Improving Human Decision-Making with Machine Learning

Anonymous Authors[1]

Abstract

A key aspect of human intelligence is their ability to convey their knowledge to others in succinct forms. However, current machine learning models are largely blackboxes that are hard for humans to learn from. We study the problem of whether we can design machine learning algorithms capable of conveying their insights to humans in the context of a sequential decision making task. In particular, we propose a novel machine learning algorithm for extracting interpretable tips from a policy trained to solve the task using reinforcement learning. In particular, it searches over a space of interpretable decision rules to identify the one that most improves human performance. Then, we perform an extensive user study to evaluate our approach, based on a virtual kitchen-management game we designed that requires the participant to make a series of decisions to minimize overall service time. Our experiments show that (i) the tips generated by our algorithm are effective at improving performance, (ii) they significantly outperform the two baseline tips, and (iii) they successfully help participants build on their own experience to discover additional strategies and overcome their resistance to exploring counterintuitive strategies.

1. Introduction

A hallmark of human intelligence is our ability to distill knowledge and expertise into clear and simple representations that can be communicated to others. This ability is critical for human progress, since even simple insights often require a great deal of costly experience and experimentation to discover – for instance, Newton's laws of motion span a few lines of text, yet represent the culmination of millennia of scientific thinking. As a more commonplace example, humans naturally pass on their knowledge to help others – for instance, doctors regularly communicate best practices to their less experienced peers to help them improve their performance (Song et al., 2017). While often succinct, these best practices typically require a great deal of trial and error to discover. By sharing our insights with others, we avoid repeatedly reinventing concepts and instead focus our efforts on expanding the sum of human knowledge.

While recent advances in deep learning have enabled machines to achieve human-level or super-human performance at many artificial intelligence tasks (Mnih et al., 2015; Silver et al., 2016), the knowledge behind their abilities are hidden within the millions of deep neural network parameters. As a consequence, despite enormous progress, deep learning has remained a tool for pattern matching and prediction rather than one for augmenting human understanding. One of the holy grails of machine learning is to develop techniques for distilling the knowledge represented by the millions of parameters in a deep neural network into compact representations can be understood by humans. Indeed, recent work on interpretable machine learning has strived to learn structured models where the computation is transparent to the user, such as decision trees, rule lists, and sparse linear models (Friedman et al., 2008; Letham et al., 2015; Yang et al., 2017). Intuitively, there often exist interpretable models that achieve performance similar to their deep neural network counterparts; instead, the challenge lies in discovering these high-performing models.

A natural question arises: can we leverage these techniques to learn representations of knowledge that are useful to humans in some way? The goal of our work is to address this question. In particular, we devise a novel algorithm for inferring simple tips that can be used to improve performance of a human user. We focus on settings where the human must make a sequence of decisions to optimize an outcome in a complex environment. These settings are particularly challenging for humans, since they must trade off short-term and long-term rewards in ways that are often opaque. Our algorithm infers a tip that best bridges the gap between the baseline performance of human users and a deep reinforcement learning model. In other words, our algorithm condenses the deep neural network into a simple tip focused on conveying insights that are novel to the human users.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

We perform an extensive user study based on a scenario where human users are asked to play a game where they manage a virtual kitchen; an illustration of this task is shown in Figure 1. In this game, the users must assign subtasks to virtual workers with varying capabilities in a way that optimizes the time it takes to complete a set of food orders. This game is challenging because the human users must make tradeoffs such as deciding whether to greedily assign a worker to a subtask that they are slow to complete, or leave them idle in anticipation of a more suitable subtask.

We demonstrate that our algorithm can generate novel insights that enable human users to substantially improve their performance compared to counterparts that are not shown the tip. Interestingly, the users do not merely adjust their actions by blindly following the tip. Instead, as they gain experience with the game, they come to understand the significance of the tip and improve their performance in ways beyond the surface-level meaning of the tip. Our findings suggest that our algorithm infers interpretable and useful insights about the underlying task to the human users.

Related work. There has been a great deal of recent interest in interpretable machine learning (Caruana et al., 2015; Letham et al., 2015; Ribeiro et al., 2016; Doshi-Velez & Kim, 2017). Most of this work has focused on whether a human expert can understand why the model is making a certain prediction, in tasks such as patient diagnosis (Caruana et al., 2015) or image and text classification (Ribeiro et al., 2016). There has also been recent work seeking to understand how humans interpret predictions of machine learning models (Narayanan et al., 2018), as well as use measures of human understanding to measure and improve interpretability (Lage et al., 2018). Finally, there has been recent work on interpretable reinforcement learning (Verma et al., 2018), which can learn policies that are easier to verify (Bastani et al., 2018), generalize better (Inala et al., 2021), or satisfy combinatorial constraints (Inala et al., 2019). In contrast to prior work, our goal is to understand whether machine learning algorithms can communicate insights to improve human performance at complex sequential decision-making tasks.

2. Inferring Tips via Interpretable Reinforcement Learning

Consider a human making a sequence of decisions to achieve some desired outcome. We study settings where current decisions affect future outcomes—for instance, if the human decides to consume some resources at the current time step, they can no longer use these resources in the future. These settings are particularly challenging for decision-making due to the need to reason about how current actions affect future decisions, making them ideal targets for leveraging tips to improve human performance. In particular, our goal is to provide insights to the human that enable them to improve their performance.

We begin by formalizing the tip inference problem. We model our setting as the human acting to maximize reward in a Markov Decision Process (MDP) $M = (S; A; R; P; \ )$ over a finite time horizon T. Here, S is the state space, A is the action space, R is the reward function, and P is the transition function. Intuitively, a state $s \in S$ captures the current configuration of the system (e.g., available resources), and an action $a \in A$ is a decision that the human can make (e.g., consume some resources to produce an item). More precisely, we represent the human as a decision-making policy $_H$ mapping states to (possibly random) actions. At each time step $t \in \{1; \ldots; T\}$, the human observes the current state s_t and selects an action a_t to take according to the probability distribution $p(a_t \mid s_t) = \ _H(s_t; a_t)$. Then, they receive reward $r_t = R(s; a)$, and the system transitions to the next state s_{t+1}, which is a random variable with probability distribution $p(s_{t+1} \mid s_t; a_t) = P(s_t; a_t; s_{t+1})$, after which the process is repeated until $t = T$. A sequence of state-action-reward triples sampled according to this process is called a *rollout*, denoted $\ = ((s_1; a_1; r_1); \ldots; (s_T; a_T; r_T))$. The human's goal is to act according to a policy $_H$ that maximizes the cumulative expected reward $J(\ _H)$, where

$$J(\ ) = \mathbb{E}_{\ D(\ )}\left[\sum_{t=1}^{\ } {}^{t} r_t\right];$$

and where $D(\ )$ is the distribution of rollouts induced by using policy $\ $.

Now, given the MDP M along with the human policy $_H$, our goal is to learn a tip $\ $ that most improves the cumulative expected reward. Formally, a tip indicates that in certain states s, the human should use action $\ (s) \in A$ instead of their own policy $_H$. For simplicity, we assume that the human always follows the tip; while this assumption does not always hold in practice, we find that it works well as long as the human can understand the tip along with its rationale. Thus, we consider tips in the form of a single, interpretable rule:

$$(s) = \text{if} \quad (s) \text{ then take action } a;$$

where $a \in A$ is an action and $\ (s) \in \{\text{true}; \text{false}\}$ is a logical predicate over states $s \in S$—e.g., it might say that a sufficient quantity of a certain resource is currently available. Intuitively, a tip $\ = (\ ; a)$ says that if the condition $\ $ is satisfied, then the human should use action a; otherwise, they should use their default action $_H(s)$. Assuming the human follows this tip exactly, then the resulting policy they use is $_H \ \ $, where

$$(\ \ )(s; a) = \begin{cases} \mathbb{1}(a = a^{0}) & \text{if} \quad (s) \\ (s; a) & \text{otherwise:} \end{cases}$$

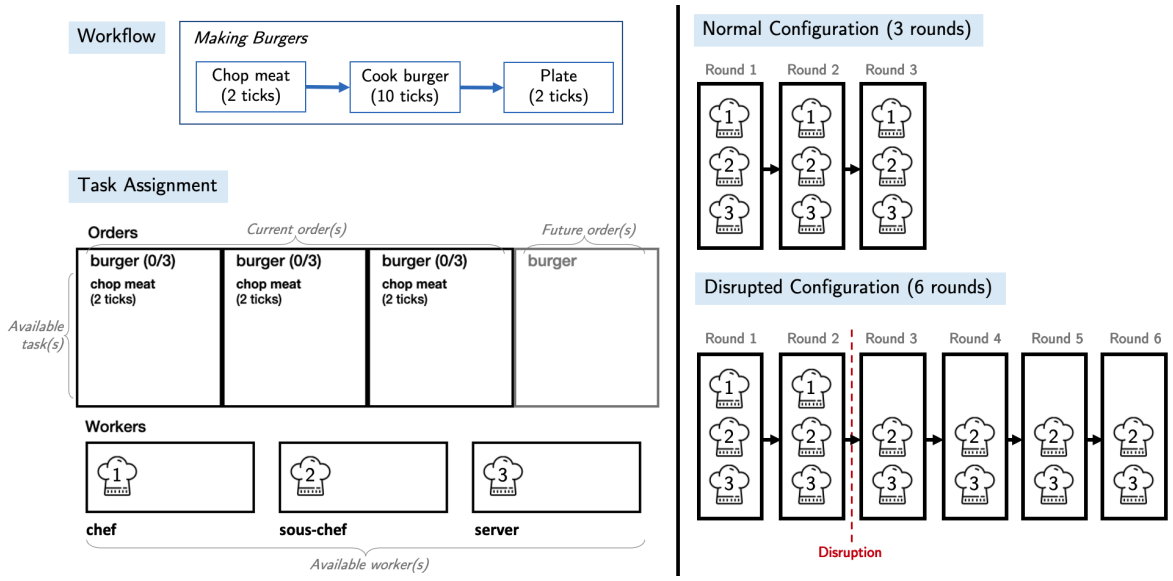


Figure 1: Overview of behavioral study: virtual kitchen management.

Then, our goal is to compute the tip that most improves the human's performance—i.e.,

$$= \arg\max J(\quad _H \quad): \tag{1}$$

The tip inference problem is to compute .

Next, we describe our algorithm for solving this problem. To guide our algorithm, we first compute the optimal Q-function Q by (approximately) solving the Bellman equations

$$Q \ (s; a) = R(s) + \quad \mathsf{E}_{a \quad (s; \ ); s^0 \quad P(s;a; \ )}[Q \ (s; a)]$$

using Q-learning (Watkins & Dayan, 1992), where we parameterize Q using a neural network with a single hidden layer. Then, we can rewrite the objective $J(\quad _H \quad)$ in (1) as follows (Bastani et al., 2018):

$$J(\quad _H \quad) = \mathsf{E}_{\quad D(\quad _H \quad)} \left[\sum_{t=1} Q \ (s_t; a_t) \ : \right]$$

Since we do not have access to samples $D(\quad _H \quad)$, we use the approximation using $D(\quad _H \quad)$ $D(\quad _H)$, which is good as long as the tip does not drastically change the human's decisions. Furthermore, we approximate the expectation in our objective using rollouts $_1; \ldots; \ _k$ $D(\quad _H)$ from the human policy $_H$, where $_i = ((s_{i;1}; a_{i;1}; r_{i;1}); \ldots; (s_{i;T}; a_{i;T}; r_{i;T}))$. Thus, our algorithm computes the tip

$$\hat{} = \arg\max \frac{1}{k} \sum_{i=1} \sum_{t=1} Q \ (s_{i;t}; (a_{i;t} \quad)(s_{i;t})); \tag{2}$$

where for a given tip $= (\ ; a)$ and action a^0, we have

$$(a^0 \quad)(s) = \begin{cases} a & \text{if} \quad (s) = 1 \\ a^0 & \text{otherwise.} \end{cases}$$

We optimize (2) by enumerating through candidate tips , evaluating the objective, and selecting the tip $\hat{}$ with the highest objective value.

3. Case Study: Kitchen-Management Game

We have developed a sequential decision-making task in the form of a virtual kitchen-management game that can be played by individual human users. In this game, the user takes the role of a manager of several virtual workers (namely, chef, sous-chef, and server) producing food orders (all burgers) in a virtual kitchen. Each order itself consists of a fixed set of subtasks (namely, chop meat, cook burger, or plate burger). The game consists of a fixed number of steps; on each step, the user must decide which (if any) subtask to assign to an idle worker. The worker becomes busy for a number of subsequent time steps, after which the subtask is completed and the worker becomes idle again. An order is completed once all its subtasks are completed, and the game is complete once all orders are completed. The user's goal is to complete the game in as few time steps as possible.

There are two key aspects of the game that make it challenging for the human to perform well. First, the subtasks have dependencies—i.e., some subtasks of an order can only be assigned to a worker once other subtasks of the same order have already been completed. In particular, "cook burger" can only be assigned once "chop meat" is completed, and "plate burger" can only be assigned once "cook burger" is

completed. Second, the virtual workers are heterogeneous—i.e., different workers take different numbers of time steps to complete different subtasks. In particular, the chef chops and cooks more quickly but plates more slowly than the server, and the sous-chef performs between the chef and server on all subtasks. As a consequence, the user faces the following dilemma. Suppose a worker becomes idle, but there is no available subtask that that worker completes quickly. Then, the user must either assign a suboptimal subtask to that worker, or leave the worker idle until such a subtask to become available. For instance, if the server is idle but all available subtasks are "cook burger", then the user must either assign "cook burger" to the server, or leave the server idle until a "serve burger" task becomes available. Furthermore, participants are not shown the exact number of steps each worker takes to complete each subtask, only the median number of steps taken. The true number of steps for a given worker is only revealed if they assign the subtask to that worker. Thus, the user must experiment to learn this information.

Finally, our experimental design is based on two scenarios of the virtual kitchen, differing in terms of which workers are available. First, in the fully-staffed scenario, the human user has access to all three virtual workers—i.e., the chef, sous-chef, and server. Second, in the understaffed scenario, the user has access to only two virtual workers—in particular, the sous-chef and server. The scenarios are otherwise identical.

4. Experiments

We study how humans interpret and follow the tips inferred by our algorithm in pre-registered behavioral experiments involving Amazon Mechanical Turk (AMT) users[1]. We show that our tips significantly improve participant performance compared to the control group, as well as compared to two baselines: (i) a baseline algorithm that naïvely tries to match the optimal policy, and (ii) tips suggested by previous participants. These results demonstrate that despite their simplicity and conciseness, our tips capture strategies that are hard for participants to learn and can significantly improve their performance. In addition, we find evidence that the participants were not blindly following our tips, but combine them with their own experience to improve performance. Finally, we also find evidence that participants build on our tips by discovering additional strategies beyond the ones stated in the tips. We describe our results below; experimental procedures as well as examples of tips and other additional results are provided in the Appendix.

[1]The full pre-registration document for our study is available at https://aspredicted.org/blind.php?x=8ye5cb

4.1. Experimental Methodology

We compare to two baselines and a control group. Our first baseline is a naïve algorithm that, given rollouts $\hat{D}$ from the optimal policy , computes the frequency $C(s;a)$ of state-action pairs in $\hat{D}$, and then selects

$$\hat{a}_{\text{bl}} = \arg\max_{a} \frac{1}{k} \sum_{i=1}^{k} \sum_{t=1}^{T} C(s_{i;t}; (a_{i;t} \quad)(s_{i;t})): \quad (3)$$

Intuitively, this objective ignores the structure of the MDP implicitly encoded in the Q-function, and instead directly tries to imitates the optimal policy. For our second baseline, we show each participant a comprehensive list of candidate tips and ask them to select the one they believe would most improve the performance of future players.

To evaluate a given tip, we have participants play our game, showing them the tip for the entire duration of the game. We have them play a sequence of scenarios (called a *configuration*) to study how their performance evolves over time. There are two configurations. In the normal configuration ($N = 1;317$ participants), each participant simply plays three rounds of the fully-staffed scenario; in this case, the participant is shown the same tip on all rounds. In the disrupted configuration ($N = 1;011$ participants), each participant plays two rounds of the fully-staffed scenario and four rounds of the understaffed scenario; in this case, the participant is shown a tip tailored to the scenario on the current round.

The disrupted configuration is designed to show how tips can help participants adapt to novel situations where the optimal strategy substantially changes. In particular, in the fully-staffed scenario, the optimal strategy is not to assign chopping or cooking tasks to the server. In contrast, in the understaffed scenario, this strategy often leaves the server idle; instead, the optimal strategy is to have the server cook twice (with the sous-chef handling the remaining two cooking tasks).

We assess performance by comparing the completion time in the final round; we also examine the fraction of participants that reached the optimal reward (20 steps for fully-staffed and 34 steps for understaffed).

4.2. User Performance

Overall, our algorithm was the most successful at improving performance. In the normal configuration, participants shown our tip completed the final round in 22.54 steps on average (optimal is 20 steps), significantly outperforming those in other arms: 23.86 (control group, $t(329) = 4{:}397$, $p < 0{:}0001$), 23.73 (human tip, $t(312) = 3{:}628$, $p = 0{:}0002$), and 23.82 (naïve algorithm, $t(334) = 4{:}232$, $p < 0{:}0001$). In general, a substantial fraction of participants learned to play the game optimally after three rounds.

((a)) Normal con guration (optimal: 20 steps) ((b)) Disrupted con guration (optimal: 34 steps)

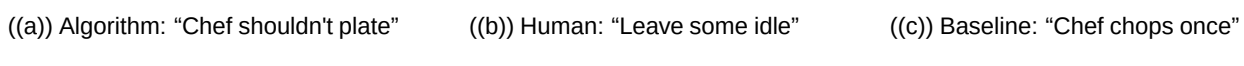
Figure 2: Performance of participants across conditions in the last round.

((a)) Normal con guration ((b)) Disrupted con guration

Figure 3: Performance over time.

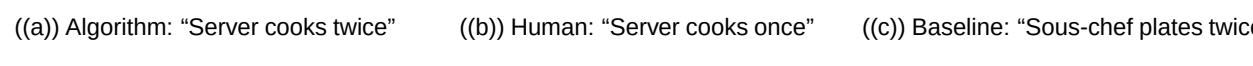
34.59% of those in our algorithm arm achieved optimal performance in the nal round, while 24.44% to 28.90% of the other groups could do the same. In the disrupted con guration, participants shown our tip completed the nal rounds in 37.05 steps, again signi cantly outperforming those in other arms: 37.92 (control group, $t(243) = -4.361$, $p < 0.0001$), 37.53 (human tip, $t(246) = -2.52$, $p = 0.0061$), and 38.40 (naïve algorithm, $t(246) = -7.348$, $p < 0.0001$). We found similar results for the optimal performance rate. Only participants given our tip achieved optimal performance 18.79% of participants in our algorithm arm played optimally, compared to 1.14% of the human tip arm, 0.99% of the naïve algorithm arm, and 0.51% of the control arm. As there were no signi cant differences in performance across treatments when playing the initial fully-staffed rounds, remarkably poor performance outside of our algorithm arm re ects the dif culty of managing the understaffed scenario.

Our results also help us understand the reasons behind the success of our tip. Intuitively, the naïve algorithm performs poorly since it blindly tries to mimic the optimal policy rather than focus on accounting for consequential decisions. For instance, in the disrupted con guration, it infers the tip "Sous-chef should plate twice", which actually reduces performance. The actions suggested by this tip occur at the end of the game, which is too late to signi cantly bene t overall performance. In contrast, our algorithm focuses on mimicking decisions made by the optimal policy that have long-term bene ts. Finally, while the human arm performs better than the control group, it could not reach the performance of our tip. The human suggested tips suffer from humans' inability to translate their strategy into a concise and speci c tip and to adapt their strategy to disruption. In the normal con guration, the human suggested tip "Strategically leave some workers idle", while capturing the key strategy, is more shallow and less speci c than other tips, leaving participants to gure out the actual strategy. In the disrupted con guration, the human suggested tip "Server should cook once" indicates that humans were starting to ad-



((a)) Algorithm: “Chef shouldn't plate” ((b)) Human: “Leave some idle” ((c)) Baseline: “Chef chops once”

Figure 4: Cross-compliance rate across the rounds (normal con guration).



((a)) Algorithm: “Server cooks twice” ((b)) Human: “Server cooks once” ((c)) Baseline: “Sous-chef plates twice”

Figure 5: Cross-compliance rate across the rounds (disrupted con guration).

just their strategy but it is not aggressive enough to achieve the optimal performance—as indicated by our tip (“Server should cook twice”). In optimal play, the server actually needs to perform a signi cantly larger share of the subtasks than the human tip suggests. Thus, the success of our tip is due in part to how it offers speci c advice that helps humans adapt to disruption and exploring potentially counterintuitive strategies.

4.3. Learning Over Time

Next, we study how performance improves across rounds as participants learn better strategies. In particular, tips can be thought of as a substitute for learning, reducing the number of rounds needed for participants to achieve a certain performance level. In the normal con guration (Figure 3(a)), our tip speeds up learning by at least one round compared to any of the other arms—i.e., the performance of participants given our tip on the k-th round was similar to the performance of participants in alternative arms on the $(k-1)$-th round. The improved speed of learning was even more apparent under the disrupted con guration: participants in the control group took four rounds to achieve the same level of performance as those provided with our tip on the rst round. Interestingly, the human arm initially improves performance comparably to our arm (see Figure 3(b)); however, it levels off towards the end whereas our arm continues to improve. These results suggest that our tip, while simple and concise, encodes a complex underlying strategy that the participants come to understand when they combine it with their own experience playing the game. In contrast, the human-suggested tip encodes a more shallow strategy that quickly improves performance but does not lead to deeper insight over time.

4.4. Learning Beyond Tips

A key question is understanding how humans internalized and actualized the strategies encoded in the tips we inferred. We study this question in two ways. First, we examine cross-compliance, which is the compliance of the participant to alternative tips other than the one we showed them. Naïvely, there is no reason to expect participants to comply with an alternative tip (assuming it does not overlap with the tip shown), beyond the cross-compliance of the control group to that tip. Thus, cross-compliance measures how showing one tip can enable participants to discover strategies beyond what is stated in that tip. Finally, we investigate whether the tips could help participants uncover the structure of the optimal policy beyond the simple rules they stated.

Cross-compliance. We nd that our tips had high cross-compliance than other tips in both con gurations. For the normal con guration, we nd that participants across all

((a)) Fully-staffed: “Server shouldn't cook” ((b)) Understaffed: “Server chops once”

Figure 6: Fraction of participants taking optimal action beyond the tips they were shown across the rounds.

arms learn not to assign plating to the chef (Figure 4(a)), strategically leave some virtual workers idle (Figure 4(b)), and let the chef chop only once (Figure 4(c)). These tips are all consistent with the optimal policy, suggesting that participants generally learn over time to improve their performance regardless of the treatment. Interestingly, participants in the algorithm arm have similar or higher cross-compliance compared to the other arms. This result suggests that our tip is the most effective as the information it encompasses the information conveyed by the other tips.

For the disrupted con guration, the cross-compliance of the the human and control arms with our tip remained at (Figure 5(a)). We observe a similar trend for the algorithm tip (Figure 5(c)). These results suggest that participants do not naturally learn this strategy over time, most likely since it is counterintuitive. In particular, simple tips can greatly improve human performance by capturing counterintuitive strategies that take a great deal of experimentation to discover. Finally, Figure 5(b) shows the cross-compliance to the human tip. Interestingly, compliance of the algorithm arm to this strategy actually decreases over time; indeed, the tip suggested by humans is a suboptimal strategy, so complying with this tip leads to worse performance.

Uncovering the optimal policy. At a high level, the optimal policy for the fully-staffed scenario has the chef cook most of the dishes, has the server plate most of the dishes, and never assigns the chef to plate or the server to cook. We observe that participants generally recovered these optimal strategies as they played more rounds. For instance, the fraction of participants in each arm that never assigned cooking to the server in each round, as if they were following the tip “Server shouldn't cook”, increases over time and within each round the fractions are not statistically different among the arms (see Figure 6(a)). This result suggests that people learned about this rule by themselves across all arms.

Furthermore, for the understaffed scenario, the optimal policy requires balanced assignment of cooking and plating tasks to both sous-chef and server while assigning most of the chopping tasks to sous-chef. A key rule beyond our tip is thus “Server chops once”. We nd that only participants in our arm cross-complied with this auxiliary tip (see Figure 6(b)), again demonstrating that our tip enables participants to discover strategies beyond what is stated in the tip.

5. Conclusion

We have proposed a novel machine-learning algorithm for automatically identifying interpretable tips designed to help improve human decision-making. Our behavioral study demonstrates that the tips inferred using our algorithm can successfully improve human performance at a challenging sequential decision-making task. In particular, our results suggest that our tip can speed up learning by up to three rounds of in-game experience, demonstrating that our tip can signi cantly reduce the cost of learning. Furthermore, in the presence of in-game disruption, our results suggest that our tip enables the participants to discover additional strategies beyond the given tip. In other words, the bene t of tips comes not just from having the human follow the letter of the tip, but from how the human builds on the tip to discover additional insights.

There are several important directions for future work. First, personalizing tips to individual workers could greatly improve the performance of our tip inference algorithm—ideally, we would infer tips personalized for different skill levels and individual worker characteristics. Next, in our approach, we only inferred tips at one point in time. In practice, our approach could also be performed every time additional data is collected, which would enable us to better understand the long-term bene ts of our approach and understanding how it affects learning behavior over a longer

period of time. Another promising direction is extending our algorithm to a collaborative setting. We have only studied how individual workers learn to improve performance, but a similar approach may help teams improve their collaboration and optimize information sharing. Finally, future work is needed to study how to better convey machine-generated tips to improve compliance. Recent work has documented human aversion to advice made by algorithms (Eastwood et al., 2012; Dietvorst et al., 2015) and shown certain conditions that alleviate such aversion (Dietvorst et al., 2018; Logg et al., 2019). In our study, a fraction of participants chose to forgo our tip and continue using their own strategy. Finding ways to build trust and encourage compliance is an important ingredient for ensuring our tips help people improve. Our empirical results suggest that following the tip depends not just on whether the user trusts the algorithm, but on whether the user understands why the proposed tip is reasonable. This insight brings a novel dimension to human trust in arti cially intelligent systems.

References

Bastani, O., Pu, Y., and Solar-Lezama, A. Veri able reinforcement learning via policy extraction. In *Advances in neural information processing systems*, pp. 2494–2504, 2018.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1721–1730, 2015.

Dietvorst, B. J., Simmons, J. P., and Massey, C. Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General*, 144(1):114, 2015.

Dietvorst, B. J., Simmons, J. P., and Massey, C. Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them. *Management Science*, 64(3):1155–1170, 2018.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Eastwood, J., Snook, B., and Luther, K. What people want from their professionals: Attitudes toward decision-making strategies. *Journal of Behavioral Decision Making*, 25(5):458–468, 2012.

Friedman, J. H., Popescu, B. E., et al. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2(3): 916–954, 2008.

Inala, J. P., Bastani, O., Tavares, Z., and Solar-Lezama, A. Synthesizing programmatic policies that inductively generalize. In *International Conference on Learning Representations*, 2019.

Inala, J. P., Yang, Y., Paulos, J., Pu, Y., Bastani, O., Kumar, V., Rinard, M., and Solar-Lezama, A. Neurosymbolic transformers for multi-agent communication. *arXiv preprint arXiv:2101.03238*, 2021.

Lage, I., Ross, A. S., Kim, B., Gershman, S. J., and Doshi-Velez, F. Human-in-the-loop interpretability prior. *arXiv preprint arXiv:1805.11571*, 2018.

Letham, B., Rudin, C., McCormick, T. H., Madigan, D., et al. Interpretable classi ers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

Logg, J. M., Minson, J. A., and Moore, D. A. Algorithm appreciation: People prefer algorithmic to human judgment. *Organizational Behavior and Human Decision Processes*, 151:90–103, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Narayanan, M., Chen, E., He, J., Kim, B., Gershman, S., and Doshi-Velez, F. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*, 2018.

Ribeiro, M. T., Singh, S., and Guestrin, C. " why should i trust you?" explaining the predictions of any classi er. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Song, H., Tucker, A. L., Murrell, K. L., and Vinson, D. R. Closing the productivity gap: Improving worker productivity through public relative performance feedback and validation of best practices. *Management Science*, 64(6): 2628–2649, 2017.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pp. 5045–5054. PMLR, 2018.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Yang, H., Rudin, C., and Seltzer, M. Scalable bayesian rule lists. In *International Conference on Machine Learning*, pp. 3921–3930. PMLR, 2017.

A. Details on Tip Inference Algorithm

A.1. MDP Formulation and Search Space of Tips

At a high level, the states encode the progress towards completing all the food orders, the actions encode the available assignments of subtasks to workers, and the rewards encode the number of ticks the user takes to complete all the orders. More specifically, the states encode the following information: (i) in all the orders, which subtasks have been completed so far, and (ii) which subtask has been assigned to each virtual worker (if any), and how many steps remain for the virtual worker to complete that subtask. Next, the actions consist of all possible assignments of available subtasks to available virtual workers. Finally, the reward is -1 at each step, until all orders are completed—thus, the total number of steps taken to complete all orders is the negative reward.

Next, we describe the search space of tips that are considered by our algorithm. Each tip is actually composed of a set of rules inferred by our algorithm. Recall that our algorithm considers tips in the form of an if-then-else statement that says to take a certain action in a certain state. One challenge is the combinatorial nature of our action space—there can be as many as $\frac{k!}{(k-m)!}$ actions, where m is the number of workers and $k = \sum_{j=1}^{n} k_j$ is the total number of subtasks. The large number of actions can make the tips very specific—e.g., simultaneously assigning three distinct subtasks to three of the virtual workers. Instead, we decompose the action space and consider assigning a single subtask to a single virtual worker. To be precise, we include three features in the predicate ϕ: (i) the subtask being considered, (ii) the order to which the subtask belongs, and (iii) the virtual worker in consideration. Then, our algorithm considers tips of the form

> if (order $= o$ $\wedge$ subtask $= s$ $\wedge$ virtual worker $= w$)
> then (assign (o, s) to w);

where o is an order, s is a subtask, and w is a virtual worker.

Even with this action decomposition, we found that these tips are still too challenging for human users to internalize since the tips are very specific. Instead, we post-process the tips inferred by our algorithm by aggregating over tuples (o, s, w) that have the same s and w—e.g., in Figure 7, instead of considering two separate tips shown at the top, we merge them into the tip shown at the bottom.[2] In other words, a tip is a combination of tips $= (\tau_1, \ldots, \tau_k)$. The score that we assign to such a tip is $J(\tau) = \sum_{i=1}^{k} J(\tau_i)$. Then, we choose the tip that achieves the highest score.

[2] We experimented with *combinations* of tips in exploratory pilots, and found that AMT workers were unable to operationalize and comply with such complex tips even though they might be part of an optimal strategy.

A.2. Algorithm Implementation

In principle, we could use dynamic programming to solve for the optimal value function V^*, and then compute the optimal Q-function based on V^*. However, while our state space is finite, it is still too large for dynamic programming to be tractable. Instead, we use the policy gradient algorithm (which is widely used for model-free reinforcement learning) as a heuristic to learn an expert policy π^* for our MDP (Sutton et al., 2000).

At a high level, the policy gradient algorithm searches over a family of policies π_θ parametrized by $\theta \in \mathbb{R}^d$; typically, π_θ is a deep neural network, and θ is the corresponding vector of neural network parameters. This approach requires featurizing the states in the MDP—i.e., constructing a feature mapping $\phi : S \to \{0,1\}^d$. Then, the neural network policy π_θ takes as input the featurized state $\phi(s)$, and outputs an action $\pi_\theta(\phi(s)) \in A$ to take in state s.[3] Then, the policy gradient algorithm performs stochastic gradient descent on the objective $J(\theta)$, and outputs the best policy $\pi^* = \pi_\theta$. In general, $J(\theta)$ is nonconvex so this algorithm is susceptible to local minima, but it is well known that it performs exceptionally well in practice.

In our implementation, the state features include the availability of each sub-task (for each order), the current status of each worker, and the time index. We take π_θ to be a neural network with 50 hidden units; to optimize $J(\theta)$, we take 10,000 stochastic gradient steps with a learning rate of 0.001. In addition, since our MDP has finite horizon, we use a discount factor of $\gamma = 1$.

Once we have computed π^*, we use supervised learning to learn an estimate $\hat{Q}$ of the optimal policy's Q-function Q^{π^*}; specifically, we choose $\hat{Q}$ to be a random forest (Breiman, 2001). The random forest operates over the same featurized states as the neural network policy—i.e., it has the form $\hat{Q}(\phi(s), a) \approx Q^{\pi^*}(s, a)$.

Finally, we apply our algorithm to inferring tips on state-action pairs collected from observing human users playing our game. Because our goal is to help human users improve, we restrict our data to the bottom 25% of human users in terms of performance. In addition, we apply two additional post-processing steps to the set of candidate tips. First, we eliminate tips that apply in less than 10% of the states occurring in the human trace data—i.e., the predicate $\phi(s) = 1$. This step eliminates high-variance tips that have large benefits, but are useful only a small fraction of the time. Second, we eliminate tips that, when they apply, disagree with the expert policy more than 50% of the time—i.e., for a tip (ϕ, a), $\phi(s) = 1$ and $a \neq \pi^*(s)$. This step eliminates

[3] To be precise, $\pi_*(\phi(s))$ outputs a probability $\pi_*(a \mid \phi(s))$ for each action $a \in A$ of taking a in state s. We take the action a with the highest probability.

if (order = burger_1 ^ subtask = cooking ^ virtual worker = chef) then (assign (burger_1; cooking) to chef)
if (order = burger_2 ^ subtask = cooking ^ virtual worker = chef) then (assign (burger_2; cooking) to chef)
$\overset{\text{merge}}{\Longrightarrow}$ assign cooking to chef 2 times

Figure 7: Merging similar tips to improve interpretability.

tips that have large benefits on average, but sometimes offer incorrect advice that can confuse the human user (or cause them to distrust our tips).

B. Details on the Experiment

B.1. Experimental Design

Recall that our tip inference algorithm requires data on the human performance so it can focus on conveying information not already known by the humans. Thus, our behavioral experiment proceeds in two phases. First, we collect data on the actions taken by humans participants when they are not provided with any tips, and use our algorithm in conjunction with this data to infer tips. Second, we evaluate whether providing these tips can improve the performance of subsequent human participants, compared both to a control group of participants who are not provided with any tips as well as two additional groups of participants who are each provided with one of our two baseline tips.

We evaluate our algorithm based on two different configurations of our game, which are designed to evaluate different conditions under which tips might be useful. First, the *normal configuration* consists of a single scenario, the *fully-staffed scenario*, throughout the study. Thus, our goal is to infer tips that help human participants fine-tune their performance at this scenario. The second, *disrupted configuration* starts with the fully-staffed scenario, but then switches to the *understaffed scenario*. Intuitively, we expect human participants to acclimate to the fully-staffed scenario; thus, they may have difficulty adapting to the understaffed scenario where the high-level strategy is very different. Thus, our goal is to infer tips that convey shifts in strategy that are needed to perform well in the new scenario. For a fixed configuration, we additionally vary the tip that is shown to each participant, including no tip (i.e., the control group), our tip, a baseline tip from the naïve algorithm, and a human suggested tip. Our goal is to understand how the choice of tip affects the performance of the human users in the context of that configuration.

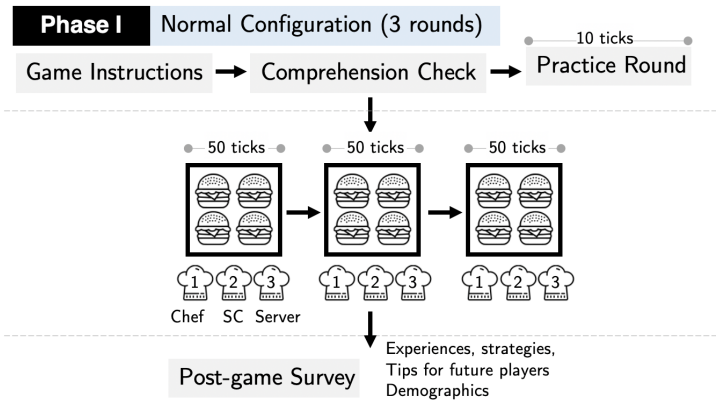
B.2. Experimental Procedure

We perform separate experiments for each configuration of our game. The high-level structure of our experimental design for each configuration is the same; they differ in terms of when we show tips to the participant and which tips we show. For each configuration, our experiment proceeds in two phases. Before starting our game, each participant is shown a set of game instructions and comprehension checks; then, they play a practice scenario twice (with an option to skip the second one).[4] Each participant receives a participation fee of $0.10 for each round they completed. We also provide a bonus based on their performance, measured by the number of steps taken to complete each round. The bonus ranges from $0.15 to $0.75 per round. Participants provided informed consent, and the Institutional Review Board of the University of Pennsylvania approved all materials and procedures in our studies.

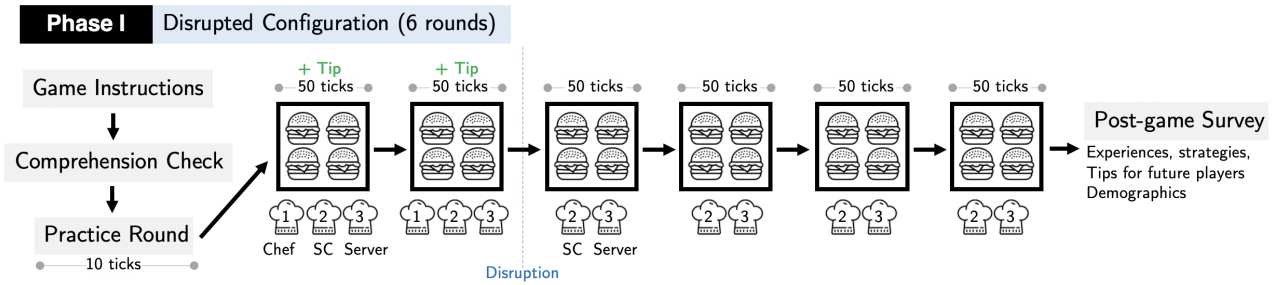
Phase I. For each configuration, we recruit 200 participants via Amazon Mechanical Turk to play the game. The study flow is illustrated in Figure 8. At the end of all rounds of play, we give users a post-game survey where we ask several questions regarding their experience with the game. Additionally, we ask the participants to suggest a tip for future players. In particular, we show each participant a comprehensive list of candidate tips and ask them to select the one they believe would most improve the performance of future players. This list of tips is constructed by merging three types of tips: (i) all possible tips of the format described in A.1 (e.g., "Chef shouldn't plate."), (ii) generic tips that arise frequently in our exploratory user studies (e.g., "Keep everyone busy at all time."), (iii) a small number of manually constructed tips obtained by studying the optimal policy (e.g., "Chef should chop as long as there is no cooking task"). Importantly, this list always contains the top tip inferred using our algorithm.

Next, we use the data from the final round played by the participants to infer tips in three ways. First, we train our tip inference algorithm described in A.2 on the experimental data. Second, we implement the naïve algorithm, which directly tries to imitate the optimal policy based on the frequency of state-action pairs observed in the experimental data. Finally, we analyze the post-game survey and rank

[4] The practice scenario is meant to familiarize participants with the game mechanics and the user interface. In this scenario, they manage three identical chefs to make a single, simple food order. This food order is significantly different than the burger order used in the main game.



((a)) Normal configuration



((b)) Disrupted configuration

Figure 8: Study flow for Phase I.

the candidate tips based on the number of votes by the participants.

Inferred tips. For the normal configuration, 183 participants[5] successfully completed the game. The top three tips inferred from each of the sources are reported in Table 1. For the algorithm tip, "Chef should never plate" is selected as it is expected to be the most effective at shortening completion time (2.43 steps). For the baseline tip, our naïve algorithm selects "Chef should chop once" as it is the most frequently observed state-action pair in the data. Finally, for the human tip, "Strategically leave some workers idle" received the most votes among the participants (28.42%). It is worth noting that all of the tips most voted by past players are in line with the optimal strategy. The first tip captures the key strategy that some virtual workers should be left idle rather than assigned to a time-consuming task. However, it is less specific than other tips. The second and third tips reflect the information participants could learn from assigning different tasks to different workers during the game: server spends the most time cooking while chef spends the most time plating.

[5]The average age of the participants is 34.6 years old, 57.38% are female, and 67.73% have at least a two-year degree.

For the disrupted configuration, 172 participants[6] successfully completed the game. Table 2 reports the top three tips inferred from each of the sources. The best algorithm tip is "Server should cook twice" with the expected completion time reduction of 2.32 steps. Interestingly, only the first and third tips are in line with the optimal policy. The second tip is slightly off as in the optimal policy sous-chef and server should each plate twice. For the baseline arm, all three tips are in line with the optimal policy, and the naïve algorithm chooses "Sous-chef should plate twice". Finally, for the human tip, "Server should cook once" or "Sous-chef should cook three times" got the most votes. Unlike in the normal configuration, the top two human tips here are not part of the optimal policy. In the optimal policy, sous-chef and server should each cook twice. The third human tip does align with the optimal policy; however, it is much less specific than the other tips. This highlights the increased difficulty for humans to identify the optimal strategy in the disrupted configuration compared to the normal configuration.

Phase II. We next evaluate the effectiveness of each of the inferred tips. In this phase, participants are randomly assigned to one of four arms, which differ in terms of the

[6]They are 36.4 years old on average, 61.63% are female, and 77.91% received at least a two-year degree.

Table 1: Top three tips inferred from different sources for the normal configuration.

Normal	**Tip #1**	Tip #2	Tip #3
Algorithm	**Chef should never plate**	Server plates three times	Server should skip chopping once
Baseline	**Chef should chop once**	Server should plate three times	Sous-chef should plate twice
Human (% voted)	**Strategically leave some workers idle** (28.42%)	Server should never cook (21.31%)	Chef should never plate (13.11%)

Table 2: Top three tips inferred from different sources for the disrupted configuration.

Disrupted	**Tip #1**	Tip #2	Tip #3
Algorithm	**Server should cook twice**	Sous-chef should plate once	Server should chop once
Baseline	**Sous-chef should plate twice**	Sous-chef should chop three times	Server should cook twice
Human (% voted)	**Server should cook once** (28.48%)	Server should never cook (23.84%)	Keep everyone busy (16.86%)

tip that is shown to them. These arms include the *control arm* (i.e., no tip), the *algorithm arm* (i.e., the tip inferred by our algorithm), the *baseline arm* (i.e., the tip inferred by the naïve algorithm), and the *human arm* (i.e., the tip with most votes from past players). We recruited 350 AMT users to play each arm in each configuration, totaling to 2,800 users.

The specific tips we show in each round depends not just on the arm, but also varies from round to round depending on the configuration. For the normal configuration, we show the tip for the current arm in all three rounds. However, for the disrupted configuration, the tip for the current arm is specific to the understaffed scenario. Thus, we only show the tip for the current arm in rounds 2-6. In all arms, for rounds 1 and 2, we show the tip inferred using our algorithm for the fully-staffed scenario from the normal configuration. By doing so, we help the human users learn more quickly to play the fully-staffed scenario before switching to the understaffed scenario. Figure 9 illustrates the study flow for the second phase.

B.3. Additional Experimental Results

Compliance. The effectiveness of a tip critically depends on whether the participant follows it; to better understand this relationship, we study how well participants complied with tips across arms. Importantly, participants were not informed of the source of the tips, so variation in compliance is entirely due to the contents of the tips. In the normal configuration, we find that participants increasingly comply with tips across rounds in all arms, as can be seen in Figure 10(a). However, in the final round, a significantly higher fraction participants complied with our tips and the human suggested tips compared to the naïve algorithm tips, suggesting that participants determined that the naïve algorithm did not suggest a useful tip. In the disrupted configuration, the compliance of the human suggested tip ("Server cooks once") is significantly higher than the others, likely because it is the most intuitive. In contrast, our tip ("Server cooks twice") is counter-intuitive since the server is slow at cooking; nevertheless, it is more effective at improving performance when followed. These results suggest that participants are not blindly following tips; instead, they only follow the tip if it suggests a strategy that makes sense to them. Furthermore, they show that our strategy is effective even though our tips are inferred under the assumption that the participant exactly follows the tip. Intuitively, we believe our approach remains effective since our objective of identifying a tip that maximizes long-term payoff is consistent with the idea that participants only follow the tip if it encodes an effective strategy; that is, they follow the strategy as long as they can understand it and it is effective.