ABSTRACT
With the growing capabilities of intelligent systems, the integration of robots in our everyday life is increasing. However, when interacting in such complex human environments, the occasional failure of robotic systems is inevitable. The field of explainable AI has sought to make complex-decision making systems more interpretable but most existing techniques target domain experts. On the contrary, in many failure cases, robots will require recovery assistance from non-expert users. In this work, we introduce a new type of explanation, $\mathcal{E}_{err}$, that explains the cause of an unexpected failure during an agent’s plan execution to non-experts. In order for $\mathcal{E}_{err}$ to be meaningful, we investigate what types of information within a set of hand-scripted explanations are most helpful to non-experts for failure and solution identification. Additionally, we investigate how such explanations can be autonomously generated, extending an existing encoder-decoder model, and generalized across environments. We investigate such questions in the context of a robot performing a pick-and-place manipulation task in the home environment. Our results show that explanations capturing the context of a failure and history of past actions, are the most effective for failure and solution identification among non-experts. Furthermore, through a second user evaluation, we verify that our model-generated explanations can generalize to an unseen office environment, and are just as effective as the hand-scripted explanations.

CCS CONCEPTS
• Human-centered computing → Interaction paradigms.

KEYWORDS
Explainable AI, Fault Recovery

1 INTRODUCTION
In homes, hospitals, and manufacturing plants, robots are increasingly being tested for deployment alongside non-roboticists to perform complex tasks, such as folding laundry [49], delivering laboratory specimens [9, 25], and moving inventory goods [22, 33]. When operating in such complex human environments, occasional robot failures are inevitable. When failures occur, human assistance is often required to correct the problem [7], and co-located users—homeowners, medical lab technicians, and warehouse workers—will be first on the scene. We classify such users as everyday users, or non-experts, because of their lack of formal training in machine learning, AI, or robotics.

In order for everyday users to be able to assist in robot failure recovery, they will need to understand why a failure has occurred. For example, a homeowner waiting for a robot to bring them coffee may need to determine why the robot suddenly stopped in the middle of the kitchen, or a production line worker may need to determine why a packing robot suddenly stopped picking up items.

The field of Explainable AI (XAI) has sought to address the challenge of understanding "black-box" systems through the development of interpretable machine learning (ML) algorithms that can explain their decision making to users [2, 21]. Furthermore, a subfield of XAI, Explainable Planning (XAIP), has focused on generating explanations specifically for sequential-decision making tasks, including explaining an agent's chosen plan and explaining unsolvable plans to end-users [12]. Such techniques hold great promise for the development of more transparent robotic systems but they do not incorporate explanations for unexpected failures during a plan execution. Additionally, the majority of existing XAI techniques are designed for technical domain experts who understand AI or ML at its core [2, 40, 42, 48, 51]. While expert understanding is crucial, such XAI methods are not suitable for the vast majority of end-users, who are non-experts [18, 20, 26].

In this work, we seek to make robotic systems more transparent to their users by leveraging techniques from explainable AI, while also extending the capabilities of XAI systems toward greater transparency for non-expert users. Specifically, our work addresses fault recovery cases in which the robot’s task execution is halted due to an error. We investigate whether providing explanations can not only help non-expert users understand the system’s point of failure, but also help them determine an appropriate solution required to resume normal operation of the task. In some cases (e.g., complex hardware failures), the user might not have the knowledge to fix the point of failure regardless of the provided error explanation. In this work, we focus on failures that we expect to be within the user’s understanding (e.g., object is too far away), and we address this
question in the context of pick-and-place manipulation tasks in the home environment. Our work makes the following contributions:

- **Formalization of error explanations**: Providing justifications for points of failures that occur unexpectedly amid an agent’s plan execution has not previously been studied within the XAIP community. We extend an existing set of explanations available in the XAI and XAIP community, introducing error explanations designed to explain failures that occur during the execution of a task.

- **Explanation content**: We empirically evaluate what information an error explanation should contain to aid non-experts in understanding the cause of failure and to select a recovery strategy. We show that explanations that include both (i) history of recently accomplished actions, and (ii) contextual reasoning about the environment, are the most effective in enabling users to identify the cause of and solution to a failure.

- **Explanation generation**: We present an automated technique for generating natural language error explanations that rationalize encountered failures in a manner that is understandable by non-experts. Specifically, we extend an encoder-decoder model for autonomously generating natural language explanations introduced by [20] to generate context-based history explanations within a continuous state-space.

- **Validation with non-expert users**: We demonstrate that explanations generated by the encoder-decoder model can generalize to an unseen environment and are as effective as hand-scripted, context-based history explanations.

We validate our approach through two user studies and computational model analysis. In the first study, we examine what information an error explanation should contain by evaluating how the content of an explanation affects user performance in identifying and assisting with a robot error (Sec. 4). From these results, we identify an explanation type that leads to the highest performance, and then contribute a computational model to automatically generate such explanations from robot states (Sec. 5). In our second study, we demonstrate that our automated explanations are as effective as hand-scripted explanations in guiding non-experts to identify the cause of a failure and its potential solution.

### 2 RELATED WORKS

The XAI community has primarily focused on developing interpretability methodologies for understanding the inner workings of black-box models [2, 40]. Many of these approaches have focused on model-agnostic implementations, designed to increase expert understanding of deep learning outputs [2, 38, 39]. Additionally, most XAI approaches [40, 42, 48, 51] have primarily focused on understanding classification-based tasks. However, classification tasks do not capture the complexity of sequential decision-making in an agent, such as a robot, may perform while having long-term interactions with users [12].

To address the need for interpretable explanations in sequential decision making tasks, the XAIP community has focused on explaining an agent’s plans to end-users. A recent survey paper highlights some of the key components of plan explanations studied by the community [12]: (1) contrastive question-answering, (2) explaining unsolvable plans, and (3) providing explicable justifications for a chosen plan. In the realm of contrastive question-answering, Krarup et al. provide a framework to transfer domain-independent user questions into constraints that can be added to a planning model [32], while Hoffmann et al. utilize common properties within a set of correct plans as an explanation for unmet properties in incorrect plans [24]. In order to explain unsolvable plans, Sreedharan et al. abstract the unsolvable plan into a simpler example through which explanations can be formulated [43]. Finally, in order to provide explicable justifications for a plan, Zhang et al. use conditional random fields (CRFs) to model human explanations of existing agent plans, and use such a human “mental model” as a constraint for generating explicable plans [52].

Techniques for plan repair enable a task plan to be adapted to overcome an error [14, 23]. Methods in this domain reuse the failing plan and search the plan space to find local deviations that allow continued execution [15], or transform the plan to adapt it to the situation [27, 34]. Such repairs are often found and executed autonomously, with no human intervention. Recently, works have considered interactive plan repair with a human-in-the-loop. Boteanu et al. showed a proof-of-concept model in which a human approved repair action is proposed by a common-sense reasoning framework [10]. However, this work was limited to errors involving missing items, and did not focus on explaining errors to users. Meanwhile, Knepper et al. investigated the grounding of natural language requests to best garner help from non-expert humans [31]. They found that the requests were successful when they were targeted, e.g., helped listeners disambiguate between multiple objects, and told them what to do. The authors developed a system to generate such requests. We build upon these findings to investigate the characteristics of natural language error explanations that allow non-experts to help a robot; unlike [31], we do not assume the robot is aware of a correct recovery and able to direct the user on the recovery process.

Plan repair for failures that occur during execution require a fault diagnosis [23] and there is a large body of ongoing work in robotics focused on fault diagnosis techniques [29]. These works use a range of methods, including unsatisfied Preconditions [10, 15, 31], first-order logic inference [50], case-based reasoning [23, 36], sensor signal processing [1, 17, 28], Bayes nets and Dynamic Bayesian Networks [8, 30], Hidden Markov Models (HMMs) [47], particle filters [44, 53], and neural networks [35, 37] to diagnose failures. Depending on the context of the work, the diagnosis either identifies what is wrong with the robot—e.g., object not visible [31] or sonar is blind [17]—or why it is wrong—e.g., there was a collision with the environment [35]. However, the prior work aims to use the diagnoses for autonomous robot recovery from failure or to facilitate debugging by experts. The problem of generating natural language explanations from a fault diagnosis to allow non-experts to help a robot recover remains largely unexplored.

In efforts to provide explanations to non-experts on infeasible agent behaviors, prior work has presented a linear temporal logic
We define the problem of providing explanations for non-expert users. (a) Data collection in failure simulations and the extraction of the agent’s state space. (b) Study of hand-scripted explanations with varying information types (Sec. 4). (c) Autonomous generated $\mathcal{E}_{err}$ explanations using an encoder-decoder model (Sec. 5).

(LTL) framework to explain actions unsatisfiable by a robot [39]. The explanations focus on what actions are unattainable by the robot, but do not include the underlying reasons for why they may be unattainable. Similarly, an algorithm called HIGHLIGHTS uses visual animations to summarize agent capabilities—what an agent can achieve—to non-expert users, based on the features dictating an agent’s reward function [3]. In our work, we show that non-experts need to be told why an action is unattainable in order to help an agent recover; explaining what is unattainable is insufficient.

Finally, prior work in XAI has found that natural language explanations can provide “justification” and are “understandable” by non-experts [20]. The study, conducted in the discrete domain of Frogger, used sequence-to-sequence learning to treat explanation generation as a neural translation problem, where an agent’s internal states are translated into natural language, with impressive results on non-experts’ abilities to comprehend the agent’s decisions [19, 20]. We build on these findings and adapt the sequence-to-sequence learning approach to a continuous robotics domain.

3 PROBLEM DEFINITION

We define the problem of providing explanations for task failures by extending the framework introduced by Chakraborti et al. [12] for producing explanations to goal-directed plans. In the framework, a planning problem $\Pi$ is defined by a transition function $\delta_{\Pi} : A \times S \rightarrow S \times \mathbb{R}$, where $A$ is the set of actions available to the agent, $S$ is the set of states it can be in, and $\mathbb{R}$ is a cost of making the transition. A planning algorithm $\Pi$ solves $\Pi$ subject to a desired property $\tau$ to produce a plan or policy $\pi$, i.e. $\Pi : \Pi \times \tau \mapsto \pi$. Here, $\tau$ may represent different properties such as soundness, optimality, etc. The solution to this problem, i.e. the plan, $\pi = (a_1, a_2, \ldots, a_n), a_i \in A$, which transforms the current state $l \in S$ of the agent to its goal $G \in S$, i.e. $\delta_{\Pi}(\pi, l) = (G, a_i \in \pi)$. The second term in the output denotes a plan cost $c(\pi)$.

Given the above framework, we define two explanation types. The first is from [12], and the second is contributed by our work:

$\mathcal{E}_\pi$: This explanation justifies to a human user that solution $\pi$ satisfies property $\tau$ for a given planning problem $\Pi$. For example, the user may ask “Why $\pi$ and not $\pi'$?”. In response to this question, $\mathcal{E}_\pi$ must enable the user to compute $\Lambda : \Pi \times \tau \mapsto \pi$ and verify that either $\Lambda : \Pi \times \tau \mapsto \pi'$, or that $\Lambda : \Pi \times \tau \mapsto \pi'$ for $\pi$, $\pi'$ which satisfy some criteria. $\mathcal{E}_\pi$ applies to the plan solution as a whole and can be elicited at any time. Approaches addressing $\mathcal{E}_\pi$ are discussed in Sec. 2.

$\mathcal{E}_{err}$: This explanation applies when an unexpected failure state, $f \in \mathcal{F}$, is triggered by a failed action in $(a_1, a_2, \ldots, a_n)$, and halts the execution of $\pi$. For example, the user may ask “The robot is at the table, but why did it not pick up my beverage?” In response to this question, $\mathcal{E}_{err}$ must allow the user to understand the cause of error in order to help the system recover.

In this work, we develop the second variant of explanations $\mathcal{E}_{err}$. We assume that both the algorithm $\Lambda$ and the plan $\pi$ are sound, and that the cause of error is triggered by a failure state $f \in \mathcal{F}$ from which an agent cannot recover without user assistance. For example, a situation in which a robot requires human help to discern an occluded object or pickup a tool out of reach. Our objective is to generate $\mathcal{E}_{err}$ such that the user (1) correctly understands the cause of failure, and (2) helps the agent recover from the error by providing a solution.

In the following sections, we present our methods to achieve the above objective. In Sec. 4, we introduce a set of information types, $\Lambda$, that are characteristics of $\mathcal{E}_{err}$. We then develop scripted explanations satisfying different $\lambda \in \Lambda$, and evaluate them to find a meaningful $\lambda$ that satisfy our objective for non-expert users (Fig. 1(b)). The results from Sec. 4 inform our efforts in Sec. 5 to automatically generate $\mathcal{E}_{err}$ without using pre-defined scripts (Fig. 1(c)).

4 INFORMATION TYPES OF $\mathcal{E}_{err}$

In order to generate $\mathcal{E}_{err}$, the first question we have to answer is: given an error while executing a plan $\pi$ for a particular task, what types of information should explanation $\mathcal{E}_{err}$ contain?
Table 1: The features that can encompass an explanation based on the study conditions. $a_t$ represents current action, $a_{t-1}$ represents last successful action, and $c_t$ represents captured environmental context.

| Study Condition | $a_t$ | $a_{t-1}$ | $c_t$ | Example Explanation for "object is occluded" failure |
|-----------------|-------|-----------|-------|-----------------------------------------------------|
| None            |       |           | N/A   | Robot could not find the object.                    |
| Action Based (AB) | ✓     |           |       | Robot could not find the object because the object is hidden from view. |
| Context Based (CB) | ✓     | ✓         |       | The robot finished scanning objects at its current location but could not find the desired object. |
| Action Based History (AB-H) | ✓     | ✓         | ✓     | The robot finished scanning objects at its current location, but could not find the desired object because the desired object is hidden from view. |
| Context Based History (CB-H) | ✓     | ✓         | ✓     | The robot finished scanning objects at its current location, but could not find the desired object because the desired object is hidden from view. |

For our application, we desire that $E_{err}$ is (1) accessible to non-experts, and (2) representative of the fault cause. Unfortunately, it is not clear from prior literature what information from an agent’s plan $\pi$, or its failure, satisfies these requirements. Ehsan et al. [29] propose that explanations for everyday users should take the form of rationales, which justify the agent’s decision in layperson’s terms. However, their rationales are trained from non-expert labels, do not reveal the true decision making process of an agent, and thus would not be able to disambiguate among visually similar robot errors (e.g., failure to grasp object due to kinematic constraints vs. object occlusion vs. a segmentation error). By contrast, prior work on fault diagnosis [29] has extensively studied how to describe error states, but such work exclusively targets expert users, with their rationales being trained from non-expert labels.

Thus, our first step is to determine what information $E_{err}$ should contain to be both accurate and interpretable by non-experts.

In this section, we define a set of information types, $\Lambda \subseteq \Lambda$, that we use to generate scripted explanations during a failure. In a user study with non-experts, we determine which $\lambda$ best help users identify the cause of a failure and suggest solutions to the failure. Specifically, we conducted a between-subjects user study in which $\Lambda$ consists of four values that are a cross-product of two factors: 2 (history, no history) $\times$ 2 (context-based, action-based). In the user study, the four explanation conditions were contrasted against a baseline condition. The five conditions are enumerated below:

- **Baseline (None):** Participants receive no explanation on the cause of error. This is the current standard in deployed robotic systems, e.g., [41].
- **Action-Based (AB):** Participants receive $E_{err}$ containing only the currently failed action $a_t$ as the cause of error.
- **Context-Based (CB):** Participants receive $E_{err}$ containing both $a_t$ and context, $c_t$, retrieved from the environment as the cause of error.
- **Action-Based-History (AB-H):** Participants receive $E_{err}$ containing the previous action $a_{t-1}$ and $a_t$ as the cause of error.
- **Context-Based-History (CB-H):** Participants receive $E_{err}$ containing $a_{t-1}$, $a_t$, and $c_t$ as the cause of error.

Table 1 summarizes the study conditions and provides example explanations for each condition. In the following sections, we discuss our experimental setup, the study procedure, and our results.

### 4.1 Goal, Action Space, & State Space

We conduct our experiment in Gazebo simulations of a Fetch robot [46]. The Fetch robot is a mobile manipulator with a differential drive base, a 7DoF arm, a parallel-jaw gripper, a pan-tilt head, and an adjustable torso. For sensing, the base includes a laser scanner and the head contains an RGB-D camera. The robot is simulated in a kitchen setting performing a pick-and-place task (as seen in Fig. 1a). The robot’s task is to move a task-specified object (e.g., milk carton) from the dining table to the kitchen counter.

Similar to prior work in robotics [5], we define the robot’s action space as the set $A = \{\text{move}, \text{segment}, \text{detect}, \text{findgrasp}, \text{grasp}, \text{lift}, \text{place}\}$, where move navigates the robot to a specified location, segment is used to identify which pixels in its sensory space correspond to objects, detect performs object detection to obtain a label for a given object, findgrasp executes grasp sampling to identify possible grasp poses for the gripper, grasp moves the robot arm into a grasp pose and closes the gripper, lift raises the arm, and place places a held object at a specified location.

The robot’s state at each time step $t$ is defined as $s_t \in S$, where $S = S_0 \cup S_1 \cup S_2 \cup S_3$. Here, $S_0 = S_0 \cup S_p$ denotes the set of names for all entities in the environment, where $S_0$ consists of [milk, coke can, ice cream, bottle, cup], and $S_p$ consists of: [dining table, kitchen counter]. $s(t) \in S_t$ is a vector of $(x, y, z)$ locations for each entity $s \in S$ at a given time step $t$. $s(t) \in S_t$ is defined by three tuples $\langle x_{vel}, y_{vel}, z_{vel} \rangle$, $\langle x_{pos}, y_{pos}, z_{pos} \rangle$, $\langle x_{pos}, y_{pos}, z_{pos} \rangle$ that describe the angular velocity, linear velocity and position of the robot at $t$. Finally, $S_k = \{k_{grasp}, k_{findgrasp}, k_{move}, k_{pick}, k_{detect}, k_{seg}\}$ where $S_k(t) \in S_k$ describes the status of each $a \in A$ at $t$, and whether each action is: active (0), completed (1) or errored (-1). Therefore, at all time steps, the number of elements in $S_k(t)$ is equal to the number of actions in $A$.

### 4.2 Failure Taxonomy

The agent’s initial state is defined as $s_0 = \{(0, 0, 0), (0, 0, 0), (0, 0, 0), (null)\}$, where the position tuple and the velocity tuples are set to zero, and the action states $s_k(0)$ are not defined. If there are no errors, the agent’s final state is defined as $s_f = \{(x_f, y_f, z_f), (0, 0, 0), (0, 0, 0), (1, 1, 1, ..., 1)\}$, where the position tuple is set to the goal location, the velocity tuples are zero, and each action state in $s_k(T)$ is 1. In this context, plan $\pi$ is the set of actions $\{a_1, a_2, ..., a_N\} \in A$ that transform the agent’s initial state $s_0$ to its final state $s_f$. We then define a failure $f$ in plan $\pi$ as the event when any action state in $s_k$ has a value -1.

Following the example of prior work [35], we study our work in the context of a representative sample of failures in robot behavior. We classify these failures using fault-tree analysis (Fig. 2).

---

1. Our fault-tree analysis identifies errors and solutions relevant to our domain, and we leave generating explanations for unknown errors for future work. Visualization of each failure is available at: https://youtu.be/YN3FaQ66OE.
To visualize the failure, participants were shown animated snapshots (GIFs) of actions leading up to a failure, and three perspective videos of the Fetch robot successfully executing the task with combinations. In our domain, each failure cause has an associated resolution action, not in the robot’s action space, but which can be selected by humans to rectify the cause of failure.

The fault-tree analysis also groups failure causes into causal groups, which we define as Internal and External. These categories roughly correspond to system and environment failures, respectively, in the prior work [35]. Internal failures are not apparent through visual cues in the environment and are often the result of failures of hardware or software modules. By contrast, external failures are often caused by unexpected conditions in the environment and are therefore visually apparent in the environment. In Section 4.5, we investigate the effect of different information types on users when the error stems from the different causal groups.

### 4.3 Study Design

Our objective is to evaluate the different information types of error explanations across a variety of failures. We simulated $|\mathcal{F}| = |\mathcal{S}_0| \times |\mathcal{F}_c| = 30$ failures to capture all possible object \times failure cause combinations. In our domain, each failure $f \in \mathcal{F}$ has a single cause in $\mathcal{F}_c$ and therefore a single resolution method $\mathcal{F}_c$. The study consisted of the following three stages.

#### Familiarization:
Participants in all conditions were first shown three videos of the Fetch robot successfully executing the task with randomly selected objects from $\mathcal{S}_0$ using a plan $\pi$. This served to accustom participants to the robot, its abilities, and its actions.

#### Baseline:
All participants were then shown six randomly sampled failure simulations from $\mathcal{F}_c$ one for every failure cause $F_c$. To visualize the failure, participants were shown animated snapshots (GIFs) of actions leading up to a failure, and three perspective shots of the robot in the final environment state. Participants were provided no explanations and asked to identify the cause of the failure and suggest a solution. Participant responses established the participants’ baseline understanding of the robot and the domain, allowing us to measure improvement in understanding.

**Explanation:** Finally, participants were exposed to twelve additional randomly sampled failures from $\mathcal{F}$ (different from Baseline), two for every failure cause $F_c$. Depending on the assigned study condition, a participant was either provided a hand-scripted explanation matching the information type of the assigned study condition, or the participant was provided no explanation if in the None condition. As before, the participant was required to identify the cause of the failure and suggest a solution. For each simulation, after identifying a failure and solution, participants received their accuracy score. This was the only feedback given to all participants.

### 4.4 Measures & Hypotheses

We evaluate participant performance using F1 score. In particular, we evaluate the difference between participant Baseline F1 score and their Explanation F1 score. The difference in F1 score is evaluated for the following measures:

- **Failure Identification ($FId$):** measures a participants’ ability to correctly identify the cause of each failure.
- **Solution Identification ($SId$):** measures a participants’ ability to correctly identify the solution to each failure.

Our data analysis then aims to answer the following questions with respect to the measures:

- **Q1:** Do action-based (AB) or context-based (CB) explanations lead to the greatest improvement in user failure identification ($FId$) and solution identification ($SId$)?
- **Q2:** Does the inclusion of history within an explanation improve users’ failure identification ($FId$) and solution identification ($SId$)?
- **Q3:** How do users’ failure identification ($FId$) and solution identification ($SId$) compare for Internal vs External robot errors?

### 4.5 User Evaluation of Scripted Explanations

**Participants.** We recruited 80 individuals from Amazon’s Mechanical Turk. Since our target audience is non-experts, we filtered out 10 participants for achieving 100% accuracy in the Baseline stage, under the assumption that they were not novices. The remaining 70 participants included 51 males and 19 females, all whom were 18 years or older (M = 35.2, SD = 9.4). Due to the exclusion criteria, each study condition had 13-15 participants. The task took on average 20 - 40 minutes and participants were compensated $3.50.

**Data Analysis.** The data on the FId and SId metrics are analyzed with a two-way ANOVA for Q1 and Q2 and a one-way ANOVA for Q3, followed by a Tukey HSD post-hoc test for each.

Fig. 3a and Fig. 3b answer Q1 by showing the benefit of including environmental context (CB, CB-H conditions) in failure identification ($FId$) and solution identification ($SId$). In both figures, we see that explanations with context have the highest improvement in FId and SId scores. Specifically, the presence of context had a significant effect on FId ($F(2,67)= 6.95, p=0.0018$), with a significant Fid improvement for Context-based explanations over both None ($t(67)=3.729, p=0.0012$) and Action-based ($t(67)=2.923, p=0.014$) explanations. Similarly, the presence of context had a trending effect on SId ($F(2,67)=2.92, p=0.06$), with a significant improvement in
Figure 3: Average F1 score across explanation conditions grouped by Context Based vs. Action Based (a-b) and History vs. No History (c-d). In Fig. 3, 4, and 6, statistical significance is reported as: *p < 0.05, **p < 0.01, ***p < 0.001

Figure 4: Average F1 score across all conditions grouped by Internal versus External errors.

SId for Context-based explanations vs. None (t(67)=3.12, p=0.007). This indicates that the inclusion of environmental context in the CB explanation conditions (CB, CB-H) helped participants better understand the underlying causes of the failures thereby allowing them to better assist the robot.

Fig. 3c and Fig. 3d answer Q2 by showing the benefit of including history (AB-H, CB-H conditions), on FId and SId. In both figures, history-based explanations have the highest improvement in FId and SId scores. Similar to the effects of including context, including history had a significant improvement on FId (F(2,67)= 3.36, p=0.003), with History vs. None as significant (t(67)=3.447, p=0.003). Although including history did not have a significant effect on SId overall (F(2,67)= 1.38, p=0.25), we observe a significant difference in improvement between History-based explanations vs. None (t(67)=3.1857, p=0.006). This supports the idea that knowledge of the most recently completed action (AB-H, CB-H conditions) can help users gauge what a robot was able to successfully accomplish, thereby helping users better pinpoint the exact cause of failure and provide correct suggestions for recovery.

Our analysis so far investigates the independent effects of including context and history on explanation utility. The results suggest that context-based explanations incorporating history, i.e. CB-H explanations, are the best suited to non-experts. We next consider each explanation type individually and their efficacy for non-experts based on the causal group of the originating fault.

Fig. 4 answers Q3 by showing the different effects of the explanation types for failures stemming from the different causal groups—Internal and External failures. Explanations have a significant effect on the improvement in FId for External errors (F(4,62)= 3.53, p=0.01), with CB-H showing the most pronounced improvement, specifically vs. AB (t(62)=-3.216, p=0.017) and vs. None(t(62)=-3.046, p=0.027). Additionally, we see a significant effect of explanations in improving FId for Internal errors (F(4,62)= 4.39, p=0.003), with a significant difference in CB-H vs. None (t(62)=-3.955, p=0.0018). With respect to improvement in SId for External errors, we see a trending effect of explanations (F(4,62)=2.16, p=0.083) with a trending difference between AB-H vs. None (t(62)=2.648, p=0.073). For Internal errors, we notice a trending effect of explanations (F(4,62)=2.37, p=0.061), but with a significant difference between CB-H and None (t(62)=2.86, p=0.044). Overall, we find that CB-H explanations are valuable to participants for both error types, but especially for the Internal case when failure causes are not discernible through the environment.

5 AUTOMATED GENERATION OF $\mathcal{E}_{err}$

In Sec. 4 we discovered CB-H explanations to be most effective. In this section, we introduce an automated explanation generation system that can generate the CB-H explanations word by word, without a template.

5.1 Encoder-Decoder Model Overview

We adapt a popular encoder-decoder network [4, 6] utilized by [20] to train a model to generate CB-H explanations from an agent’s state. The model’s features, $U$, are derived from the state space, $S$ (see Sec. 5.2), and are comprised of environment features $X$, continuous.
The x-axis represents the true labels, and the y-axis represents the predicted labels.

The encoder and decoder are comprised of Gated Recurrent Units (GRU) [16]. Given a sequence of environment features, $X = \{x_1, x_2...x_n\}$, the encoder generates the context embedding at each step $h_n$ conditioned on the previous context embedding $h_{n-1}$, the environment features $e$, and the object of interest $o$, as its initial input, $s_0$. The decoder also generates and uses a weighted attention vector, $c_i$, for step $i$ (initialized with $c_0 = 0$). At each step, $c_i$ attends over the features in $s_0$ and $s_i$, the decoder’s input at the previous step. The decoder then updates its state according to the function $s_i = GRU(s_i, y_{i-1}, c_i)$, where $y_{i-1}$ is the previous predicted word, and a word, $y_i$, is predicted from the maximum softmax probability over $s_i$. A complete explanation is generated when the decoder predicts the ‘END’ token.

5.2 Feature Set

Recall from Sec. 4.1 that the agent’s state space is defined as $S = S_e \cup S_j \cup S_k \cup S_p$. We derive the features, $U = x \cup u \cup o \cup a$ for the encoder-decoder model from $S$. The object of interest, $o \in S_e$, is specified as part of the task and represented by its word embedding in $U$. The environment, $X$, is comprised of the word embeddings of the names of the objects, $Ob_{jk}$, located in the robot’s area of interest, such that $\forall o^j \in Ob_{jk}$. $o^j \in S_e$. The remaining continuous features, $N = \{Rel_{_a-Goal}, Rel_{_a-o}, u_{ang}, v_{lin}, S_k, Rel_{_a-Ob_{jk}}, o_p\}$, characterize the robot and the target object in the environment. $Rel_{_a-Goal}$ is the distance of the robot from its goal location, $Rel_{_a-o}$ is the distance of the robot from the target object, $u_{ang}$ and $v_{lin}$ are the angular and linear velocities of the robot base, and $S_k$ are the action statuses as defined in Sec. 4.1. Additionally, $Rel_{a-Ob_{jk}}$ is the distance between the desired object $o$ and the objects in $Ob_{jk}$, and $o_p$ is a boolean that evaluates to true if $o \in Ob_{jk}$. Note that not all the features in $N$ contain valid values at all times. If a feature value is invalid, the feature is masked before it is concatenated into $U$.

5.3 Data Collection

To further evaluate the generalizability of our method across environments, we expand our data to include simulations from an office environment (Fig. 1(a)), in addition to the kitchen environment introduced in Sec. 4. The office environment contains different objects and locations, i.e., entities $S_e$, but the robot’s pick-and-place task remains the same. Entities in the office environment have a one-to-one correspondence to the entities in the kitchen environment.

Our dataset $D$ consists of 72 simulations (60 and 12 from the kitchen and office environments, respectively). Each timestep in $D$ is defined by $u_t$, where $u_t \in U$ represents the input features to our encoder-decoder model at timestep $t$. Each simulation begins with $n$ active or successful action timesteps, denoted by $s_0(t) = 0$ or $s_k(t) = 1$, and ends with $m$ error timesteps, denoted by $s_0(t) = -1$. In our work, $n$ ranges from 15 to 20, and $m = 10$. The $m$ error timesteps simulate a robot repeatedly attempting to autonomously remedy a failure upon encountering it, reflecting a real-world solution to errors where robots try to repeat actions that fail [5].

Given our dataset, we annotate error timesteps with a CB-H explanation, $E_{err}$, and annotate successful or active timesteps with a natural language rationalization of the state, $E_{corr}$, as in [19]. In our work, examples of such rationales include, “robot moving to dining table” and “robot segmented objects in the scene”. Additionally, $E_{corr}$ explanations were only used for model training, and were not a focus of the human subjects study in Sec. 5.5. The total size of $D$ is 2100 timesteps where there are 1380 successful or active timesteps, and 720 error timesteps.

5.4 Model Training & Evaluation

Our encoder-decoder model is trained with the 60 kitchen simulations using a two-step grouped leave one out cross validation (LOOCV) with 10 folds, where the grouped LOOCV leaves out an entire simulation for each failure cause, $F_c$. The first grouped LOOCV creates a split between the training set, $d_{tr}$, and test set, $d_{te}$, while the second LOOCV creates the validation set, $d_{va}$. As a result, in each fold, $d_{tr}$ includes 48 simulations with 480 error explanations, while $d_{va}$ and $d_{te}$ include 6 simulations, each with 60 error explanations. To evaluate each fold, we utilize an evaluation set, $d_{eval}$, which includes the 12 office simulations with 120 error explanations.

Training. Our models trained for an average of 180 epochs, depending on the validation loss. We train with a batch size of 20. Our GRU cells in the encoder have a hidden state size of 20 and the GRU cells in the decoder have a hidden state size of 49. We train our model using a Cross Entropy loss optimized via Adam with a learning rate of 0.0001.

Evaluation. Fig. 5 shows the average performance of the model on $d_{eval}$ across the 10 folds of cross-validation. The confusion matrix
We wished to evaluate whether (1) the model generated explanations improved participants’ failure and solution identification compared to the None condition, and (2) the model generated explanations performed on par with the hand-scripted explanations in improving participants’ performance.

Participants. We recruited 45 individuals from Amazon’s Mechanical Turk. After applying the exclusion criteria as before, the remaining 41 participants included 25 males and 16 females, all whom were 18 years or older (M = 39, SD = 11.3). Due to the exclusion criteria, each study condition had 12-15 participants. The task took roughly 20 - 40 minutes and participants were compensated $3.50.

Data Analysis. The data on Fl and Sld metrics are analyzed with a one-way ANOVA followed by a Tukey HSD post-hoc test.

Fig. 6 answers our hypotheses by showing that CB-H-M explanations are just as effective as CB-H scripted explanations. We observe a significant effect of explanations on Fl (F(2,38)=10.52, p=0.0002) and Sld (F(2,37)=3.94, p=0.027). With respect to Fl we observe a significant improvement in participant accuracy between CB-H-M and None (t(38)=-4.158, p=0.00049), and no significant difference between CB-H and CB-H-M (t(38)=-0.208, p=0.97). With respect to Sld we observe a trending difference in improvement between CB-H-M vs. None (t(37)=2.354, p=0.060), a significant difference between CB-H vs. None (t(37)=2.561, p=0.038) and no significant difference between CB-H and CB-H-M (t(37)=0.215, p=0.974). Thus, we conclude that given CB-H-M explanations, participants perform just as well in helping the robot as when given CB-H explanations.

6 CONCLUSION & DISCUSSION

In this work, we investigate what types of information within an explanation help non-experts identify robot failures and help assist in recovery. We introduce a new type of explanation, E_{err}, which has not been previously addressed in the XAI community, and which describes the cause of unexpected failures amidst plan execution. Our results indicate that for explanations to improve failure and solution identification, they should encompass both environmental context and history of past successful actions. Furthermore, in our first user evaluation we showcase the importance that context-based-history explanations serve in the cases of Internal errors, which are not visually observable through environmental changes. Additionally, we investigate a method to autonomously generate such explanations, and verify that they are as effective as its scripted counterpart and generalizable across environments.

Our work brings XAI techniques into the domain of fault recovery and aims to aid non-expert users (1) understand unexpected failures of a complex robot system and (2) provide recovery solutions in such an event. Although our work includes important contributions, there are limitations that should be addressed by future work. First, while the context-based-history explanations are useful for assisting in failure recovery, they are not guaranteed to be useful to all non-experts. Therefore future work can explore tailoring explanations to individual users, perhaps with the reinforcement learning techniques used in recommender systems [45]. Second, our work has characterized the utility of context and history in providing meaningful E_{err}, but we have assumed that explanations can be arbitrarily long. Future work should investigate additional factors that characterize a good E_{err}, and the tradeoffs of providing more information vs. remaining concise. Finally, while the current encoder-decoder model can generalize over varying failure scenarios, there is still room to improve its generalizability to additional situations. Future work can investigate a wider range of simulation domains, tasks, and failures.

7 ACKNOWLEDGMENTS

This material is based upon work supported by the NSF Graduate Research Fellowship under Grant No. DGE-1650044.
REFERENCES

[1] Boussad Abci, Maan El Badouei El Najar, Vincent Coquemont, and Gérard Dherbomez. 2020. An informational approach for sensor and actuator fault diagnosis for autonomous mobile robots. Journal of Intelligent & Robotic Systems 99, 2 (2020), 387–406.

[2] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). IEEE Access 6 (2018), 52138–52160.

[3] Dan Amir and Ofra Amir. 2018. Summarizing agent behavior to people. In Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems. 1168–1176.

[4] Andrea Bauer, Dirk Wollherr, and Martin Buss. 2008. Human–robot collaboration: a survey. International Journal of Humanoid Robotics 5, 01 (2008), 47–66.

[5] Joost Bastings. 2018. The Annotated Encoder-Decoder with Attention. TESLA 2018.

[6] Richard Bloss. 2011. Mobile hospital robots cure numerous logistic needs. Indus. Collaborative Industrial Robots.. In Proc. of the IEEE international conference on robotics and automation. 510–518.

[7] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. 2020. Understanding its effects on human perceptions. In Proc. of the IEEE international conference on robotics and automation. 510–518.

[8] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. 2019. Interpreting deep learning. arXiv preprint arXiv:1711.06178 (2017).

[9] Richard Blass. 2008. Mobile robot coordination. In Proc. of the 25th International Conference on Intelligent User Interfaces. 518–519.

[10] Eliahu Khalastchi and Meir Kalech. 2018. A sensor-based approach for fault detection and diagnosis for robotic systems. Autonomous Robots 42, 6 (2018), 1231–1248. https://doi.org/10.1007/s10462-017-9688-z

[11] D. Crestani, K. Godary-Dejean, and L. Lapierre. 2015. Enhancing fault tolerance of failure-handling approaches for planning systems—Replanning vs. recovery. In Proc. of the IEEE international conference on robotics and automation. 2703–2710.

[12] Lynne Parker and Balajee Kannan. 2006. Adaptive Causal Models for Fault Diagnosis and Recovery in Multi-Robot Teams. In IROS. IEEE, 137–141.

[13] Ola Pettersson, L. Karlsson, and A. Saffiotti. 2007. Model-Free Execution Monitoring in Behavior-Based Robotics. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) 37, 4 (aug 2007), 890–901. https://doi.org/10.1109/TSMCB.2007.895359

[14] Mohammad Zaman, Gerald Steinbauer, Johannes Maurer, Peter Lepej, and Suzana Dherbomez. 2020. An informational approach for sensor and actuator fault detection and diagnosis for robotic systems. Autonomous Robots 46, 6 (2018), 1231–1248. https://doi.org/10.1007/s10462-017-9688-z

[15] Eliahu Khalastchi and Meir Kalech. 2018. On fault detection and diagnosis in robotic systems. ACM Computing Surveys (CSUR) 51, 1 (2018), 1–24.

[16] Safdar Zaman, Gerald Steinbauer, Johannes Maurer, Peter Lepej, and Suzana Dherbomez. 2020. An informational approach for sensor and actuator fault detection and diagnosis for robotic systems. Autonomous Robots 46, 6 (2018), 1231–1248. https://doi.org/10.1007/s10462-017-9688-z

[17] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. 2019. Explicability? legibility? predictability? transparency? privacy? security? The emergent landscape of interpretable agent behavior. In Proc. of the international conference on automated planning and scheduling, Vol. 29. 86–96.

[18] Richard Blass. 2011. Mobile hospital robots cure numerous logistic needs. Indus. Collaborative Industrial Robots.. In Proc. of the IEEE international conference on robotics and automation. 510–518.

[19] Xing Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtuo Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In 2018 IEEE International Conference on Data Mining (ICDM). IEEE, 587–596.

[20] Ama Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). IEEE Access 6 (2018), 52138–52160.

[21] D. Crestani, K. Godary-Dejean, and L. Lapierre. 2015. Enhancing fault tolerance of failure-handling approaches for planning systems—Replanning vs. recovery. In Proc. of the IEEE international conference on robotics and automation. 510–518.

[22] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. 2020. The Emerging Landscape of Explainable AI Planning and Decision Making. arXiv preprint arXiv:2002.11697 (2020).

[23] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampti. 2017. Plan explanations as model evaluation: Moving beyond explanation as soliloquy. arXiv preprint arXiv:1701.08317 (2017).

[24] Kai-Hsiung Chang, Chang, Hyungoo Han, and William B Day. 1993. A comparison of failure-handling approaches for planning systems—Replanning vs. recovery. Applied Intelligence. 3, 4 (1993), 275–300.

[25] Joost Bastings. 2018. The Annotated Encoder-Decoder with Attention. TESLA 2018.

[26] Sarath Sreedharan, Siddharth Srivastava, David E Smith, and Subbarao Kambhampti. 2019. Explicability? legibility? predictability? transparency? privacy? security? The emergent landscape of interpretable agent behavior. In Proc. of the international conference on automated planning and scheduling, Vol. 29. 86–96.

[27] Chao Chen, Rui Xu, Shengying Zhu, Zhaoyu Li, and Huiping Jiang. 2020. RPRS: A reactive Plan repair strategy for rapid response to failure problems of deep space missions. Acta Astronautica (2020).

[28] Mike Wu, Michael C Hughes, Solon Barocho, Mauricio Zatti, Volker Roth, and Finale Doshi-Velez. 2017. Beyond sparsity: Tree regularization of deep models for interpretability. arXiv preprint arXiv:1711.06178 (2017).

[29] Martin Hägele, Klas Nilsson, J Norberto Pires, and Rainer Bischoff. 2016. Industrial Robot: An International Journal 41, 5–10.

[30] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). IEEE Access 6 (2018), 52138–52160.

[31] Tathagata Chakraborti, Anagha Kulkarni, Sarath Sreedharan, David E Smith, and Subbarao Kambhampati. 2019. Explicability? legibility? predictability? transparency? privacy? security? The emergent landscape of interpretable agent behavior. In Proc. of the international conference on automated planning and scheduling, Vol. 29. 86–96.

[32] Eliahu Khalastchi and Meir Kalech. 2018. A sensor-based approach for fault detection and diagnosis for robotic systems. Autonomous Robots 42, 6 (2018), 1231–1248. https://doi.org/10.1007/s10462-017-9688-z

[33] Ola Pettersson, L. Karlsson, and A. Saffiotti. 2007. Model-Free Execution Monitoring in Behavior-Based Robotics. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) 37, 4 (aug 2007), 890–901. https://doi.org/10.1109/TSMCB.2007.895359

[34] Varun Rai. 2020. Explainable AI From black box to glass box. Journal of the Academy of Marketing Science 48, 1 (2020), 1–24.

[35] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampti. 2020. The Emerging Landscape of Explainable AI Planning and Decision Making. arXiv preprint arXiv:2002.11697 (2020).

[36] Lynne Parker and Balajee Kannan. 2006. Adaptive Causal Models for Fault Diagnosis and Recovery in Multi-Robot Teams. In IROS. IEEE, 137–141.

[37] Eliahu Khalastchi and Meir Kalech. 2018. A sensor-based approach for fault detection and diagnosis for robotic systems. Autonomous Robots 42, 6 (2018), 1231–1248. https://doi.org/10.1007/s10462-017-9688-z

[38] Sarath Sreedharan, Siddharth Srivastava, David E Smith, and Subbarao Kambhampti. 2019. Why Can’t You Do That HAL? Explaining Unsolvability of Planning Problems. In Proc. of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 1135–1144.

[39] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). IEEE Access 6 (2018), 52138–52160.

[40] Ran Pratik Patil, Devi Patil, and Dhiruv Bhatra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proc. of the IEEE international conference on computer vision. 618–626.

[41] Gayane Kazhoyan, Arthur Niedzwiecki, and Michael Beetz. 2020. Towards Plan Transformations for Real-World Mobile Fetch and Place. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. 6261–6270.
[52] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. 2017. Plan explicability and predictability for robot task planning. In 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 1313–1320.

[53] Zhengjiang Zhang and Junghui Chen. 2019. Fault detection and diagnosis based on particle filters combined with interactive multiple-model estimation in dynamic process systems. ISA transactions 85 (2019), 247–261.