious actors that modify the surroundings to intentionally impact the system's performance, reliability, or efficiency [5].

One class of such environmental modification is adversarial attacks that optimize for patterns or objects that, when photographed, induce errors in learning-based systems – e.g., adversarial stickers [13], t-shirts [40], or 3D objects [4]. Adversarial attacks directly optimizing digital inputs are more common and have been applied to image classifiers [9], natural language processing models [29, 42], sequential decision making policies [15], and even multimodal vision language models (VLMs) [24, 46].

In this work, we study adversarial attacks for the VLN task – a multimodal sequential decision making task requiring vision-and-language reasoning. We design a whitebox environmental adversarial attack paradigm that leverages differential rendering in 3D mesh environments to optimize the appearance of 3D *attack objects* in VLN environments to induce specific behavior from a trained VLN agent. These attacks are optimized to either (1) force the VLN agent to immediately terminate the episode by issuing a stop command even if far from its instructed goal or (2) follow an attacker-defined trajectory to a location not specified by the original instruction. We study the effectiveness of these attacks in response to varying natural language instruction and navigation history.

To assess our method and explore model vulnerabilities in VLN, we study its effects on a representative VLN agent [8] in the popular R2R [3] and RxR [19] datasets. For both stopping and trajectory-following attacks, we find the presence of attacked objects results in significant disruption of VLN performance – e.g., a reduction in success rate from 82.42% to 53.85% on a trajectory-following attacked subset of R2R. Moreover, we find varying degrees of success for inducing adversary-desired behaviors for novel instruction-trajectory pairs – e.g., stop attacks causing episode termination in 75% of cases and trajectory-following attacks causing agents to arrive at the new destination 20% of the time in R2R. For instruction-trajectory pairs used during attack optimization, these rates increase substantially. Finally, we assess the influence of method hyperparameters and factors

2642-9381/25/$31.00 ©2025 IEEE
DOI 10.1109/WACV61041.2025.00594

like object size and category on attack success.

Contributions. Summarizing this work, we:

- Develop an adversarial attack framework for controlling the trajectories of VLN agents that uses differentiable rendering to modify the appearance of 3D scene objects.
- Demonstrate that the resulting attacks are effective at altering the behavior and performance of a representative VLN model [8] when generalizing to new instruction-trajectory instances in the attacked scene on representative VLN datasets.
- Present statistical analysis to better understand what factors influence the success of these attacks.

2. Related Work

Vision-and-Language Navigation (VLN). VLN requires an embodied agent to traverse an environment by grounding natural language instructions to visual observations [3]. In this work, we study HAMT [8] as a representative VLN agent with a history mechanism that preserves information from past visual inputs for future decision-making. We refer readers to a recent survey for additional background [14].

Generalizable Adversarial Attacks. Adversarial Machine Learning (AML) [5] is critical for exposing vulnerabilities in ML systems by identifying inputs that induce failure states or controllable behaviors. Adversarial attacks, a common form of AML, craft inputs that provoke specific responses from models. While early attacks focus on RGB image perturbations that cause misclassifications in vision systems [28,38], the scope of these attacks has broadened to include other domains like natural language processing [29,42], control policies [15], and more [9].

Prior work demonstrates the generalization of adversarial attacks beyond pathological fine-grained RGB perturbations, highlighting vulnerabilities in safety-critical systems. [28] demonstrate attacks that generalize across images, while [15] show that similar attacks can extend and generalize to neural network policies. [24] study transferable attacks for vision-and-language models such that perturbing a single image would mislead all predictions given different textual prompts. Notably, [17,20] study robustness of adversarial images to digital-to-physical transformations like printing, showing that adversarial attacks are generalizable to the real world. Our work extends the understanding of adversarial attacks by focusing on multimodal embodied agents in the VLN setting, showing that localized consistent appearance modifications on 3D objects can significantly disrupt agent behavior, hijacking agent's sequential decision making process. Unlike previous studies demonstrating generalization across images, models, and policies, our attacks must generalize across unseen natural language instructions and navigation trajectories.

Adversarial Attacks on Embodied Agents. While early work attacking reinforcement learning (RL) policies consider perturbing an agent's visual inputs when playing Atari games [15,22,43], recent work considers the more complex task of attacking embodied agents in 3D environments [33]. Two works in particular are most similar to ours. [45] consider attacking VLN agents, but do so in the setting of federated learning by leveraging a malicious client to output poisoned data causing the global agent to disregard the natural language instruction and instead follow a series of visual triggers. [23] study 3D adversarial attacks on EQA [10]. Unlike our work, they consider synthetic (rather than natural) QA-like language instructions, attack the question-answering output of the agent rather than its navigation trajectory, and operate in unrealistic synthetic scenes.

Environment Attacks. Adversarial attacks often target the environment where a digital image is captured rather than altering the image pixels directly. [6] develop "adversarial patches" that induce specific misclassifications when placed on images. Similarly, [13] create robust physical sticker-based attacks that mimic graffiti and remain effective under varied real-world conditions. Several works consider camouflage textures that can be applied to real objects. [40] design adversarial shirt patterns to deceive object detection models. Similar methods are applied to varied objects [12] and cars [39] to fool image classifiers and object detectors, respectively. [21] use physical semi-translucent stickers on camera lenses, attacking the camera apparatus instead of objects in the environment. Several recent works leverage differentiable rendering to perturb 3D scenes. [4, 30] modify the texture of 3D objects to induce robust 2D adversarial image attacks, while [26, 44] focus on attacks against models which operate in 3D environments themselves.

In contrast, our work studies environmental adversarial attacks on VLN agents. This approach involves permuting the RGB texture of an object in the scene during navigation, directing the agent to follow a different path from the one specified in the natural language instruction. Unlike prior work primarily targeting static image classification and detection, this work focuses on dynamic navigation tasks and the interplay between visual and linguistic inputs.

3. Adversarial Attacks on VLN Agents

Before describing our adversarial environmental attack framework, we establish notation for the task setting.

VLN Episodes and Agents. A single Vision-and-Language Navigation (VLN) episode can be defined as a tuple ($\mathcal{E}$, I, τ) consisting of an environment $\mathcal{E}$ in which the agent is situated, a natural language navigation instruction I, and a trajectory τ through $\mathcal{E}$ corresponding to accurately following I. In standard discrete VLN settings [3], the environment $\mathcal{E}$ is represented as an undirected navigation graph with nodes $\mathcal{V}_{\mathcal{E}}$ corresponding to a discrete set of environment locations the agent can visit and edges $L_{\mathcal{E}}$ between nodes which the

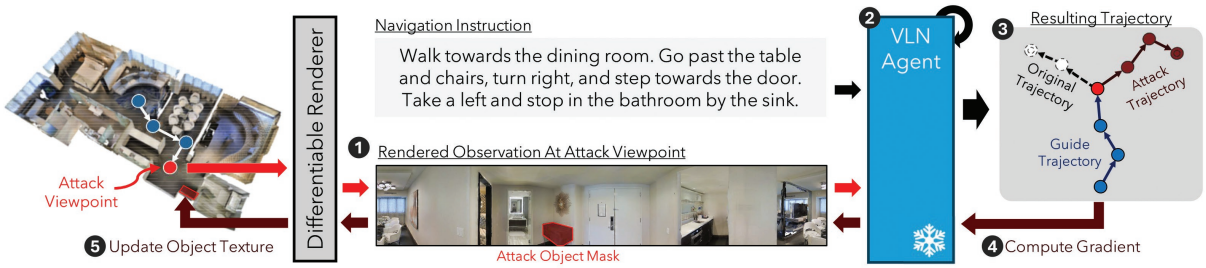

Figure 1. We directly optimize the appearance of an in-environment object to control the trajectory of a trained VLN agent using a differentiable renderer. ❶ Adversarial observations are rendered at an attack viewpoint containing the attack object. ❷ The VLN agent takes this observation as input and ❸ we supervise the agent's trajectory from this point to match a predetermined attack trajectory. ❹ We compute loss gradients with respect to the object texture and use them to ❺ update the object's appearance in the 3D mesh.

agent is allowed to traverse. A trajectory is then a sequence $\tau = [v_0, v_1, ..., v_n]$ where all $v_t \in \mathcal{V}_\mathcal{E}$ and $(v_{t-1}, v_t) \in L_\mathcal{E}$. Further, each location v_i is associated with a panoramic observation of $\mathcal{E}$ taken at v_i which we denote as o_{v_i}.

Successfully following an instruction in a new environment requires making a sequence of accurate predictions about where to navigate next. Given an instruction I, a partial trajectory $[v_0, ..., v_t]$ up to the current step t, and the corresponding observations $[o_{v_0}, ..., o_{v_t}]$, a VLN agent π_θ produces a distribution over candidate navigation actions $\mathcal{A}(v_t) = N(v_t) \cup \{\texttt{STOP}\}$, where $N(v_t)$ is the set of nodes connected to v_t and the STOP action terminates the episode. The agent's action is then selected as

$$\underset{a \in \mathcal{A}(v_t)}{\text{argmax}} \ \pi_\theta\big(\, a \mid I,\ [v_0, ..., v_t]\ ,\ [o_{v_0}, ..., o_{v_t}]\,\big).$$

The trajectory terminates when the agent issues the STOP command and agent performance is based on the similarity of the resulting trajectory to the ground truth τ.

Adversarial Attacks on VLN Agents. As the VLN task is a proxy for real language-guided embodied systems, we consider how an adversary might gain influence over a real VLN agent's behavior when it is deployed in uncontrolled environments. For instance, how a delivery robot might be diverted to a desired location or a surveillance drone be made to end its rounds prematurely – regardless of the original instructions they were given by their users. In both, an adversary would like to be able to dictate the sequence of actions an agent takes after encountering the attack.

In this embodied setting, the agent's visual observations would be acquired using an onboard camera. Consequentially, standard adversarial attacks that directly modify image pixels [28] would only be possible if an adversary has gained software access to the agent. Alternatively, transparent film-based attacks that are applied over a physical camera lens have been developed [21] that might be applied, but assume physical access to the agent a priori.

Rather than assuming access to individual agents, we consider a class of object-based environmental attacks where an adversary has control over the appearance of an *attack object* in the 3D environment. Further, we do not assume the adversary knows the particular instruction given to the agent or the partial trajectory it has taken before encountering the attack object. That is to say, a successful attack must generalize to new language instructions and observation histories. While many goals for the adversary are possible, we consider *trajectory* attacks where an attacked agent is made to follow an adversary-defined trajectory after encountering the attack object. This requires the attack to influence a whole sequence of decisions, even when the attack object itself may no longer be visible.

Formalizing this slightly, we seek a procedure to adjust an object's appearance in order to maximize the expected likelihood of a given VLN agent following an attack trajectory after entering the attack object's location. To promote generalization of the attack, this expectation is over pairs of natural instructions and corresponding agent trajectories prior to encountering the attack object. We examine this in the whitebox setting where the adversary also has access to the VLN model structure and weights being deployed as well as the target environment.

3.1. Attack Methodology

Our attack framework is demonstrated in Fig. 1 for a single trajectory. Given a textured 3D mesh representation of the environment, we select a viewpoint and object to attack. ❶ For an agent approaching this *attack viewpoint*, we use a differentiable rendering pipeline to produce the corresponding panoramic observation. ❷ Given the history of its trajectory so far, the instruction, and the rendered adversarial observation, the VLN agent produces a distribution over next actions. ❸ We supervise the agent to assign high probability to the attack trajectory and ❹ backpropagate gradients of this loss all the way to the object texture in the 3D mesh. ❺ With this gradient, we update the object's texture while respecting attack magnitude constraints. To provide generalization, we perform this optimization over a set of episodes that pass through the attack viewpoint.

Attack Viewpoint and Trajectory. We denote the viewpoint where the attack object is located as the attack viewpoint v_{ATK}. After entering the attack viewpoint, a successfully attacked VLN agent would then follow the attack trajectory $\hat{\tau} = [\hat{v}_0, \hat{v}_1, ..., \hat{v}_L, \texttt{STOP}]$ terminating at the target location by issuing the `STOP` command. We refer to the agent's trajectory up to encountering v_{ATK} as the *guide trajectory* and denote it as $\tau_{\leq v_{\text{ATK}}}$. We denote the corresponding observation sequence in a similar fashion as $o_{\leq v_{\text{ATK}}}$.

Observation Rendering. Given a 3D model of the environment $\mathcal{E}$ consisting of a 3D mesh denoted $M_{\mathcal{E}}$ and corresponding texture atlas image $T_{\mathcal{E}}$, we apply the PYTORCH3D [35] differentiable renderer to produce observations through which gradients can affect object textures. In contrast to image-level attacks, our framework takes into account the view consistency of the attacked object by directly optimizing an explicit 3D representation.

As differentiable rendering has a high memory and computational cost, we limit its use to a subset of viewpoints along the attack trajectory. Specifically, we render observations for the attack viewpoint and the following next two steps. This allows the object's appearance to not just affect the decision where the object is most visible, but also future decisions more effectively [34, 36] while ensuring consistent appearance across these different views. To further reduce costs, we only render panorama sub-images that contain the attack object. When attacking an object, we produce an object mask D such that D_i is 1 if the i^{th} face in the mesh is used to render the object and 0 otherwise.

Optimization. As we would like our attacks to generalize to new trajectories and instructions, we consider a set of VLN episodes $\{(\mathcal{E}, I^{(i)}, \tau^{(i)})\}_{i=1}^{n}$ for training the attack such that each ground truth trajectory τ_i includes v_{ATK}. To maximize the expected likelihood described above, we consider the cross entropy loss for the agent following the attack trajectory after arriving at the attack viewpoint v_{ATK}, denoting it as $\ell_{\text{CE}}^{(i)}(T_{\mathcal{E}})$ equal to

$$-\sum_{t=0}^{L+1} \ln \pi_\theta \Big(\hat{v}_t \mid I^{(i)}, [\underbrace{\tau^{(i)}_{\leq v_{\text{ATK}}}}_{\text{Guide Traj.}}, \underbrace{\hat{\tau}_{<t}}_{\text{Attack Traj.}}], [\underbrace{o^{(i)}_{\leq v_{\text{ATK}}}, o_{\hat{v}_{<t}}}_{\text{Panoramic Observations}}] \Big)$$

where the summation traverses each step in the attack trajectory and $\hat{\tau}_{<t}$ and $o_{\hat{v}_{<t}}$ denote the attack trajectory and observations up to time step t. We denote the aggregation of this loss across a batch of attack training episodes as $\mathcal{L}_{\text{CE}}$.

To update object appearance, we take the gradient of $\mathcal{L}_{\text{CE}}$ with respect to the object texture, computing $D \odot \nabla_{T_{\mathcal{E}}} \mathcal{L}_{\text{CE}}$ where $\odot$ denotes a gradient masking for non-object mesh faces and $T_{\mathcal{E}}$ is the texture atlas image. With this, we use the ADAM optimizer [18] to update $T_{\mathcal{E}}$. In line with existing Projected Gradient Descent-based attack strategies [25], we limit the L_∞ norm of the changes from the original to some attack magnitude constant ϵ and clamp color values to [0,1]. This iterates over multiple batches and update steps. The final output of this optimization process is an altered 3D environment model $\mathcal{E}'$ in which the appearance of the attack object has been modified. We can then evaluate the attack on new instruction-trajectory pairs.

3.2. Generating and Evaluating Attacks

To evaluate our attack strategy, we consider VLN episodes from the Room-2-Room (R2R) dataset [3] consisting of nearly 22,000 episodes across 90 indoor environments from the MP3D dataset [7]. These episodes are divided into `R2R-train`, `R2R-val-seen`, `R2R-val-unseen`, and `R2R-test`. For our purposes, we focus on `R2R-train` and `R2R-val-seen` which share the same set of environments but different trajectory-instruction pairs. We consider episodes in `R2R-val-seen` to select attack viewpoints and objects that have sufficient supporting episodes in `R2R-train` to train an attack. Note that while VLN agents have been trained on `R2R-train`, `R2R-val-seen` episodes are novel instruction-trajectory pairs to the model and attacks.

We also consider RxR [19] to additionally verify effectiveness of our method. RxR is a multilingual and larger VLN dataset consists of more than 16,000 paths and 120,000 fine-grained grounded instructions. RxR situated in the same set of house environment as R2R and has longer instruction and longer path on average: 78 *vs*. 29 words, 8 vs 5 edges. Similar as in R2R, we consider `RxR-train-guide` and `RxR-val-seen-guide` that share the same set of environments, and use `RxR-train-guide` to train attack while using `RxR-val-seen-guide` to select attack viewpoints and objects.

We use `train-data` and `val-seen-data` for ease of description on data processing over R2R and RxR.

Candidate Attack Objects. To select attack objects, we consider object visibility annotations from [32] associated with 3D object segmentations from MP3D [7] – examining objects from the most common categories of `chair`, `cabinet`, `table`, `plant`, `sofa`, and `TV monitor`. For each viewpoint in R2R/RxR, we compile a list of these objects visible from that location. Further, we discard any object that does not make up at least 40% of the pixels in at least one of the 36 panoramic sub-images for this viewpoint. To compute this visibility score, we render the object mask at each of the sub-images and compute the pixel ratio.

Constructing Attack Instances. For each episode $(\mathcal{E}, I, \tau)$ in `val-seen-data`, we check if there exists any viewpoints $v \in \tau$ with a valid candidate attack object X and at least 5 episodes in `train-data` that pass through v. Further, we exclude any training episode if its guide trajectory

matches that of the val-seen episode to ensure our evaluation represents a novel observation history. If multiple viewpoints meet this criteria, we select the one corresponding to the object with the largest visibility score and denote it as the attack viewpoint $v_{\texttt{ATK}}$ for this instance. Finally, we generate an attack trajectory by selecting a viewpoint that is at least three meters away from the original final viewpoint v_K in τ and find a shortest path $\tau_{\texttt{ATK}} = v_{\texttt{ATK}}, ..., v_K$ to it from the attack viewpoint. A resulting attack instance j then consists of an environment $\mathcal{E}_j$, instruction I_j, ground truth trajectory τ_j, attack object X_j, attack viewpoint $v_{\texttt{ATK}_j}$, attack trajectory $\tau_{\texttt{ATK}_j}$, and set of training instances $\mathcal{D}_j = \{\mathcal{E}_j, I^{(i)}, \tau^{(i)}\}_{i=1}^{N} \subset \texttt{train-data}$. When optimizing our attack, we split $\mathcal{D}_j$ 80/20 into $\mathcal{D}_j^{\texttt{Train}}$ and $\mathcal{D}_j^{\texttt{Val}}$ to produce a validation set. In total, this process yields a test set with 273 attack instances and 254 attack instances for R2R and RxR respectively. Notice we sample a subset of attack instances constructed from RxR to accommodate compute resource and time limit.

Evaluation. For each attack instance, we run our optimization procedure described in the preceding subsection, yielding an attacked environment $\mathcal{E}'$. During evaluation, we render all observations using $\mathcal{E}'$ rather than just near the attack viewpoint. To evaluate the effect on the attack instance, we force the VLN agent through the guide trajectory until reaching $v_{\texttt{ATK}}$ and then allow it to autoregessively navigate the environment until it terminates by issuing the `STOP` action. We consider two evaluation strategies. In the first, we measure how the attack affects the agent's ability to reach the goal specified by the instruction. For the second, we measure whether the attacked environment actually causes the agent to follow the corresponding attack trajectory. For both, we use standard VLN path comparison and success metrics described in the following section – comparing the agents trajectory after $v_{\texttt{ATK}}$ with either the rest of the original trajectory τ or the attack trajectory $\tau_{\texttt{ATK}}$.

4. Experimental Results

Metrics. For evaluating attack effectiveness, we focus on two primary VLN metrics [16] – success rate (SR) and normalized dynamic time warping (nDTW). Given a predicted trajectory and a reference trajectory, success requires the final viewpoint in the predicted trajectory be no further than 3m from that of the reference. Meanwhile, nDTW is a path similarity metric which is maximized if paths match exactly.

VLN Model. We conduct our attack experiment using the HAMT [8] model. HAMT is a widely-used VLN model in the literature and representative of common architectures. Visual inputs are encoded using a ViT [11] image encoder. Trajectory history is encoded in a hierarchical fashion then jointly combined with encodings of instruction text and the current observation to predict the next action. It is trained in multiple stages with auxiliary losses followed by finetuning on VLN tasks with supervised and reinforcement learning. We use the best performing model weights provided by authors and keep weights frozen at all times.

	R2R		RxR	
Metrics	Real	Rendered	Real	Rendered
SR ↑	75.61	69.64	59.37	53.00
nDTW ↑	78.33	72.16	65.41	59.13

Table 1. *Top:* Real scene images (top row) are higher fidelity than rendered images (bottom row). *Bottom:* Despite this, we find VLN performance for HAMT on the `val-seen` split for R2R and RxR only drop by marginal amounts when replacing real images with rendered ones.

Implementation Details. For attack optimization, we set ADAM's learning rate to 1e-2 and exponential moving average rates to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For attack magnitude, we set $\epsilon = 0.3$. To further reduce computational and memory costs, we decimate the environment meshes from MP3D and use lower-res texture atlas images. Optimization runs for 300 batches (iterations) with batch size 16, and saves checkpoints every 30 iterations. For RxR dataset, we increase to 600 batches to help convergence. After training, the checkpoint with highest nDTW with respect to attack trajectory $v_{\texttt{ATK}_j}$ on the validation set $\mathcal{D}_j$ of each attack instance j is retained. Recall that each attack instance is trained independently. Each attack training was performed on an NVIDIA A40 GPU and took on average 40 minutes for R2R and 90 minutes for RxR per attack.

Quantifying Domain Gap from Rendering. Our approach relies on rendering images from reconstructed MP3D environments; however, these reconstructions can be noisy and we further reduce their fidelity to accommodate available compute. As a result, the rendered images are lower quality than those with which VLN agents are typically trained – especially for scenes containing thin or reflective structures. If this alone significantly affects VLN agent performance, we need to account for it when evaluating the impact of rendered attacks. To evaluate this gap, we compare performance of HAMT on the original R2R/RxR panoramas and those rendered from unattacked 3D reconstructions of the same environments. As shown in Tab. 1, the experiment using rendered imagery does demonstrate some reduced performance – approximately 6-7 points in success

R2R	Train (⊂ R2R train)			Validation (⊂ R2R train)			Test (⊂ R2R val-seen)		
Attacked:	✓	✗	Δ	✓	✗	Δ	✓	✗	Δ
Stop % ↑	98.28	1.80	96.48	80.70	1.97	78.73	75.98	0.98	75.00

RxR	Train (⊂ RxR train)			Validation (⊂ RxR train)			Test (⊂ RxR val-seen)		
Attacked:	✓	✗	Δ	✓	✗	Δ	✓	✗	Δ
Stop % ↑	46.05	13.68	32.37	33.04	13.91	19.13	25.20	14.17	11.03

Table 2. *Single-step Attacks for R2R (left) and RxR (right).* We compare stop rate for the HAMT agent operating in attacked (✓) and unaltered (✗) environments as well as the difference (Δ). Across all settings, we see our attacks substantially increase the likelihood of a VLN agent terminating prematurely – especially for R2R where 75% of novel trajectory-instruction pairs are ended by the agent.

or path following metrics – but the model retains the bulk of its capabilities. To be considered successful, our attacks will need to illicit larger reductions than these.

Evaluation. We evaluate in the following conditions:
– `Train`. While our primary focus is on attacks that generalize to new trajectory-instruction pairs, evaluating attack success for the examples on which they are optimized is a common setting for adversarial examples [9]. We evaluate each attack instance on the examples used to optimize the attack – i.e., each episode in $\mathcal{D}_j^{\texttt{Train}}$ is evaluated in the attacked environment $\mathcal{E}'_j$. Metrics are aggregated across all training episodes from all attack instances. Note that these instances belong to the `R2R-train` / `RxR-train-guide` set and thus have been seen by the HAMT model during training. As such, they require *no* generalization from the attack *or* the VLN model.
– `Validation`. Analogously, episodes from each validation set $\mathcal{D}_j^{\texttt{Val}}$ are evaluated on their corresponding attacked environment $\mathcal{E}'_j$ and metrics are aggregated across all attack instances. As these episodes are also from `R2R-train` / `RxR-train-guide`, they are only novel to the attack and not the model. Note that checkpoint selection uses these episodes so attack performance is likely overestimated here.
– `Test`. Finally, we evaluate each attack instance I_j, τ_j taken from `R2R-val-seen` / `RxR-val-seen-guide` on the corresponding attacked environment $\mathcal{E}'_j$. These trajectory-instruction pairs have not been seen when optimizing the attack or training the VLN model. This introduces potential out-of-distribution signals in the text and history features, leading to unpredictable behavior when these inputs are processed by the model in the attacked environment $\mathcal{E}'_j$. Therefore, we consider this to be the most challenging evaluation settings.

4.1. Single-step Attacks

Before reporting results for our trajectory-level attacks, we examine a special case of our attack framework – producing attacks that cause agents to stop immediately. By setting the attack trajectory to be empty ($\tau_{\texttt{ATK}} = \{\}$) agents are supervised to issue the `STOP` action and terminate the episode at the attack viewpoint. All other details of our methodology remain as described. This setting is analogous to environmental attacks against image classification systems; however, we consider a VLN model which is also conditioned on language and navigation history. Additional filtering is applied during attack generation to ignore episodes which already terminate at $v_{\texttt{ATK}}$. We show stop rates in Tab. 2 for the HAMT agent operating in attacked (✓) and unaltered (✗) environments. Note that unaltered (✗) environments are rendered with PYTORCH3D as in Tab. 1.

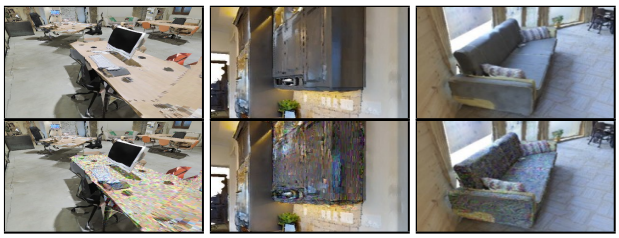

Figure 2. Example original and attacked objects for a desk (left), cabinet (middle), and sofa (right) from trajectory-level attacks.

For R2R (left), we find this simple attack is highly effective and would be extremely disruptive. The agent stops immediately in 75.98% of `Test` instances – a 76x increase from the unaltered environments. For `Train` and `Validation` samples, stop rates approach 98.28% and 80.70% respectively. If deployable, these attacks could make no-go zones for VLN agents where the stop action is taken regardless of the user's original instruction.

For RxR (right), we also see increased stop rates; however, the magnitude of these effects are significantly smaller than in R2R. We believe this difference is due to RxR having substantially longer path lengths – roughly ~6.1 vs. ~10.5 average steps – inducing strong biases in the trained VLN models. Our attack trajectories require an agent to stop between step 4 and 6 on average, requiring attacks to override a strong prior from model training.

4.2. Trajectory-level Attacks

We show trajectory-level attack results in Tab. 3. Overall, we find strong evidence that the attacks can induce multi-step trajectory following behaviors.

For R2R (left), the HAMT agent operating attacked environments (✓) is roughly 5x more likely to arrive at the terminal location of the attack trajectory than in unaltered environments (✗) – success rate climbing from 3.66% to 21.61% in `Test`. Likewise, nDTW trends that suggest the agent is more likely to adhere to the attack trajectory. We emphasize that examples in `Test` do not contribute to at-

R2R	Train (⊂ R2R train)			Validation (⊂ R2R train)			Test (⊂ R2R val-seen)		
Attacked:	✓	✗	Δ	✓	✗	Δ	✓	✗	Δ
SR ↑	30.26	1.15	29.11	22.01	1.04	20.97	21.61	3.66	17.95
OSR ↑	43.00	6.17	36.83	34.32	6.93	27.39	38.46	11.36	27.10
nDTW ↑	53.38	29.11	24.27	47.48	29.34	18.14	46.65	30.60	16.05

RxR	Train (⊂ RxR train)			Validation (⊂ RxR train)			Test (⊂ RxR val-seen)		
Attacked:	✓	✗	Δ	✓	✗	Δ	✓	✗	Δ
SR ↑	25.26	5.30	19.95	16.23	2.61	13.62	20.47	5.51	14.96
OSR ↑	69.74	23.45	46.29	59.13	18.55	40.58	78.74	35.43	43.31
nDTW ↑	30.64	18.26	12.38	25.56	15.05	10.51	21.25	14.44	6.80

Table 3. *Trajectory-level Attacks for R2R (left) and RxR (right).* We report results for trajectory-level attacks that cause agents to follow specific attack trajectories. We compare success rate (SR), oracle success rate (OSR), and normalized Dynamic Time Warping (nDTW) for the HAMT agent operating in attacked (✓) and unaltered (✗) environments as well as the difference (Δ). Higher values imply better adherence to the attack trajectory. Across all settings, our attacks increase the agent's likelihood of following the attack trajectory.

R2R	Train (⊂ R2R train)			Validation (⊂ R2R train)			Test (⊂ R2R val-seen)		
Attacked:	✓	✗	Δ	✓	✗	Δ	✓	✗	Δ
SR ↑	46.11	91.35	-45.24	55.46	91.33	-35.87	53.85	82.42	-28.57
nDTW ↑	47.44	87.73	-40.29	58.21	87.06	-28.85	53.68	81.18	-27.50

RxR	Train (⊂ R2R train)			Validation (⊂ R2R train)			Test (⊂ R2R val-seen)		
Attacked:	✓	✗	Δ	✓	✗	Δ	✓	✗	Δ
SR ↑	24.47	63.17	-38.70	29.86	62.90	-33.04	25.59	59.06	-33.47
nDTW ↑	27.17	66.94	-39.77	32.48	66.91	-34.43	29.28	62.67	-33.39

Table 4. *Impact on VLN Performance for R2R (left) and RxR (right).* We report the impact on VLN performance for the HAMT agent encountering trajectory-level attacks in attacked (✓) and unaltered (✗) environments as well as the difference (Δ). Reductions in performance resulting in negative differences indicate our attacks are successful at disrupting the VLN agent's ability to follow given instructions.

tack optimization or the training of the VLN agent. Each of these successes represent an attack generalizing to new natural language and observation sequences by correctly inducing the desired multi-step behavior in the multimodal sequential decision making policy of HAMT. Example attack objects are shown in Fig. 2.

For RxR (right), we find attacked agents are roughly 3x more likely to arrive at the terminal location of the attack trajectories, with `Test` success increasing by 15 points compared to unaltered environments. As in the previous experiments, absolute change in SR and nDTW are lower in RxR than R2R. To examine this, we also include oracle success rate (OSR) which checks if the agent arrives at the target location at any point in its trajectory – essentially ignoring the `STOP` component of navigation. With OSR, we see a significant effect for RxR. Similar as Single-step case, we find the guiding trajectory followed by the attack trajectory averages 2-3 steps shorter than standard RxR training instances. This again suggests that while the attack is successful at redirecting the agent along the attack trajectory, the strong model bias for certain path lengths may be difficult to overcome. Further, RxR's broader set of instructions and paths may result in a more robust agent in terms adhering to the instruction – as discussed in its proposal [19].

Effect on VLN Performance. Beyond inducing the desired trajectory, our trajectory-level attacks also significantly disrupt the agent's ability to follow its original instructions. In Tab. 4, we show standard VLN performance for HAMT in attacked (✓) and unaltered (✗) environments. We find our attacks cut success rate by nearly 35% and 57% in `Test` instances for R2R (left) and RxR (right) respectively with similar drops in other metrics. As in our prior results, these effects are stronger for samples from `Train` and `Validation` that are involved in the training of the attack. Note that performance in unaltered environments for `Test` does not match that reported in Tab. 1 because only generated attack instances (Sec. 3.2) are considered here rather than all of `R2R-val-seen` / `RxR-val-seen-guide`.

4.3. Method Ablations

To better understand the impact of hyperparameter choices on our attacks, we perform several ablations for trajectory-level attacks in R2R. Further details can be found in the supplementary materials. For space, we present results on `Validation` and `Test` instances only.

Steps Rendered. For computational reasons, we only render observations during training for three viewpoints (see Sec. 3)– the attack viewpoint and proceeding two viewpoints where the attack object is likely still visible. In Tab. 5 (a), we vary the number of rendered observations during training between 1 (only the attack viewpoint) and 3. As in all our experiments, all observations are rendered during evaluation. We find rendering more steps improves attack performance significantly. For example, we observe a nearly 4x increase in `Test` success rate when rendering 3 steps instead of just 1. Rendering multiple observations better matches the evaluation setting and optimizes the attack to affect the agent across different viewing perspectives.

Attack Budget. Varying ϵ allows for differing maximum object texture perturbations with larger ϵ denotes a stronger but more distorted and identifiable attack [9]. We use ϵ's of 0.1, 0.3, and 0.5 in Tab. 5 (b). As expected, we find larger attack magnitudes lead to better attack performance.

Instructions. Each trajectory in R2R is paired with 3 in-

R2R	Validation (⊂ R2R train)		Test (⊂ R2R val-seen)	
Steps	SR ↑	nDTW ↑	SR ↑	nDTW ↑
1	6.86	41.18	4.08	33.17
2	17.97	47.57	12.25	42.76
3*	**21.57**	**49.09**	**19.05**	**46.49**

(a) Steps Rendered

R2R	Validation (⊂ R2R train)		Test (⊂ R2R val-seen)	
ϵ	SR ↑	nDTW ↑	SR ↑	nDTW ↑
0.1	17.97	46.83	16.33	42.57
0.3*	21.57	49.09	19.05	46.49
0.5	**21.90**	**49.68**	**22.45**	**46.55**

(b) Attack Budget

R2R	Validation (⊂ R2R train)		Test (⊂ R2R val-seen)	
Instr.	SR ↑	nDTW ↑	SR ↑	nDTW ↑
1	18.63	46.73	12.93	40.84
2	**21.90**	**49.37**	17.69	42.95
3*	21.57	49.09	**19.05**	**46.49**

(c) Instructions

R2R	Validation (⊂ R2R train)		Test (⊂ R2R val-seen)	
Iters	SR ↑	nDTW ↑	SR ↑	nDTW ↑
300*	21.57	49.09	19.05	46.49
600	30.39	55.43	26.53	50.26
900	**31.70**	**56.31**	**26.53**	**51.71**

(d) Training Iterations

Table 5. *Ablations of Attack Hyperparameters in R2R.* We use * to denote baseline hyperparameters. During each ablation study, we vary only one hyperparameter at a time while use baseline hyperparameters for others.

structions. In Tab. 5 (c), we vary how many of these are included during attack optimization per trajectory. Note we keep the training iterations the same across all runs. We find using more instructions during attack optimization generally leads to better generalization to the novel instructions encountered in `Validation` and `Test` instances.

Training Iterations. Keeping batch size fixed at 16 and checkpoint selection on `Validation`, we vary the number of training iterations in Tab. 5 (d). We find longer training times lead to substantial improvements in attack generalization – e.g., increasing `Test` success rates by 7.5 points when training time is tripled from 300 to 900 iterations.

4.4. Factors that Influence Attack Success

We investigate the effects from different factors on trajectory-level attack effectiveness on R2R `Test`. To facilitate our analysis, we frame experiments as paired-measurements on individual attack instances with nDTW measured pre- and post-attack. We assess the statistical significance of selected factors affecting nDTW by fitting linear mixed effect regression (`lmer`) models and evaluate significance with ANOVA and post-hoc t-tests.

Object Size. We first examine effects of the attack object's size across navigation viewpoints. For each sub-image within each panoramic viewpoint, we calculate the attacked object's size as the percent of the sub-image covered by the object. For example, an object which cannot be seen will have a coverage of 0% and one which fills the entire view 100%. We find strong evidence (p=0.005) of a difference between pre/post-attack groups attributed to the size of the object in the selected view at the attack viewpoint. The coefficient of the effect indicates (p=0.001) a +0.59 point increase to attack nDTW per percent coverage. When considering the average coverage over all selected views in the attack trajectory, we again observe a significant difference between groups (p<0.001) and find the strength of the effect to increase to +1.50 (p<0.001) points per average percent covered. We note that the maximum observed coverage for a single selected view is 47% and we would not suspect a linear trend in nDTW up to 100% coverage.

Object Category. We analyze if any particular object category is easier or harder to attack and find strong evidence (p=0.002) for a difference between groups based on object category. Specifically, we find that the categories `sofa` (+15.15 nDTW, p=0.028) and `table` (+8.65 nDTW, p=0.042) are particularly susceptible to attack. Congruent with the size analysis above, we hypothesize that this is due at least somewhat to these objects being larger.

Training Episode Diversity. We find some evidence of a difference between groups (p=0.081) attributed to heading entropy when entering the attack viewpoint during training. We calculate heading entropy over the normalized rates of discrete entrance angles for training episodes entering the attack viewpoint. We find a positive correlation between entropy and nDTW (+8.41 nDTW/unit, p=0.036) indicating that greater trajectory diversity may improve performance.

Object in Instruction. Finally, we test for an effect from the attacked object being mentioned in the instruction. We use Porter stemming [31] to reduce instruction words to their root form and then check for overlap in the WordNet [27] synsets of those words and the attacked objects class. We do not find any significant (p=0.467) effects. This observation is potentially supported by prior work that suggests object grounding is weak in VLN agents [41].

5. Discussion

In this work, we developed an adversarial attack framework that uses differentiable rendering to modify the appearance of 3D scene objects and demonstrated that it can allow an adversary to significantly impact the performance of VLN agents and even exert some level of control over their navigation trajectories regardless of the original user's instructions. At current, our work does not address the deployability of these attacks in the real world; however, prior work in adversarial machine learning [17, 20] and sim2real transfer of VLN agents [2] suggest it may be possible in future work. For now, our attacks also require substantial computation to produce due to the demands of differentiable rendering and assume a whitebox setting for the adversary; however, developing a blackbox analog is plausible at an increased computational expense. Together, this suggests that adequate adversarial defenses should be established prior to deployment of instructable embodied agents.

References

[1] Anas Abou Allaban, Maozhen Wang, and Taşkın Padır. A systematic review of robotics research in support of in-home care for older adults. *Information*, 11(2):75, 2020. 1

[2] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *Conference on Robot Learning*, pages 671–681. PMLR, 2021. 8

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *CVPR*, 2018. 1, 2, 4

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *ICML*, 2017. 1, 2

[5] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, page 16–25, New York, NY, USA, 2006. Association for Computing Machinery. 1, 2

[6] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv*, 2017. 2

[7] 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 *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE, 2017. 4

[8] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. In *NeurIPS*, 2021. 1, 2, 5

[9] Joana C Costa, Tiago Roxo, Hugo Proença, and Pedro RM Inácio. How deep learning sees the world: A survey on adversarial attacks & defenses. *IEEE Access*, 2024. 1, 2, 6, 7

[10] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[11] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 5

[12] Ranjie Duan, Xingjun Ma, Yisen Wang, James Bailey, A Kai Qin, and Yun Yang. Adversarial camouflage: Hiding physical-world attacks with natural styles. In *CVPR*, 2020. 2

[13] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *CVPR*, 2018. 1, 2

[14] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. *ACL*, 2022. 2

[15] Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and P. Abbeel. Adversarial attacks on neural network policies. *ICLR (Workshop)*, 2017. 1, 2

[16] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *arXiv preprint arXiv:1907.05446*, 2019. 5

[17] Steve T.K. Jan, Joseph Messou, Yen-Chen Lin, Jia-Bin Huang, and Gang Wang. Connecting the digital and physical world: Improving the robustness of adversarial attacks. In *AAAI*, 2019. 2, 8

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[19] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *arXiv preprint arXiv:2010.07954*, 2020. 1, 4, 7

[20] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR*, 2017. 2, 8

[21] Juncheng Li, Frank R. Schmidt, and J. Zico Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. *ICML*, 2019. 2, 3

[22] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *IJCAI*, 2017. 2

[23] Aishan Liu, Tairan Huang, Xianglong Liu, Yitao Xu, Yuqing Ma, Xinyun Chen, Stephen J. Maybank, and Dacheng Tao. Spatiotemporal attacks for embodied agents. *ECCV*, 2020. 2

[24] Haochen Luo, Jindong Gu, Fengyuan Liu, and Philip Torr. An image is worth 1000 lies: Adversarial transferability across prompts on vision-language models, 2024. 1, 2

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018. 4

[26] Yibo Miao, Yinpeng Dong, Junyi Zhu, and Xiao-Shan Gao. Isometric 3d adversarial examples in the physical world. *NeurIPS*, 2022. 2

[27] George A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. 8

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CVPR*, 2016. 2, 3

[29] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020. 1, 2

[30] Camilo Pestana, Naveed Akhtar, Nazanin Rahnavard, Mubarak Shah, and Ajmal S. Mian. Transferable 3d adversarial textures using end-to-end optimization. *WACV*, 2022. 2

[31] M.F. Porter. An algorithm for suffix stripping. In *Program*, pages 130–137, 1980. 8

[32] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020. 4

[33] You Qiaoben, Chengyang Ying, Xinning Zhou, Hang Su, Jun Zhu, and Bo Zhang. Consistent attack: Universal adversarial perturbation on embodied vision navigation. *Pattern Recognition Letters*, 168, 2022. 2

[34] Shilin Qiu, Qihe Liu, Shijie Zhou, and Wen Huang. Adversarial attack and defense technologies in natural language processing: A survey. *Neurocomputing*, 492:278–307, 2022. 4

[35] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 4

[36] Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. Token-modification adversarial attacks for natural language processing: A survey. *AI Communications*, (Preprint):1–22, 2021. 4

[37] Enoch Oluwademilade Sodiya, Uchenna Joseph Umoga, Olukunle Oladipupo Amoo, and Akoh Atadoga. Ai-driven warehouse automation: A comprehensive review of systems. *GSC Advanced Research and Reviews*, 18(2):272–282, 2024. 1

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 2

[39] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. Dual attention suppression attack: Generate adversarial camouflage in physical world. *CVPR*, 2021. 2

[40] Zuxuan Wu, Ser-Nam Lim, Larry Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. *ECCV*, 2020. 1, 2

[41] Zijiao Yang, Arjun Majumdar, and Stefan Lee. Behavioral analysis of vision-and-language navigation agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2023. 8

[42] Jin Yong Yoo and Yanjun Qi. Towards improving adversarial training of nlp models. *arXiv preprint arXiv:2109.00544*, 2021. 1, 2

[43] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan D. Liu, Duane S. Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *NeurIPS*, 2020. 2

[44] Jinlai Zhang, Lyujie Chen, Binbin Liu, Bojun Ouyang, Qizhi Xie, Jihong Zhu, and Yanmei Meng. 3d adversarial attacks beyond point cloud. *Information Sciences*, 2021. 2

[45] Yunchao Zhang, Zonglin Di, Kaiwen Zhou, Cihang Xie, and Xin Eric Wang. Navigation as the attacker wishes? towards building byzantine-robust embodied agents under federated learning. *arXiv*, 2022. 2

[46] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Chongxuan Li, Ngai-Man Cheung, and Min Lin. On evaluating adversarial robustness of large vision-language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 1