



## Similar failures of consideration arise in human and machine planning

Alice Zhang <sup>a,b</sup>\*, Max Langenkamp <sup>b</sup>, Max Kleiman-Weiner <sup>a</sup>, Tuomas Oikarinen <sup>b</sup>, Fiery Cushman <sup>c</sup>

<sup>a</sup> Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America

<sup>b</sup> Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America

<sup>c</sup> Department of Psychology, Harvard University, Cambridge, MA 02139, United States of America

### ARTICLE INFO

Dataset link: <https://osf.io/et89w>

**Keywords:**

Decision-making  
Cognitive puzzles  
Consideration  
Machine learning

### ABSTRACT

Humans are remarkably efficient at decision making, even in “open-ended” problems where the set of possible actions is too large for exhaustive evaluation. Our success relies, in part, on processes for calling to mind the right candidate actions. When these processes fail, the result is a kind of puzzle in which the value of a solution would be obvious once it is considered, but never gets considered in the first place. Recently, machine learning (ML) architectures have attained or even exceeded human performance on open-ended decision making tasks such as playing chess and Go. We ask whether the broad architectural principles that underlie ML success in these domains generate similar consideration failures to those observed in humans. We demonstrate a case in which they do, illuminating how humans make open-ended decisions, how this relates to ML approaches to similar problems, and how both architectures lead to characteristic patterns of success and failure.

### 1. Introduction

Many everyday decision making problems present too many possible solutions for exhaustive consideration: Imagine all the books we could choose to read next; all the people we could spend a Sunday afternoon with; all the different ways to express an idea. Fortunately, when facing these kinds of open-ended problems, we efficiently call to mind a small set of good candidates (Hauser, 2014; Morris, Phillips, Huang, & Cushman, 2021; Nedungadi & Hutchinson, 1985; Phillips & Cushman, 2017; Zhang et al., 2021). Indeed, efficient consideration of promising options during decision-making is a cornerstone of human intelligence (Phillips, Morris, & Cushman, 2019; Simon, 1955).

However, efficient consideration also generates characteristic errors. In some cognitive puzzles, the correct solutions systematically fail to come to mind (Batchelder & Alexander, 2012; Gilhooley & Murphy, 2005). For example, in chess, a novice player might not consider a sequence of moves that requires sacrificing a valuable piece to guarantee a win later on. If such a solution were proposed to them, they could understand why it is favorable. However, the cognitive puzzle arises because the player does not ever consider the correct sequence of moves in the first place. This kind of problem arises not only in decision-making, of course, but also in many other tasks (Dasgupta, Schulz, & Gershman, 2017) such as causal inference (Bramley & Xu,

2023), theory learning (Ullman, Goodman, & Tenenbaum, 2012), and reasoning (Wason, 1968).

Our goal is to better understand how people efficiently generate candidate solutions to decision-making problems, and also how this leads to predictable failures and puzzles. We focus on one element of candidate generation in particular: Value generalization, or the way that people rely on past experience of reward to predict which candidate actions will be rewarding in the future. The key idea is simple: If the necessary action to solve the *current* problem has been assigned high heuristic value based on prior experience, it will come quickly to mind. If not, the correct solution will evade consideration, resulting in a cognitive puzzle. Since value generalization plays a key role in contemporary machine learning (ML) architectures for choice, our approach is to ask whether comparable patterns of success and failure arise for humans and for a representative ML architecture.

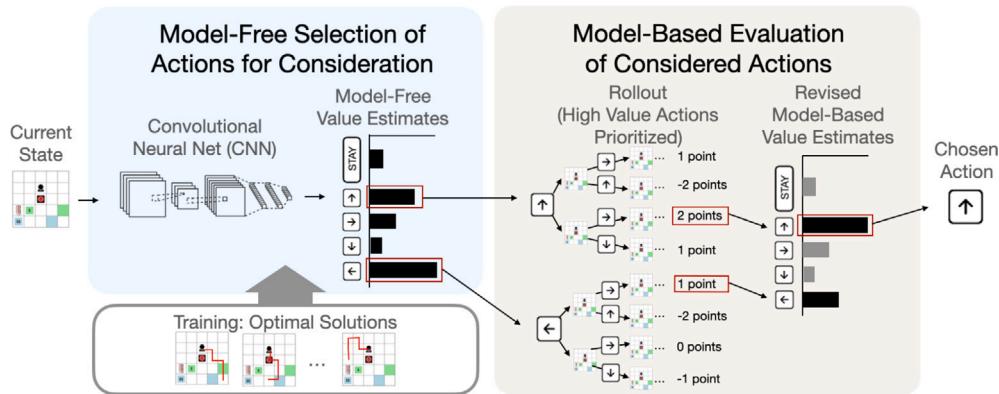
#### 1.1. Candidate generation by value generalization

There are many different ways in which people come to generate candidate solutions to problems. We focus on one method in particular: Generating candidate solutions to a current task by generalizing from the solutions that proved valuable in similar contexts in the past (Johnson & Raab, 2003; Kaiser et al., 2013; Klein, 1993; Morris et al., 2021;

\* Corresponding author.

E-mail addresses: [alice.zhang@psy.ox.ac.uk](mailto:alice.zhang@psy.ox.ac.uk) (A. Zhang), [maxlangenkamp@me.com](mailto:maxlangenkamp@me.com) (M. Langenkamp), [maxkw@uw.edu](mailto:maxkw@uw.edu) (M. Kleiman-Weiner), [toikarinen@ucsd.edu](mailto:toikarinen@ucsd.edu) (T. Oikarinen), [cushman@fas.harvard.edu](mailto:cushman@fas.harvard.edu) (F. Cushman).

URLs: <https://orcid.org/0009-0005-5037-0995> (A. Zhang), <https://orcid.org/0000-0002-6929-9982> (F. Cushman).



**Fig. 1.** In the family of ML planning architectures that inspired our approach, a model-free estimator guiding a model-based evaluator is able to efficiently solve open-ended problems by preferentially evaluating actions with high estimated value. The model-free component is implemented by a Convolutional Neural Net (CNN) trained on optimal solutions. The model-based component is implemented by Monte-Carlo rollout; in our case, First Visit Monte-Carlo. This motif of model-free value estimates guiding efficient model-based evaluation is echoed in contemporary work on human planning.

Peters, Fellows, & Sheldon, 2017; Zhang et al., 2021). For instance, suppose that a person is trying to generate candidate chess moves in a position they have never played before. The moves that come to mind might be generalized from those that proved useful in similar past games. Or, suppose that person is trying to generate some options to cook for a dinner party on a generous budget but a short time-frame; the dishes that come to mind might be those that worked well under similar constraints in the past. The key advantage of generalizing from past cases is that it can be computationally frugal, helping people to make good decisions under resource constraint (Lieder, Griffiths, & Hsu, 2018; Simon, 1955). In the language of reinforcement learning, generalizing from the solutions that worked in the past is a form of “model free” solution.

Once a small set of good candidate options has been generated, however, it is feasible to apply more computationally demanding methods to refine the value estimates of each of them and choose the best. For instance, the chess player who has called to mind candidate moves might evaluate them by simulating the future states of the board that would likely arise, based on their knowledge of the rules of the game and the likely moves of their opponent. Or, the cook who has called to mind several candidate dishes might evaluate them by considering how much their guests would appreciate them, how difficult they would be to make, etc., given their knowledge of the situation. In the language of reinforcement learning, this method of value estimation is “model-based”, because it derives a value estimate from a model of the causal dynamics of the environment. And, indeed, people often evaluate candidate actions by model-based methods ([Dolan & Dayan, 2013](#)).

It is not surprising that different methods of evaluation are favored for distinct stages of the decision-making process (i.e., option generation versus selection). The problems presented by each stage are distinct and, in general, we should expect different kinds of algorithms to be favored for different kinds of optimization problem ([Wolpert & Macready, 1997](#)).

In summary, then, existing evidence suggests that when faced with open-ended problems people sometimes use distinct mechanisms to generate candidates and to evaluate those candidates. In these cases, generation depends on fast-and-frugal model-free methods, while evaluation depends on effortful but more accurate model-based mechanisms (Hauser & Wernerfelt, 1990; Kalis, Kaiser, & Mojisch, 2013; Morris et al., 2021). Similar heuristic methods of option generation (or hypothesis generation) have been considered for other kinds of tasks, such as learning causal models or theories (Zhao, Lucas, & Bramley, 2024). This is what gives rise to the failures of consideration characteristic of certain puzzles. Model-free mechanisms may assign low value to the correct answer, making it unlikely to come to mind; yet,

model-based processes may be able to quickly and decisively recognize the value of those solutions *if only* they came to mind. We interrogate cognitive puzzles of this kind.

### 1.2. Machine learning model of value function approximation

Recently, machine learning architectures have achieved performance on par with human experts in several open-ended problems such as the games of chess and Go (Silver et al., 2018). These successes depend in part on an architectural motif that efficiently selects candidate actions for consideration and further evaluation by using heuristic value estimates based on past experience. As we have seen, this motif parallels some current models of how humans generate efficient consideration sets. We therefore ask whether it can be used to understand not only the shared successes of human and ML planning, but also their characteristic failures of consideration.

Specifically, several recent ML algorithms exhibiting excellent performance in open-ended problems, such as AlphaGo and its descendants, rely on common architectural principles (Fig. 1). Specific sequences of candidate actions are evaluated by simulating their likely consequences, a model-based method of evaluation sometimes called roll-out or tree search. In open-ended problems there are, however, too many action sequences for exhaustive consideration. A key innovation addresses this problem by guiding tree search towards the most promising candidates based on a heuristic estimate related to their value. This estimate is furnished by a myopic neural net trained on previously played or observed games, a model-free method. Since players encounter novel circumstances — states of the game that have never before been observed — the network must generalize from past experience. In this manner, the neural network efficiently guides model-based roll-outs towards promising candidates. In sum, this architecture employs computationally cheap but imprecise model-free estimates of value generate a feasible set of actions for consideration, and then devotes computationally expensive and precise model-based estimates of value to the promising candidates within this set.

This suggests a potential high-level correspondence between human and machine approaches to open-ended decision problems, although the precise implementations surely differ in their details. One obvious approach to establishing correspondence between humans and ML methods is to see whether they show comparable areas of success. Our experiments provide this kind of evidence. Yet, our main focus is not areas of shared success, but rather on a common pattern of failures. After all, starkly different algorithms can be used to arrive at the same successful solutions. Often, an algorithm's most distinctive fingerprint is instead its pattern of failures (Saxe, 2005). In order to elicit failures, we designed a task for which we expected certain solutions to systematically evade consideration, whether by humans or machine planners.

### 1.3. A sequential planning task to elicit failures of consideration

We designed a sequential decision-making task of the kind that the relevant ML architectures are optimized to solve. Specifically, we designed a gridworld game in which an agent can move valuable objects onto target locations to earn points. We described these objects to humans as cargo, and the setting as a train yard. The grid also contained trains (autonomous moving objects that destroy any objects they hit, resulting in a loss of points and bringing the train to a stop) and switches (which change the direction of motion of the trains).

We trained humans and an ML planning architecture on a set of grids comprising several mundane types, such as moving cargo onto its target location (“Push Control” cases), or flipping a switch to redirect a train away from valuable cargo (“Switch Control” cases), among others (Fig. 2). Reflecting the underlying structure of the problem, these training cases often present situations in which a train is heading towards valuable cargo and flipping a switch to redirect the train saves it. And, they often present situations in which a train is *not* heading towards valuable cargo, and to push that cargo in the way of the train would be costly and pointless. Crucially, however, our training regime omitted two particular types of configuration that would rarely arise naturally. In the first (“Switch Sacrifice” cases), the optimal action is to redirect the train away from collision with a high-value object such that it unavoidably collides with a low-value object. In the second (“Push Sacrifice” cases), the optimal action is to push a low value object into the path of the train in order to stop it, thus preventing it from hitting a high-value object.

To preview our results, both humans and the ML planner reliably identified the optimal solution to Switch Sacrifice cases, but often failed on Push Sacrifice cases. This occurs despite the fact that the potential outcomes for both types of case are equated, and despite the fact that neither type of case is present in training. Our findings suggest that this occurs because, based on training, switching a train away from cargo is heuristically estimated to have high value, while pushing cargo into the path of a train is heuristically estimated to have low value. Because these heuristic value estimates govern which actions get evaluated, the optimal action is quickly discovered in Switch Sacrifice cases but not in Push Sacrifice cases.

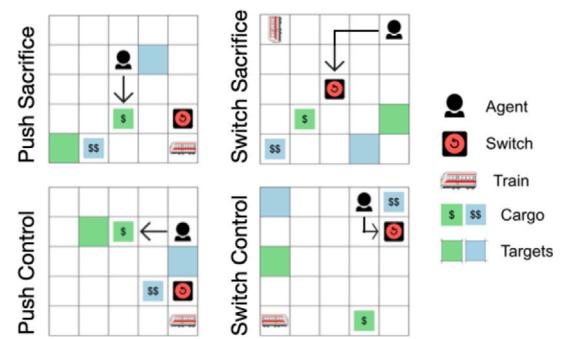
Against this background, we use a variety of manipulations to establish the robust correspondence between the performance of the ML architecture and the performance of human participants. We also take advantage of the *in silico* nature of the computational architecture to interrogate the precise causes of the characteristic failures of consideration it shares with human participants.

### Relationship to the trolley problem

Our task bears an obvious resemblance to the well-studied “trolley problem” (Foot, 1967; Greene, 2014), and was inspired by it. But, it also differs from the trolley problem in two very important ways. First, it does not pose a moral dilemma: The objects in question are merely cargo, not people. We have no reason to believe that participants viewed any of the tasks we presented as presenting *moral* dilemmas. Second, our data suggest that participants do not consider and then reject the possibility of pushing cargo in front of a train; rather, they *fail to consider* the action in the first place. Thus, our “trolley puzzle” is quite different from the classic trolley problem, in which *harm to people* is *considered and then rejected* on moral grounds. In the discussion, however, we return to consider ways in which these two distinct phenomena may nevertheless be related.

## 2. General methods

We asked participants to solve a series of railway operations problems in a stylized grid (or “gridworld”; Fig. 2). These were structured as finite Markov decision processes—the kinds of problems to which standard reinforcement learning mechanisms are well suited. Participants’ objective was to maximize points by pushing cargo onto target



**Fig. 2.** Example starting state and optimal solution for each test case type in our task. Points are earned when cargo are pushed onto their targets and points are lost when the train collides with cargo. Push Sacrifice and Switch Sacrifice cases favor the sacrifice of the lower value cargo in order to prevent the train from colliding with the higher value cargo. Push Control and Switch Control cases favor pushing cargo onto their target locations and hitting the switch to prevent collisions, respectively. Control cases were among the categories featured in training, and were also included at test. Push Sacrifice and Switch Sacrifice cases were omitted from training and presented exclusively at test.

locations, earning two points for high value cargo and one point for low value cargo, while avoiding collisions with the train. A train moved across the screen in a straight line, one grid cell per time step, unless the participant flipped a switch to turn it 90° counterclockwise. The participant lost points in the event of a train collision: one for low value cargo, two for high value cargo, and four for a collision with themselves. Collisions always stopped the motion of the train. Participants performed a sequence of five actions by moving in any cardinal direction or standing still.

Participants completed a training phase comprising four problem-types, but omitting the Switch Sacrifice and Push Sacrifice cases. Specifically, training included Push Control cases that favored pushing object(s) into their target locations (47%), Switch Control cases that favored redirecting the train away from an object collision (25%), “Push Away” cases that favored pushing an object out of the path of the train (20%), and “Do Nothing” cases where nothing could be done to prevent an object collision (8%). Grids of each type were procedurally generated by randomly setting the starting positions of key items within a set of constraints (see SI).

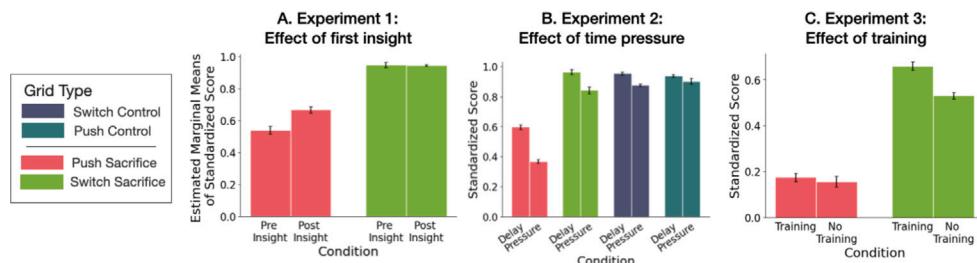
Then, during the test phase, we presented participants with Push Sacrifice and Switch Sacrifice cases for the first time, as well as novel instances of the Switch Control and Push Control types seen in training. In order to standardize the comparison of participant scores across distinct categories of test grids we constructed all test grids so that the minimum and maximum attainable scores differed by exactly one point. Then, we standardized participant scores to range (0, 1) by subtracting the grid-specific minimum attainable score.

All studies were approved by the Institutional Review Board and performed with participants’ informed consent. Participants were recruited from Amazon MTurk through CloudResearch and paid for their participation. All study materials, code, and analysis scripts are available at <https://osf.io/et89w>.

### 2.1. Experiment 1: Eliciting failures of consideration

The goal of Experiment 1 was to characterize overall patterns of failure and success across different types of test trials. Although all test trials were novel in their particulars, recall that two classes of test trial were represented in training (Push Control and Switch Control), and two classes were not (Push Sacrifice and Switch Sacrifice).

Naturally, participants might more easily solve the types of cases represented in training, compared to the types unrepresented in training. Our focus, however, is on a comparison between the two types



**Fig. 3.** Behavioral results for Experiments 1–3. Error bars represent the standard error of the mean (SEM). A. Push Sacrifice cases elicit worse performance overall but, uniquely, relative improvement following the first success. Pre-insight refers to all trials up to and including the first success on that problem type. Post-insight refers to all trials thereafter. B. Time pressure impaired performance especially for Push Sacrifice cases. C. Training experience improved scores on Switch Sacrifice cases but not on Push Sacrifice cases.

cases not represented in training. Specifically, we ask whether either of these classes, Switch Sacrifice and Push Sacrifice cases, reliably elicited failures, presumably because participants fail to even consider the highest-scoring solution in the first place. We assessed this in two ways. First, we simply measured overall levels of success vs. failure on these items. Obviously a case does not reliably elicit consideration failures if participants do not exhibit many failures at all.

Second, we asked whether participants showed evidence of a discontinuity in performance following their *first* success—in other words, having solved a particular class of problem once, do they become notably more likely to solve it correctly thereafter? This pattern would be consistent with the idea that once participants first appreciate the utility of the action in question (a sacrificing switch, or a sacrificing push), this candidate solution becomes more cognitively available in future problems of the same type.

### 2.1.1. Methods

After several pilot studies of the paradigm, we preregistered our experiment and analyses at aspredicted.org/GCL\_1KQ. Of 364 participants, 109 were excluded for failing to complete all assigned rounds or scoring less than zero points cumulatively in training. The rules of the game were provided in an extensive, interactive training procedure to all participants. All subjects completed 60 training rounds, followed by 50 test rounds. The 50 test rounds consisted of eight of each grid-type of interest (Push Sacrifice, Switch Sacrifice, Push Control, and Switch Control) and 18 filler grids drawn from the training distribution. All test grids were performed under a 7 s time delay to ensure full task engagement.

### 2.1.2. Results

Participants performed well on problems during the training phase, achieving an average accuracy of 77.3% (SEM = 0.5%, see SI figure for learning during training trials).

At test, participants performed optimally on most control cases of the types included in training (Push Control: M = 94.0%, SEM = 0.5%; Switch Control: M = 96.0%, SEM = 0.4%). Notably, they also did so similarly on the novel type of Switch Sacrifice cases (M = 94.5%, SEM = 0.5%). However, they were less likely to have optimally solved the novel type of Push Sacrifice cases (M = 62.5%, SEM = 1.1%). These data offer some preliminary support for the conclusion that Push Sacrifice cases uniquely elicit consideration failures, on the assumption that participants who successfully *consider* push option in these cases will take it, since it is the best option—and, thus, that failures to take this option stem for a failure to consider it in the first place. To provide further support for this inference, we fit a series of preregistered linear mixed models, separately for Switch Sacrifice cases and for Push Sacrifice cases, that predicted standardized scores on these test items with one term capturing linear improvement over the test block and a second term capturing a discontinuity in performance following the first correct answer. Put simply, these models ask: “Does the experience of solving the very first problem of a certain class increase the likelihood of solving subsequent ones?” The estimated

marginal means from the full model indicate that, after participants successfully solved their first Push Sacrifice case, they became 12.8% (95% CI [7.0%, 18.6%]) more likely to successfully solve subsequent Push Sacrifice cases. An ANOVA comparing a model with and without a term for discontinuity after first success revealed a significant effect ( $\chi^2(1) = 18.55, p < .001$ ). No such effect was evident in Switch Sacrifice cases,  $\chi^2(1) = 0.02, p = .89$ . In the full model for Switch Sacrifice cases, the estimated effect size was  $b = 0.00, 95\% \text{ CI} [-0.037, 0.032]$ . Additional details in SI, see Fig. 3A.

Why might Push Sacrifice cases uniquely elicit failures of consideration? We suggest that this may be because the problems presented in training generalize poorly to Push Sacrifice cases and bias consideration away from the optimal action sequence. One early hint to support this idea is that exposure to Push Sacrifice cases over the course of the test phase led to improved performance on this case type (see SI Fig S2). In an exploratory analysis, we find that performance on Push Sacrifice cases improves linearly with increasing experience over the course of the test phase. An ANOVA comparing a model with and without a term for linear improvement in training revealed a significant effect  $\chi^2(1) = 167.96, p < .001$ . The effect size for this term in the full model (including a term for linear improvement and a term for discontinuity after first success) was  $b = 0.034, 95\% \text{ CI} [0.024, 0.044]$ . The same is not true for Switch Sacrifice cases. An ANOVA comparing a model with and without a term for linear improvement in training did not reveal a significant effect,  $\chi^2(1) = .33, p = .57$ . The effect size for the linear improvement term in the full model was  $b = 0.00, 95\% \text{ CI} [-0.006, 0.004]$ . This is consistent with the idea that prior experience influences the solutions that later come to mind. We further explore the effect of training experience and provide evidence for this idea in Experiment 3.

In Experiment 1, we show that Push Sacrifice cases present a unique challenge for participants. It seems that, by default, participants often fail to consider solutions in which one uses a less costly collision to prevent a more costly one. Once they “see” the possibility of this solution, however, it becomes relatively more available in subsequent problems of the same type. Our next experiments further explore the possibility that Push Sacrifice cases elicit uniquely potent failures of consideration.

### 2.2. Experiment 2: Manipulating deliberation time

Logically, failures of consideration will arise the most often when the fewest number of items is considered. (After all, if *all* possible candidates are considered, no failure of consideration is possible—any failure would be due to the process of *evaluating* candidates). Prior research indicates that the more time people have to solve a problem, the more candidate solutions they evaluate, all else being equal (Morris et al., 2021). Thus, we reasoned that restricting participants’ time budget — and thus limiting the number of items they could consider — would selectively exacerbate failures of consideration.

Experiment 2 therefore extended the basic design of Experiment 1 to conditions of time pressure versus time delay. We reasoned that

suboptimal performance due to a failure of consideration would be sensitive to this manipulation. In contrast, alternative explanations for suboptimal performance (e.g., an unwillingness to “sacrifice” one object for another, or a misunderstanding of the rules or scoring of the task) would not be sensitive to this manipulation.

### 2.2.1. Methods

We preregistered our experiment and analyses at aspredicted.org/QTQ\_ESN. Of 200 participants, 64 were excluded for failing to complete all assigned rounds, timing out in more than six rounds in the “time pressure” condition, or scoring less than zero points in training. Participants were randomly assigned to the “time pressure” condition (limit of 7 s to complete each test round) or the “time delay” condition (wait 7 s before taking first action). Under time pressure, not completing the round in time results in a four point penalty.

### 2.2.2. Results

Again, participants performed well on problems during the training phase, achieving an average accuracy of 80.0% (SEM = 0.7%, see SI figure for learning during training trials).

At test, we analyzed the effect of time pressure versus time delay on performance for each of our test grid types of interest. Under time pressure, when we expect constraints on consideration to be most applicable, participants solved a smaller proportion of Push Sacrifice cases ( $M = 36.9\%$ ,  $SEM = 1.4\%$ ; see Fig. 3B) than Switch Sacrifice cases ( $M = 84.3\%$ ,  $SEM = 2.4\%$ ), Push Control cases ( $M = 90.2\%$ ,  $SEM = 2.0\%$ ), or Switch Control cases ( $M = 87.6\%$ ,  $SEM = 0.8\%$ ). Additionally, imposing a time delay prompted a greater improvement in performance on Push Sacrifice cases ( $M = 59.7\%$ ,  $SEM = 1.5\%$ ) than the remaining three types of cases (Switch Sacrifice:  $M = 96.3\%$ ,  $SEM = 1.7\%$ ; Push Control:  $M = 93.8\%$ ,  $SEM = 0.8\%$ ; Switch Control:  $M = 95.5\%$ ,  $SEM = 0.5\%$ ). This suggests that Push Sacrifice cases elicit uniquely large consideration failures in our task. We fit a series of preregistered linear mixed models that predicted standardized scores using the lme4 package in R to test several contrasts of interest (see SI for details). An ANOVA comparing a model with and without a term for time constraint revealed a significant effect across all grid types,  $\chi^2(1) = 20.45$ ,  $p < .001$ . In the full model including terms for time constraint, test grid type (Switch Sacrifice vs Push Sacrifice), and their interaction, the effect size for time constraint was  $b = 0.11$ , 95% CI [0.07, 0.16]. Model comparison additionally revealed significantly higher scores for Switch Sacrifice than for Push Sacrifice cases,  $\chi^2(1) = 20.03$ ,  $p < .001$  (effect size from full model,  $b = -0.42$ , 95% CI [-0.55, -0.29]), and a significant interaction such that the effect of time delay versus time pressure was greater for Push Sacrifice cases than for Switch Sacrifice cases  $\chi^2(1) = 6.65$ ,  $p = .01$  (effect size from full model,  $b = 0.11$ , 95% CI [0.03, 0.19]).

In summary, time pressure exerts a uniquely large effect on Push Sacrifice cases, supporting the conclusion that these are especially susceptible to failures of consideration.

## 2.3. Experiment 3: Manipulating training

Why would Push Sacrifice cases be especially likely to elicit failures of consideration, compared with Switch Sacrifice cases? Potentially, this occurs because of the way that participants generalize from training cases (which include neither Push Sacrifice nor Switch Sacrifice cases) to the novel test cases. During training, participants encounter cases in which switching a train away from cargo is the best action overall. (These cases differ from our test Switch Sacrifice cases because they do *not* involve sacrificing a low value piece of cargo for a high value piece). This may lead them to assign a high heuristic value estimate to the action of switching trains away from cargo, and thus lead them to often spontaneously consider switch actions in novel situations. As a result, failures of consideration are infrequent. In contrast, during training, participants encounter cases in which pushing cargo into the

path of a train is not the optimal action (These cases differ from our test Push Sacrifice cases because they do not use the destruction of a low-value piece of cargo to prevent the destruction of a higher-value piece). This may lead them to assign a low heuristic value estimate to the action of pushing cargo in front of trains, and thus lead them to rarely spontaneously consider push actions in novel situations. As a result, failures of consideration are more frequent.

Thus, Experiment 3 contrasted a “training” condition, where participants complete the training phase as normal, and a “no training” condition where participants complete the test phase directly after the task instructions. All participants completed the task under time pressure, where failures of consideration are most evident. We predicted that, while training might enhance performance overall by increasing task familiarity, it would do more to enhance performance on test Switch Sacrifice cases than on test Push Sacrifice cases. Indeed, in theory, it might even impair performance on test Push Sacrifice cases.

### 2.3.1. Methods

We preregistered our experiment and analyses at aspredicted.org/KL2\_CTC. Of 600 participants, 202 were excluded for failing to complete all assigned rounds, timing out more than six times, timing out more than twice on either test case type of interest, or scoring less than -35 points at test. Participants were randomly assigned to either the “training” or “no-training” condition. During the test phase, all subjects were placed under time pressure with a 7 s time limit. Each participant completed 24 test grids, consisting of three Push Sacrifice grids, three Switch Sacrifice grids, and twelve filler grids drawn from the training distribution. This reduction in number of test grids of interest was made in order to reduce the effect of learning from Push Sacrifice and Switch Sacrifice experiences during test.

### 2.3.2. Results

In Experiment 3, participants who were assigned to the training condition performed well on problems in the training phase, achieving an average accuracy of 71.8% (SEM = 0.9%, see SI figure for learning during training trials).

As expected, training had differential effects on performance for Switch Sacrifice and Push Sacrifice cases at test. The inclusion of training had little effect on participants’ scores in Push Sacrifice cases (with training:  $M = 17.4\%$ ,  $SEM = 1.8\%$ ; without training  $M = 15.6\%$ ,  $SEM = 2.3\%$ ) (Fig. 3C). For Switch Sacrifice cases, however, participants performed significantly worse without training ( $M = 53.1\%$ ,  $SEM = 1.5\%$ ), than with training ( $M = 66.0\%$ ,  $SEM = 1.9\%$ ). We fit a series of preregistered linear mixed models using the lme4 package in R modeling standardized scores by case type (Push Sacrifice versus Switch Sacrifice), training, and their interaction. Random effects in the final model included grid ID and participant ID. An ANOVA comparing a model with and without a term for training experience revealed a significant effect of training  $\chi^2(1) = 5.31$ ,  $p = .02$ . In the full model including terms for training experience, test grid type (Switch Sacrifice vs Push Sacrifice), and their interaction, the effect size for training experience was  $b = 0.07$ , 95% CI [0.02, 0.12]. Model comparison additionally revealed a significant effect of case type  $\chi^2(1) = 26.50$ ,  $p < .001$  (effect size from full model  $b = -0.43$ , 95% CI [-0.55, -0.31]), and a significant interaction between training experience and case type  $\chi^2(1) = 4.53$ ,  $p = .03$  (effect size from full model  $b = -0.11$ , 95% CI [-0.22, -0.01]).

The differential effect of training provides some indication that heuristic value estimates derived from training support *success* on switch trials, but do not support success for Push Sacrifice trials.

### 3. ML planning architecture and experiments

Next we explored whether similar patterns of performance arise if we attempt to solve our task using an architectural motif behind recent advances in machine decision-making in open-ended problems. Our goal was to adopt methods that are broadly representative of one family of current approaches, not to introduce *ad hoc* innovations. In other words, our aim is to ask, “what happens when model-based planning cannot evaluate all possible actions, and thus prioritizes search based on a heuristic estimate of value generalized from past experience?”

We expected that, like our human participants, models from this family would fall prey to failures of consideration on Push Sacrifice problems but not Switch Sacrifice problems. This is because we expect heuristic value estimation to *over-estimate* the value of switching actions (which saves points at no cost in training, but saves points at a cost in test), but to *under-estimate* the value of pushing actions (which costs points without benefit in training, but cost points with an overriding benefit in test). By overestimating the value of switching actions, subsequent model-based search will reliably evaluate the candidate switching actions and discovers they are optimal (albeit at a lower score than initially estimated). In contrast, by underestimating the value of pushing actions, subsequent model-based search will often fail to evaluate pushing actions, and therefore fail to discover that they are optimal (obtaining a higher score than initially estimated).

We begin by describing the architecture in more detail, and then report the results of *in silico* experiments designed to mirror those we performed on humans.

#### 3.1. Architectural details

Our architecture has two parts (Fig. 1): A neural net that approximates the state-action value function based on training, and a model-based method for improving these value estimates by Monte Carlo rollouts. The heuristic value estimates are used to prioritize tree search (i.e., rollouts), directing model-based exploration towards promising candidate actions.

##### 3.1.1. Approximating the value function with a trained neural network

We constructed a simple Convolutional Neural Network (CNN; Le-Cun, Bengio, & Hinton, 2015) that predicts the Q-values (Sutton & Barto, 1998) for each available action (up, right, down, left, and stay) given an input game state. The state input representation contains information about the time step (1–5), the locations of all objects in the grid, and the direction of the train. Put simply, the function of this CNN is to generate a heuristic estimate of the value of each action by generalizing from its experience during training. These value estimates are used to guide the search for promising actions (see the next section: *Action evaluation and control...*).

To train the CNN, we generated a set of potential inputs (200,000 grid problems from a fixed distribution) and outputs (optimal Q-values derived from value iteration). CNN training grids were drawn from the same distribution of problem types that was used to train participants in the behavioral task. Then, for each grid, we identified the optimal action sequence and associated set of states visited. These states, and the optimal Q-values for each action in these states, comprised the training inputs and outputs. (This parallels human experience during training, in which 80% of human-generated action sequences were optimal for those participants included in analysis.) See the supporting information text for more details about the CNN architecture and training.

#### 3.1.2. Action evaluation and control by first visit Monte Carlo

Our architecture next evaluated candidate actions by a standard model-based method: First-Visit Monte Carlo (FVMC) (Sutton & Barto, 1998). This algorithm refines its policy over a series of iterations, where each iteration involves simulating a full action sequence and updating Q-values based on the simulated returns. A random simulation strategy is shared by other Monte-Carlo methods, including Monte-Carlo tree search which was featured in AlphaGo (Silver et al., 2018). However, FVMC is best suited to our task, which has a fixed number of time steps and wider range of possible outcomes. First-visit Monte Carlo computes Q-values as the averaged returns following the first visit to a given state. In our task, it is not possible to visit the same state twice in one simulated trajectory, meaning that there is no meaningful distinction between first-visit and every-visit algorithms. Exploration for FVMC was determined by an  $\epsilon$ -greedy strategy with  $\epsilon = .2$ .

Crucially, we initialized the Q-values that guided FVMC using the estimates from our trained CNN. In this manner the CNN guided “consideration” of candidate actions, especially early in the process of model-based evaluation (See SI for details). Put simply, the function of the FVMC algorithm is to use its model of the task to determine the precise consequences of taking various sequences of actions, in order to guide choice. Determining *which* actions to evaluate, however, is guided by the CNN described in the previous section (*Approximating the value function...*).

#### 3.2. *In silico* experiments

Next we asked whether this architecture generates the same patterns of consideration failure observed in humans.

##### 3.2.1. Model failures of consideration

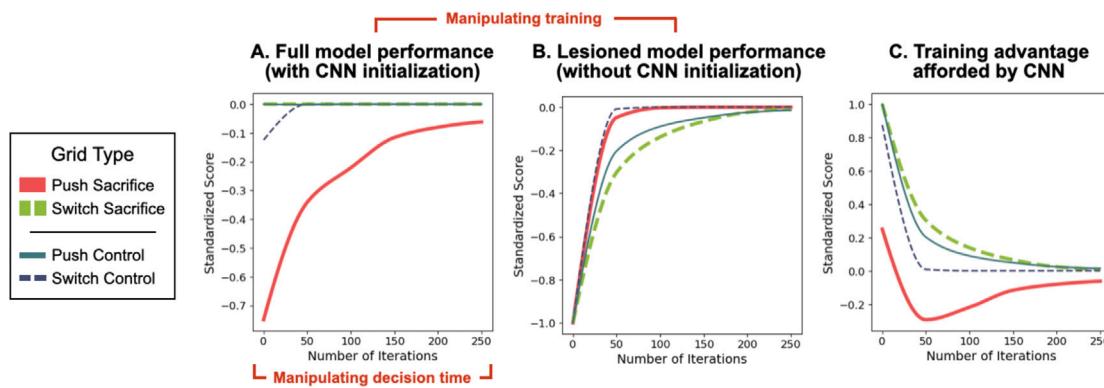
First, we tested our architecture on the same novel grids that we presented to humans. We predicted that, like humans in Experiment 1, it would exhibit especially poor performance on Push Sacrifice cases, failing to identify the optimal action, relative to Switch Sacrifice cases. As evident in Fig. 4A, the model consistently performs worse on Push Sacrifice cases compared to other problem types like our human participants.

Next, we fit our model to human data, comparing it with two alternative models designed to reflect alternative cognitive hypotheses. The first implements pure model-free control, in which value estimates from a trained CNN directly determine action selection. The second implements pure-model based control, in which a FVMC agent performs rollouts without biasing exploration towards actions with high estimated value. We compare these two models to our architecture in which model-free value estimates guide the evaluation of candidate action sequences.

We investigate how well each of these three models fits human behavior on Push Sacrifice and Switch Sacrifice cases in Experiment 1. To do so, we fit each model to the data and compare the resulting Bayesian Information Criterion (BIC) scores, which reflect the likelihood of the data under each model while penalizing for model complexity.

In order to fit the models to human data we optimized each model's free parameters at the group level, i.e., applying the same parameter settings to all participants in order to maximize the aggregate likelihood of the data. We adopt this approach because the relatively small number of observations per participant renders subject-level estimation vulnerable to overfitting.

All three models involve a conversion of action value estimates to choice probabilities by softmax with a free parameter  $\beta$  dictating the balance between exploitation and exploration. For the models featuring an FVMC component, we additionally fit the number of FVMC iterations to run. For the CNN-only candidate model, we use bounded local minimization to fit  $\beta$  (Brent, 1973; Grund, 1979). For our architecture and the FMVC-only candidate model, we concurrently fit the number



**Fig. 4.** In silico experiments with our model. The results in all figures are obtained by running each model at increments of 50 iterations, and averaging standardized scores achieved over 500 runs. This is done to compute a stable value for standardized score, as each Monte Carlo rollout occurs with some degree of randomness. Points falling between these increments are computed using PCHIP interpolation. Error bars are omitted as they are too small to be visible. Additionally, two specific problems (from the Push Control category) are omitted from these visualizations because the FVMC algorithm takes notably more iterations to solve them correctly. This is done to present a clearer visual comparison between problem types. However, we replicate the same pattern of model results when these are included (see SI Fig S3). A. Average standardized score of the ML architecture as a function of the number of FVMC iterations, for each test trial type. Analogous to human performance on Push Sacrifice cases under time pressure, ML performance on push cases is uniquely degraded at small numbers of iterations. B. Average standardized score of the ML architecture when removing the CNN component (no training experience). In the lesioned model, performance on all four grid types improves similarly with increased model-based rollouts. C. Performance advantage afforded by a trained CNN as a function of the number of FVMC iterations, for each test trial type. Values are computed as the difference between full model performance (4 A) and lesioned model performance (4B). Analogous to human results, training experience induces a large performance advantage especially at a small number of iterations, for Switch Sacrifice cases but not for Push Sacrifice cases. In fact, there is a performance decrement induced by training uniquely for Push Sacrifice cases. [Fritsch and Butland \(1984\)](#).

of FVMC iterations and the softmax temperature using Bayesian optimization with Gaussian processes ([The scikit-optimize contributors, 2018](#)). This optimization method was chosen because it is well suited for optimizing stochastic and computationally expensive functions. See SI for more details on parameter bounds and convergence.

Using this method, we find that our architecture (NLL = 23542.2, BIC = 47104.2,  $\beta$  = 0.2, FVMC iterations = 6405) better accounts for participant performance on Push Sacrifice and Switch Sacrifice test cases than the CNN-only (NLL = 26156.9, BIC = 52323.6,  $\beta$  = 0.4) and FVMC-only (NLL = 23629.5, BIC = 47278.9,  $\beta$  = 0.2, FVMC iterations = 35016) models. By convention, the BIC advantage exceeding 10 indicates strong evidence for our model compared with these alternatives.

### 3.2.2. Manipulating “decision time”

In Experiment 2, we manipulated people’s decision time to show that having more time to consider options improves performance more on Push Sacrifice cases compared to Switch Sacrifice cases. In our architecture, the “decision time” variable is analogous to the number of candidate action sequences that FVMC is allowed to consider before selecting the best-yet-considered as its final choice. In the language of FVMC, the key variable is the number of “iterations”. We therefore explored the effect of this parameter setting.

[Fig. 4A](#) shows the performance of our architecture on the four different types of test cases as a function of the number of iterations. As with humans under time pressure, when the number of iterations is small, the model performs much worse on Push Sacrifice cases than the other types of test case. As with humans under time delay, when the number of iterations increases, this disparity is reduced.

We performed a model comparison to provide further evidence for the correspondence between human thinking time and the number of FVMC iterations in our model. Here, we compared two versions of our proposed architecture: One that fits a single number of FVMC iterations to both time-pressure and time-delay conditions, and another that fits separate numbers of FVMC iterations to these two experimental conditions. For both models, we concurrently fit an additional free parameter for Softmax beta. As above, parameter estimation was conducted at the group level. We find that a model that allows for a different number of FVMC iterations in the time delay and time pressure conditions fits human data better than a model with the same number of FVMC iterations across conditions. The model that allows

FVMC iterations to differ across conditions (NLL = 13054.59, BIC = 26137.03,  $\beta$  = 0.23, Pressure FVMC iterations = 2822, Delay FVMC iterations = 3892) has a Bayesian Information Criteria (BIC) advantage of more than 10 compared with a model that does not (NLL = 13076.32, BIC = 26171.2,  $\beta$  = 0.2, FVMC iterations = 2900), indicating strong evidence by convention.

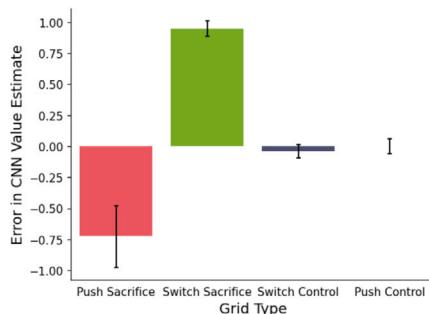
As predicted, in the better fitting model, the number of FVMC iterations that best fits human choices under time pressure is lower than that for human choices under time delay.

### 3.2.3. Manipulating training

Next, we asked how our architecture’s performance on the four types of test case varies as a function of training. In this context, the relevant training is encoded in the CNN that approximates the value function. We therefore ask how performance on test cases changes if the contribution of this CNN is removed. In the lesioned architecture, the state-action value estimates that guide FVMC are therefore uniformly initialized to zero. This manipulation of the architecture analogizes to Experiment 3, in which participants either received 60 training rounds or none. Of course, we assume that humans bring some relevant experience to the task that allows them to approximate a value function even in the absence of task-specific training. Thus, we might expect differences in performance without training and in the magnitude of the effect of training across humans versus AI.

Comparison of panels A and B of [Fig. 4](#) shows the differential effects of training on Switch Sacrifice and Push Sacrifice trial performance. In panel A, which shows performance (as a function of the number of iterations) for the trained model, we see the characteristic pattern where the model performs excellently on Switch Sacrifice cases and poorly on Push Sacrifice cases, especially with a low number of iterations. In contrast, Panel B shows the results with a lesioned model that does not benefit from any training. Here, we see that without CNN training, the lesioned model does not demonstrate failures of consideration on Push Sacrifice cases. This does not fully capture the behavior of our human participants without training, however. As we have noted, this discrepancy may be a result of the prior experience our human participants likely bring to the task.

To better visualize the “training advantage” (or disadvantage), we plot the difference in average standardized scores obtained by the full architecture and the lesioned one, as a function of both test case type and the number of FVMC iterations. This is shown in Panel C of [Fig. 4](#).



**Fig. 5.** An analysis of the CNN's value estimates for the key action of interest (pushing cargo or hitting the switch) for each test type. For the test types included in training (Push Control and Switch Control) value estimates are unbiased. For the test types not included in training value estimates are biased, but in opposite directions: Upwards for Switch Sacrifice cases, and downwards for Push Sacrifice cases.

4. We find that training generates a performance advantage on Switch Sacrifice, Switch Control, and Push Control cases. This aligns with the results of our human experiment, where training trials tended to improve performance on the same types of cases. In contrast, we find that training interferes with model performance on Push Sacrifice cases. This partially aligns with the results of our human experiment, where training trials provided no benefit in Push Sacrifice cases but did not interfere, either.

### 3.2.4. Interrogating value function generalization

We have proposed that both humans and our ML architecture fail to consider the optimal action in Push Sacrifice cases because their heuristic value estimates systematically underestimate the value of pushing an object into the path of a train. In the case of our ML architecture we can explore these value estimates directly. To do so, we computed the average signed error in value estimates generated by our trained CNN for the key action in each category of test case: Pushing an object into the path of the train (for Push Sacrifice cases), redirecting the train away from one object and towards another (for Switch Sacrifice cases), redirecting the train safely (for Switch Control cases), and pushing the object onto the target location (for Push Control cases). We did this for all of the test cases used in our human and *in silico* experiments (Fig. 5)

As expected, the CNN systematically underestimates the value of the optimal action in Push Sacrifice cases (a class on which it was not trained), while it exhibits no meaningful bias in Switch Control and Push Control cases (classes on which it was trained). Notably, however, the CNN also exhibits systematic error in its value estimate of the Switch Sacrifice cases. This is not unexpected, since Switch Sacrifice cases were not included in the training set. But, of course, we find that neither humans nor our ML architecture have much difficulty identifying the optimal solution for Switch Sacrifice cases. This is presumably because of the *sign* of the bias. In Switch Sacrifice cases value is overestimated (reflecting a training set in which switching the train often saves some cargo without causing harm to other cargo). This leads to model-based evaluation of the optimal action early, ensuring its proper consideration. In Push Sacrifice cases, value is instead underestimated (reflecting a training set in which pushing cargo into a train harms that cargo without saving any other cargo). This inhibits model-based evaluation of the optimal action, leading to a systematic failure of consideration.

## 4. Discussion

We generated a set of sequential decision-making problems to be solved by humans and an ML planning architecture. We find that the same class of problems that reliably generate failures of consideration

in humans also generates failures in our ML planner. This suggests an intriguing correspondence between the broad organizational architecture of human and machine approaches to decision-making in open-ended problems.

Three similarities characterized human and machine planners. First, both performed worse on Push Sacrifice than Switch Sacrifice cases. Second, this performance decrement was especially pronounced when time devoted to model-based evaluation was limited and, therefore, the initial value estimates used to determine consideration presumably played their greatest role. Third, training improved out-of-sample performance on Switch Sacrifice cases but failed to improve (in humans) or actually inhibited (in ML) out-of-sample performance on Push Sacrifice cases.

Interrogating our ML planning architecture, it is clear why Push Sacrifice cases elicited consideration failures. The architecture prioritizes evaluation of candidate actions that are assigned high value estimates by a neural net. The neural net must generalize from training to novel types of cases in which low-value cargo must be destroyed in order to save high-value cargo. We find that the neural net provides biased value estimates across both Switch Sacrifice cases and Push Sacrifice cases, but in opposite directions: It overestimates the value of switch actions, but underestimates the value of push actions. As a consequence, it quickly evaluates the switch actions and discovers that, while less valuable than expected, they are better than the alternatives. In contrast, it inhibits evaluation of push actions and therefore fails to discover that they, too, are better than the alternatives. The result is a cognitive puzzle specific to Push Sacrifice cases.

The qualitative similarities between human and ML performance suggest that they may employ broadly similar approaches to choice in this task: Consideration of candidate actions generated by a statistical approximation of the value-function trained on prior experience in related tasks, followed by refinement of those value estimates via model-based rollout. This is consistent with existing work on how people use value-based methods to solve certain open-ended problems (Hauser & Wernerfelt, 1990; Morris et al., 2021). Our findings suggest that participants do not rely exclusively on model-based evaluation, unguided by any heuristic estimate of the value of considering different sequences of actions. Such a model could not explain why participants are far more adept at finding the solution to switch cases than push cases, or why they show selective effects of training and learning for push cases. Moreover, when we fit participant behavior to an architecture that uses model-based methods only, it significantly underperforms the hybrid, two-stage architecture.

Alternatively, is it possible that when participants failed to select optimal sequence of actions, they actually considered those sequences, but underestimated their value based on a heuristic, model-free mechanism? Although possible, several factors suggest this is not likely. The structure of our task — simple, intuitive, and deterministic — means that the true (model-based) value of a sequence of actions is obvious as soon as you consider it. For instance, the optimal sequence in push cases avoids hitting the high value target, and no other sequence achieves this. It thus seems unlikely that a participant would consider this sequence of actions and *then* reject on the grounds that its model-free value is low. Further evidence against this possibility again comes from our model fitting: We find that a model-free-only architecture does worse than our proposed hybrid two-stage architecture. Of course, other architectures might be entertained, perhaps incorporating promising insights from prior research (Huys et al., 2012; Keramati, Smitenaar, Dolan, & Dayan, 2016), and these remain for future research to explore.

To the extent that human and ML performance align, we suggest that it is for two reasons: one about the structure of experience, and another about the structure of planning. First, like our model, humans experience many situations in which it makes sense to avoid collisions to prevent harm, and few situations in which it makes sense to cause collisions to prevent greater harm. Second, statistical approaches

to value estimation may be biased when generalizing to new case types and, when these heuristic value estimates are used to guide consideration, this will lead to planning failures.

Of course, there are several ways in which human performance likely diverges from our ML planning architecture in its details. We designed our CNN to be representative of standard approaches in contemporary machine learning, but we do not assume, nor does our data imply, any strong correspondence between the construction of the CNN and the neural mechanisms that humans use to approximate a value function in our task. Moreover, humans' heuristic value estimates are likely guided by a wealth of experience and structured knowledge (e.g., about trains, cargo, etc.) that go well beyond the specific instructions and training that we provided in the context of our experiment. Indeed, we know from existing work that insight can be guided not just by the kinds of value representations we interrogate here but also by semantic structures (Nedungadi & Hutchinson, 1985; Zhang et al., 2021) and, in particular, from conceptual restructuring (Batchelder & Alexander, 2012; Gilhooly & Murphy, 2005). It remains to future work to explore whether, and how, these more structured representations might be incorporated into the kind of ML framework employed here. A conceptual restructuring of this kind might play an important part in explaining why, once humans solved their first push-type case, they became relatively more likely to solve subsequent cases. (An effect of this kind was foreclosed in principle for our ML architecture, since no experiences during the test phase were used to retrain the CNN).

Our task resembles the famous "trolley problem" (Foot, 1967; Greene, 2014). Unlike the real trolley problem, however, in our "trolley puzzle" participants are unlikely to have conceived of cargo operations in moral terms, and our ML architecture was not designed for moral evaluation. Also, the trolley problem has not traditionally been used to demonstrate a failure to consider sacrificial harm, but instead the judgment that it is *wrong*. Nevertheless, the same kinds of mechanisms that furnish model-free value estimates guiding option consideration in our puzzle could potentially play a parallel role in certain forms of moral evaluation, as has been suggested elsewhere (Crockett, 2013; Cushman, 2013; Phillips & Cushman, 2017). This deserves further study.

Currently, humans are unparalleled in their capacity for intelligent, creative thought, and yet we are also prone to certain startling and predictable failures. Recent advances in ML have begun to carve away selected domains where human performance can be matched, or even exceeded. Presumably some ML architectures have close human parallels and others do not. Here, focusing on a case of apparent alignment, we show that human and machine planning is characterized not just by the same dazzling successes, but also by the same failures.

#### CRediT authorship contribution statement

**Alice Zhang:** Writing – review & editing, Writing – original draft, Software, Investigation, Formal analysis, Conceptualization. **Max Langenkamp:** Software, Investigation, Formal analysis, Conceptualization. **Max Kleiman-Weiner:** Conceptualization. **Tuomas Oikarinen:** Software, Investigation. **Fiery Cushman:** Writing – review & editing, Writing – original draft, Supervision, Funding acquisition, Formal analysis, Conceptualization.

#### Acknowledgments

This work was supported by grants N000141912205 and N000142212205 from the Office of Naval Research, United States.

#### Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.cognition.2025.106108>.

#### Data availability

All study materials, code, and analysis scripts are available at <https://osf.io/et89w>.

#### References

- Batchelder, William H., & Alexander, Gregory E. (2012). Insight problem solving: A critical examination of the possibility of formal theory. *The Journal of Problem Solving*, 5(1), 6.
- Bramley, Neil R., & Xu, Fei (2023). Active inductive inference in children and adults: A constructivist perspective. *Cognition*, 238, Article 105471.
- Brent, Richard P. (1973). *Algorithms for minimization without derivatives* (1st ed.). Englewood Cliffs, New Jersey: Prentice-Hall.
- Crockett, Molly J. (2013). Models of morality. *Trends in Cognitive Sciences*, 17(8), 363–366.
- Cushman, Fiery (2013). Action, outcome, and value: A dual-system framework for morality. *Personality and Social Psychology Review*, 17(3), 273–292.
- Dasgupta, Ishita, Schulz, Eric, & Gershman, Samuel J. (2017). Where do hypotheses come from? *Cognitive Psychology*, 96, 1–25.
- Dolan, Ray J., & Dayan, Peter (2013). Goals and habits in the brain. *Neuron*, 80(2), 312–325.
- Foot, Philippa (1967). The problem of abortion and the doctrine of the double effect.
- Fritsch, F. N., & Butland, J. (1984). A method for constructing local monotone piecewise cubic interpolants. *SIAM Journal on Scientific and Statistical Computing*, 5(2), 300–304. <http://dx.doi.org/10.1137/0905021>.
- Gilhooly, Kenneth J., & Murphy, P. (2005). Differentiating insight from non-insight problems. *Thinking & Reasoning*, 11(3), 279–302.
- Greene, Joshua (2014). *Moral tribes: Emotion, reason, and the gap between us and them*. Penguin.
- Grund, Friedrich (1979). Forsythe, G. E. / Malcolm, M. A. / Moler, C. B., Computer Methods for Mathematical Computations. Englewood Cliffs, New Jersey 07632. Prentice Hall, Inc., 1977. XI, 259 S. Zamm-Zeitschrift für Angewandte Mathematik und Mechanik, 59, 141–142, URL <https://api.semanticscholar.org/CorpusID:121678921>.
- Hauser, John R. (2014). Consideration-set heuristics. *Journal of Business Research*, 67(8), 1688–1699.
- Hauser, John R., & Wernerfelt, Birger (1990). An evaluation cost model of consideration sets. *Journal of Consumer Research*, 16(4), 393–408.
- Huys, Quentin J. M., Eshel, Neir, O'Nions, Elizabeth, Sheridan, Luke, Dayan, Peter, & Roiser, Jonathan P. (2012). Bonsai trees in your head: how the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS Computational Biology*, 8(3), Article e1002410.
- Johnson, Joseph G., & Raab, Markus (2003). Take the first: Option-generation and resulting choices. *Organizational Behavior and Human Decision Processes*, 91(2), 215–229.
- Kaiser, Stefan, Simon, Joe J., Kalis, Annemarie, Schweizer, Sophie, Tobler, Philippe N., & Mojzisch, Andreas (2013). The cognitive and neural basis of option generation and subsequent choice. *Cognitive, Affective, & Behavioral Neuroscience*, 13(4), 814–829.
- Kalis, Annemarie, Kaiser, Stefan, & Mojzisch, Andreas (2013). Why we should talk about option generation in decision-making research. *Frontiers in Psychology*, 4, 555.
- Keramati, Mehdi, Smitenaar, Peter, Dolan, Raymond J., & Dayan, Peter (2016). Adaptive integration of habits into depth-limited planning defines a habitual-goal-directed spectrum. *Proceedings of the National Academy of Sciences*, 113(45), 12868–12873.
- Klein, Gary A. (1993). A recognition-primed decision (RPD) model of rapid decision making. In J. Klein, R. Orasanu, C. Calderwood, & C. E. Zsambok (Eds.), *Decision making in action: models and methods* (pp. 138–147). New York: Ablex Publishing Corporation.
- LeCun, Yann, Bengio, Yoshua, & Hinton, Geoffrey (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Lieder, Falk, Griffiths, Thomas L., & Hsu, Ming (2018). Overrepresentation of extreme events in decision making reflects rational use of cognitive resources.. *Psychological Review*, 125(1), 1.
- Morris, Adam, Phillips, Jonathan, Huang, Karen, & Cushman, Fiery (2021). Generating options and choosing between them depend on distinct forms of value representation. *Psychological Science*, 32(11), 1731–1746.
- Nedungadi, Prakash, & Hutchinson, W. (1985). The prototypicality of brands: Relationships with brand awareness, preference and usage. *Advances in Consumer Research*, 12, 489–503.
- Peters, Sarah L., Fellows, Lesley K., & Sheldon, Signy (2017). The ventromedial frontal lobe contributes to forming effective solutions to real-world problems. *Journal of Cognitive Neuroscience*, 29(6), 991–1001.
- Phillips, Jonathan, & Cushman, Fiery (2017). Morality constrains the default representation of what is possible. *Proceedings of the National Academy of Sciences*, 114(18), 4649–4654.
- Phillips, Jonathan, Morris, Adam, & Cushman, Fiery (2019). How we know what not to think. *Trends in Cognitive Sciences*, 23(12), 1026–1040.

- Saxe, Rebecca (2005). Against simulation: the argument from error. *Trends in Cognitive Sciences*, 9(4), 174–179.
- Silver, David, Hubert, Thomas, Schrittwieser, Julian, Antonoglou, Ioannis, Lai, Matthew, Guez, Arthur, et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419), 1140–1144.
- Simon, Herbert A. (1955). A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1), 99–118.
- Sutton, Richard S., & Barto, Andrew G. (1998). *Reinforcement learning: An introduction* (1 ed.). Cambridge: MIT Press.
- The scikit-optimize contributors (2018). Scikit-optimize: Bayesian optimization for machine learning. <https://scikit-optimize.github.io>.
- Ullman, Tomer D., Goodman, Noah D., & Tenenbaum, Joshua B. (2012). Theory learning as stochastic search in the language of thought. *Cognitive Development*, 27(4), 455–480.
- Wason, Peter C. (1968). Reasoning about a rule. *Quarterly Journal of Experimental Psychology*, 20(3), 273–281.
- Wolpert, David H., & Macready, William G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.
- Zhang, Zhihao, Wang, Shichun, Good, Maxwell, Hristova, Siyana, Kayser, Andrew S., & Hsu, Ming (2021). Retrieval-constrained valuation: Toward prediction of open-ended decisions. *Proceedings of the National Academy of Sciences*, 118(20), Article e2022685118.
- Zhao, Bonan, Lucas, Christopher G., & Bramley, Neil R. (2024). A model of conceptual bootstrapping in human cognition. *Nature Human Behaviour*, 8, 125–136.