Adversarial vulnerabilities of human decision-making

Amir Dezfouli[a,1], Richard Nock[a,b], and Peter Dayan[c,d]

[a]Data61, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Eveleigh, NSW 2015, Australia; [b]Australian National University, Canberra, ACT 0200, Australia; [c]Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany; and [d]University of Tübingen, 72074 Tübingen, Germany

Edited by James L. McClelland, Stanford University, Stanford, CA, and approved October 3, 2020 (received for review August 10, 2020)

Adversarial examples are carefully crafted input patterns that are surprisingly poorly classified by artificial and/or natural neural networks. Here we examine adversarial vulnerabilities in the processes responsible for learning and choice in humans. Building upon recent recurrent neural network models of choice processes, we propose a general framework for generating adversarial opponents that can shape the choices of individuals in particular decision-making tasks toward the behavioral patterns desired by the adversary. We show the efficacy of the framework through three experiments involving action selection, response inhibition, and social decision-making. We further investigate the strategy used by the adversary in order to gain insights into the vulnerabilities of human choice. The framework may find applications across behavioral sciences in helping detect and avoid flawed choice.

decision-making | recurrent neural networks | reinforcement learning

Advertisers, confidence tricksters, politicians, and rogues of all varieties have long sought to manipulate our decision-making in their favor, against our own best interests. Doing this efficiently requires a characterization of the processes of human choice that makes good predictions across a wide range of potentially unusual inputs. It therefore constitutes an excellent test of our models of choice (1). We have recently shown that recurrent neural network (RNN) models provide accurate, flexible, and informative treatments of human decision-making (2–4). However, how well these RNNs can interpolate and extrapolate outside the range of conventional inputs, and then the insights they can offer into human choice frailty, are unclear. To examine these issues, we require a systematic way of 1) finding the vulnerabilities in models of choice and 2) proving that, and how, these vulnerabilities are also exhibited by humans. Here, we provide a general framework for doing this (Fig. 1) based on RNNs fitted to behavioral data.

One line of systematic study of vulnerabilities started in image classification, with seminal early observations from Szegedy et al. (5) that deep artificial neural networks are brittle to adversarial change in inputs that would otherwise be imperceptible to the human eye. This computer vision weakness of the machine has been an angle of attack to design adversaries for reinforcement-learning (RL) agents (6), followed by general formal insights on adversarial reinforcement learning on the more classical bandit settings (7). To analyze human choice frailty, our framework involves two steps, the key one also involving a machine-vs.-machine adversarial step in which a (deep) RL agent is trained to be an adversary to an RNN; this latter model is trained in a previous step to emulate human decisions following refs. 2–4. This provides the general blueprint to tackling the first question. To show the promise of this framework to surface such vulnerabilities in human subjects, we applied it to three decision-making tasks involving choice engineering (1), response inhibition (8), and a social exchange game (9) and tested the resulting adversaries on volunteers to assess the biases. We show that in all of the tasks the framework was able automatically to specify adversarial inputs which were effective in steering choice processes to favor particular target actions or goals. We further use simulations to illustrate and interpret the strategies used by the adversaries. Note that although we refer throughout to "adversaries," exactly the same framework can be used for cooperative ends, or to increase social welfare. Indeed, to show this, we create an adversary in the social exchange game whose intent is to enforce a fair outcome, rather than an unfair one.

The Adversarial Framework

The interaction of the subjects with the task is shown in Fig. 1*A*. On each trial t, the subject (n) receives what we call the learner reward (r^n_{t-1}), which is delivered on the basis of their previous action (a^n_{t-1}) and is provided with the current observation (o^n_t; e.g., cues on a computer screen). They then take the next action, a^n_t. The process then repeats with the subjects receiving the learner reward of the action chosen (r^n_t) and the next observation (i.e., o^n_{t+1}).

In the nonadversarial case, learner rewards and observations are typically generated by a particular, predefined Markov or partially observed Markov decision process. For instance, actions might lead to learner rewards with certain fixed or roving probabilities. By contrast, in our case, an adversarial model operates behind the scenes, deciding the learner reward for each action and the next observation that will be shown to the subject. To avoid triviality, adversaries work within budget constraints, which might limit or equalize per action the total number of learner rewards that the adversary could deliver. Within such constraints, the adversary faces a sequential decision-making problem to pick the learner rewards and observations that will shape subjects' behaviors over the long run according to the target pattern or goal, expressed either in terms of actions (e.g., getting them to prefer a particular action) or goals (e.g., maximizing the adversary's return in a game-theoretic context).

Significance

"What I cannot efficiently break, I cannot understand." Understanding the vulnerabilities of human choice processes allows us to detect and potentially avoid adversarial attacks. We develop a general framework for creating adversaries for human decision-making. The framework is based on recent developments in deep reinforcement learning models and recurrent neural networks and can in principle be applied to any decision-making task and adversarial objective. We show the performance of the framework in three tasks involving choice, response inhibition, and social decision-making. In all of the cases the framework was successful in its adversarial attack. Furthermore, we show various ways to interpret the models to provide insights into the exploitability of human choice.

Author contributions: A.D., R.N., and P.D. designed research; A.D. and P.D. performed research; A.D. analyzed data; and A.D., R.N., and P.D. wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission.

[1]To whom correspondence may be addressed. Email: amir.dezfouli@data61.csiro.au.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2016921117/-/DCSupplemental.

First published November 4, 2020.

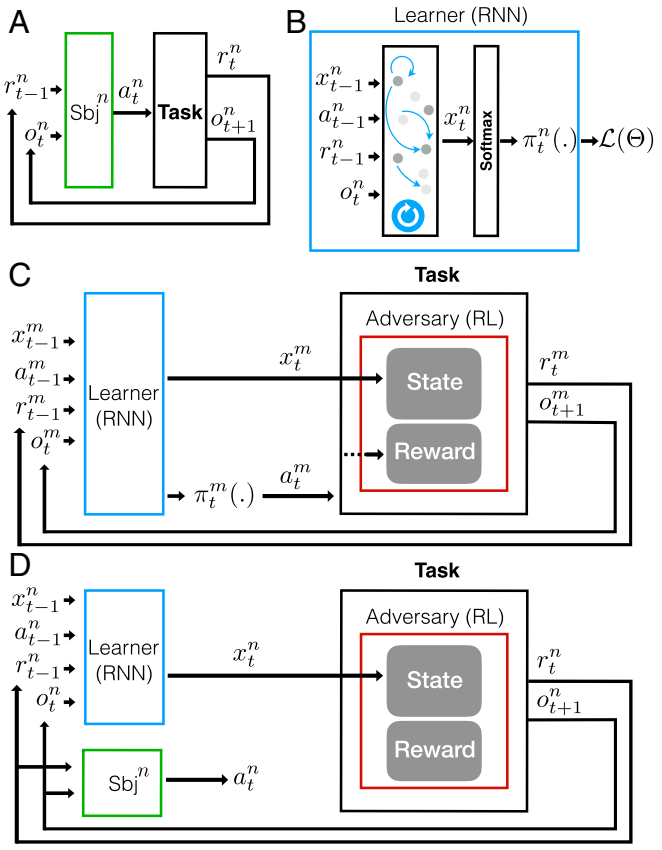

Fig. 1. The framework. (*A*) The interaction of the subjects with the task. At each trial *t*, subject *n* receives the learner reward (r^n_{t-1}) for its previous action and a new observation (o^n_t) from the environment and takes action a^n_t. The environment then sends a new learner reward and observation back to the subject. This cycle continues until the end of the task. (*B*) The behavior of the subjects is modeled by an RNN (parameters Θ). The inputs to the RNN are the previous action (a^n_{t-1}), learner reward (r^n_{t-1}), and the current observations from the task (o^n_t) along with the previous internal state of the RNN (x^n_{t-1}). After receiving the inputs, the RNN updates its internal state, which is then mapped to a softmax layer to predict the next action $\pi^n_t(.)$. These predictions are then compared with the actual actions taken by the subjects to build loss function $\mathcal{L}(\Theta)$, which is used for training the model. The trained model is called the learner model. (*C*) The adversary is an RL agent which is trained to control the learner model. We consider closed- and open-loop adversaries. For the former, the internal state of the learner model (i.e., x^m_t for simulation *m*), which summarizes its learning history, is used as the state of the adversary. The adversary then determines the learner reward and the next observation to be delivered to the learner model. The learner model subsequently takes its next actions and this cycle continues. The adversarial reward (Reward), which is used to train the adversary, depends on the alignment between the action taken by the learner model (a^m_t) and the adversary's objectives. By contrast, an open-loop adversary does not have access to the internal state of the learner model and generates rewards and observations without reference to them (not shown in the figure), only to the adversarial rewards, which do depend on them. In the open-loop case, the adversarial choices specify rewards and observations for all possible actions of the learner model (or the subject). Thus, action a^m_t is *not* passed to the adversary, but is passed to the task to calculate the reward that should be delivered to the learner based on the reward that the adversary has assigned to each action. (*D*) Using the trained adversary and the learner model for generating (here, closed-loop) adversarial interactions with humans. Human subject *n* receives the learner reward for their actions and the observations from the trained adversary. They subsequently take an action (a^n_t) which is received by the learner model to update its internal state x^n_t. This state is then sent to the adversary to determine the learner reward for the action and the next observation. This cycle continues until the end of the task.

We modeled the adversary using an RL agent. On each trial, the adversary receives the learning history of the subject as input and produces as output the learner reward and the next observation to be delivered to the subject. In RL terms, the learning history of the subjects constitutes the state of the adversary, and its output are the adversarial choices*, which determine the learner reward and the observation inputs for the subjects. The immediate reward provided to the adversary to criticize its adversarial choices (we call this the adversarial reward) reflects whether its output made the subjects meet the target behavior or goals. Within this structure, the adversary is trained to earn the maximum-sum adversarial reward over the whole task, corresponding to producing outputs which most effectively push the subjects toward target actions or goals.

In principle, the adversary could be trained through direct interactions with humans. In practice, however, this approach is unfeasible given the delays involved with interacting with humans and the large number of training samples required. Instead, we use an alternative, model-based approach based on recent RNN models of human choice processes (2–4). Here, an RNN, called the learner model is trained from data collected from humans playing the task nonadversarially and is then used to predict human behavior under the different inputs that the adversary might try. RNNs are suitable because they provide a flexible differentiable family of models which are able to capture human choice processes in detail. Our approach has two additional advantages over training the adversary against humans. First, it can be carried out in settings where substantial training data (nonadversarial) already exist; second, our approach comes at the reduced human cost of just modeling the human behavior for the nonadversarial task. Provided the effect of covariate shift is not too severe, this can then be used as input for diverse adversarial training scenarios.

Learner Model. In detail, the learner model (Fig. 1*B*; determined by parameters Θ) comprises an RNN and a softmax layer which maps the RNN internal state to a probability of selecting each action (4). On trial t for subject n, the RNN layer has an internal state denoted by vector x^n_{t-1}, which reflects the RNN's inputs on trials $1 \ldots t-2$. This state is recurrently updated based on the previous action (a^n_{t-1}) and learner reward (r^n_{t-1}) and the current observations (o^n_t). The output of the RNN layer x^n_t is passed to a softmax layer to predict the next action (i.e., $\pi_t(\cdot)$). The prediction is compared with the subject's actual action a^n_t, resulting in loss $\mathcal{L}(\Theta)$ that is used for training the network. See *Materials and Methods* for more details.

Training the Adversary. The adversary is modeled as an agent whose adversarial choices are learner rewards and observations on each trial, emitted with the goal of optimizing target behaviors subject to constraints. Since the adversarial choice on one trial can affect all of the subsequent actions of the learner model (hopefully characterizing similar dependence in the subjects), the adversary faces a sequential decision-making problem, which we address in an RL framework (Fig. 1*C*).

The future actions of the learner model after trial t depend on its prior history only through its internal state x^m_t (for simulated learner m). Therefore, the RL adversary uses x^m_t as the state of the environment to decide the learner reward r^m_t and next observation o^m_{t+1} that it provides to the learner model. The process repeats with the new state of the learner model (x^m_{t+1}) being passed to the adversary. Within this structure, the policy of the adversary in choosing learner rewards and observations is trained to yield maximum cumulative adversarial reward subject

*For clarity, we refer throughout to decisions in the task as actions and the output of the adversary as adversarial choices.

to constraints. We used the advantage actor–critic method (A2C) (10) and deep Q-learning (DQN) (11) for training the adversary. Note that the learner model is not trained in this stage and its weights are frozen. See *Materials and Methods* for more details on training and the constraints.

Along with this closed-loop adversary, which is guided by the past actions of the subject (via the internal state of the learner model), we also consider an open-loop adversary. This chooses learner rewards and observations without receiving the subject's actions or the internal state of the learner model. Its policy is trained directly just using samples generated from the learner model.

Using the Adversary. Fig. 1*D* depicts how the trained closed-loop adversary and the learner model are used in an experiment involving human subjects. The learner model does not choose actions but receives the actions made by subject n as input and tracks their learning history using x_t^n. In turn, on trial t, x_t^n is received by the adversary to determine the learner reward r_t^n and the next observation o_{t+1}^n which the subject will use to choose their next action a_{t+1}^n. The same input, along with the actual action and learner reward, is delivered to the learner model. This cycle continues until the end of the task.

Results

Bandit Task. This experiment is based on the adversarial bandit task introduced in ref. 1. On each trial, subjects make choices between two squares, one on the left of the screen and the other on the right. After each choice, the subjects receive feedback about whether (smiley face) or not (sad face) their action earned a learner reward. A priori, before each choice, the adversary assigns a potential learner reward to both potential actions (e.g., that the left action will be rewarded and the right action will not get rewarded), and this is faithfully delivered ex post based on the subject's choice. The goal of the adversary is to assign learner rewards to the actions in a way that makes subjects prefer one of the actions (called the "target" action) over the other one (the "nontarget" action). The target action is predetermined (e.g., before the experiment starts the left action is set as the target action). The adversary is required to achieve this goal under a constraint: It must assign exactly 25 a priori learner rewards to each action, that is, it cannot simply always assign learner rewards to the target action and no learner reward to the nontarget action.

Q-Learning Model. We first evaluated the framework in the synthetic setting of Q-learning. We generated data from a Q-learning algorithm (1,000 learners) with the same parameters used in ref. 1 and used these data to train the learner model. Then we used RL to train the adversary to exploit the learner model. The adversary received an adversarial reward every time the learner model chose the target action. The constraint was enforced at the task level, that is, after the adversary has allocated 25 learner rewards to an action, no more learner rewards will be allocated to that action. Conversely, if the adversary has only assigned $25-k$ learner rewards to an action by trial $T-k$ (for $k>0$, where T is the maximum trial number), that action will be assigned a learner reward on the remaining k trials.

The trained adversary was evaluated against both the learner model and the Q-learning model. The main dependent variable is the "bias," which is the percentage of trials on which the target action was chosen (Fig. 2*A*). As the figure shows, the adversary was able to guide the choices toward the target action. The average bias in playing the Q-learning model was 73.4% (ADV vs. QL column). This is similar to the results obtained in ref. 1, but here the results are obtained without knowing that the underlying algorithm is Q-learning. The average bias when the adversary was simulated against the learner model is 73.8% (ADV vs. LRN column), which is comparable with the results against the actual Q-learner.

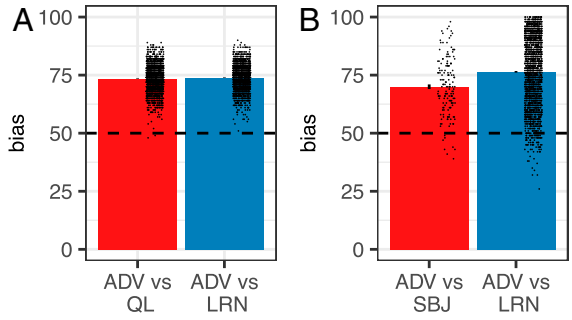

Fig. 2. The performance of the adversary (ADV) in the bandit experiment against various opponents. "Bias" is the percentage of target actions selected by the opponent in response to learner rewards assigned by the adversary; error bars represent 1 SEM. Each black dot (jittered for display) represents one simulation. (*A*) ADV against an actual *Q*-learner (QL, red) or the learner model trained on *Q*-learners (LRN, blue). (*B*) ADV against human subjects (SBJ, red) or the learner model trained on other human subjects from ref. 1 (LRN, blue). Horizontal dashed lines represent equal selection of actions.

Next, we sought to uncover the strategy used by the adversary. Two 100-trial simulations are shown in Fig. 3*A* (ADV vs. Q-learning). The blue and red circles indicate that the learner model selected the target and nontarget action respectively. The vertical blue and red lines indicate that a learner reward was assigned to the target and nontarget actions respectively. No line indicates that the learner reward was not assigned to the corresponding action. The green shaded area shows the probability the learner model awards to the target action. The general tactic used by the adversary is to assign a few learner rewards to the target action in the first half of the task; these few learner rewards, however, were sufficient to keep the probability of choosing the target action around 70 to 80% (shown by the green area), which is because the adversary never delivered nontarget learner rewards in this period. Toward the end of the task, the adversary "burns" the nontarget learner rewards whenever the probability of the target action is above chance; at the same time the density of target learner rewards is increased to cancel the effect of nontarget learner rewards. The combination of these strategies made the learner model choose the target action around 73% of the trials.

Human Subjects. We next applied the framework to develop an adversary for human choices in the task. The learner model was trained using the (nonadversarial) data published in ref. 1 for $N=484$ subjects.

The trained adversary and the learner model were used to collect data from humans using Amazon Mechanical Turk ($N=157$). The results are shown in Fig. 2*B*. The bar "ADV vs SBJ" shows the bias of subjects when playing against the adversary. The average bias was 70%, which is significantly higher than the equal selection of actions that might naively be implied by the equal numbers of learner rewards available for each choice (50% bias baseline; Wilcoxon signed-rank test, $P<0.001$). This shows that the adversary was successful in leading subjects to choose the target action. The Fig. 2*B* bar "ADV vs LRN" shows the bias when the adversary is playing against the learner model (simulated). The average bias is 76.4%, which is better than when the adversary is pitted against human subjects. One potential difference is the subject populations used to train vs. test the adversary.

To investigate the adversary's strategy, we again simulated it, but now against the human-trained learner model (Fig. 3*B*). The adversary appears to seek to prevent subjects from experiencing the learner rewards assigned to the nontarget action, but to

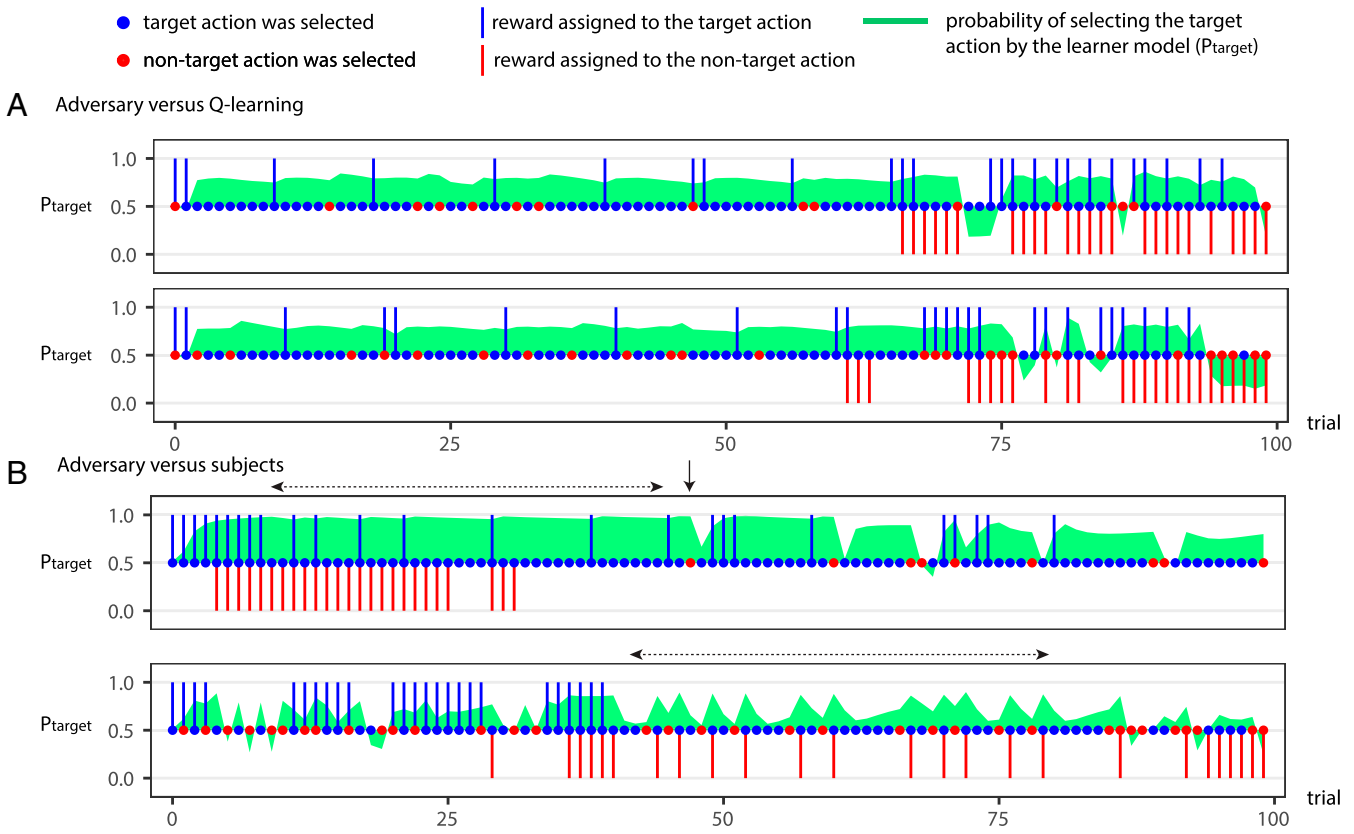

Fig. 3. Simulations of the adversarial interactions over 100 trials in the bandit task. Blue and red circles indicate the selection of the target and nontarget actions, respectively. Vertical blue and red bars indicate the prospective assignment of learner rewards to target and nontarget actions, respectively. The green area shows the probability the learner model accords to the target action. (*A*) Two simulations of the adversary against a *Q*-learning model. (*B*) Two simulations of the adversary against the learner model trained using human behavior. Solid and dashed arrows refer to the trials mentioned in the main text.

make them see the learner rewards assigned to the target action, and therefore to select it. To achieve this, learner rewards are assigned to the target actions when this action is likely to get selected in the next trial, but for the nontarget action, learner rewards are assigned when this action is unlikely to get selected.

Some of the tactics the adversary employs are evident in these simulations. In in Fig. 3*B*, *Top* the adversary starts by continuously assigning learner rewards to the target action. Once the subject was set on choosing the target action, learner rewards are assigned to the nontarget action to "burn" them without subjects noticing. A second tactic is using partial reinforcement after the initial serial learner reward delivery on the target action (shown by the dashed horizontal line above the panel). This saves target learner rewards for later while not materially affecting choice probabilities. A third tactic is applied when the learner model takes a nontarget action, as shown by the vertical arrow on the panel; here, the adversary briefly increases target learner reward density to bring the subject back to choosing the target action.

The second simulation (Fig. 3*B*, *Bottom*) shows a more complex strategy which allows the adversary to burn the nontarget rewards "discreetly" for a learner model that has a tendency to alternate. In the period indicated by the horizontal dashed arrow, when the learner model tries the nontarget action (the red circle) without getting reward, on the next trial it tends to take the target action (blue circle). This pattern of behavior is detected by the adversary and exploited by assigning a learner reward to each nontarget action after each selection of the nontarget action. This efficiently hides nontarget learner rewards from the subjects. Altogether, such tactics substantially bias the subjects toward the target action.

Across the two experiments, it is evident that the strategy used against humans is quite different from the strategy used against Q-learning. Indeed, if we apply the adversary adapted to the humans to a Q-learning learner, the average bias is 55.2 and if we apply the adversary developed for Q-learning to humans (on a learner model trained using human data) the average bias is 58.1 (*SI Appendix*, Fig. S1). These differ markedly from the biases reported when the adversaries play against their corresponding learner models.

Go/No-Go Task. Our second experiment involved a go/no-go task that was implemented in ref. 8. On each of 350 trials, subjects see either a go stimulus (e.g., an orange circle) or a no-go stimulus (e.g., a blue triangle). Go stimuli are common (90% of trials) and subjects are required to press the space bar in response to them. No-go stimuli are rare and require subjects to withhold responding. In the nonadversarial case, no-go stimuli are uniformly distributed across trials. By contrast, the adversary rearranges the no-go stimuli to encourage the largest number of mistakes (i.e., pressing space bar to no-go stimuli, or withholding responding to go stimuli), without changing the total number of each stimulus type. Similar to the previous experiment, the constraint (exactly 90% of trials should be no-go) is enforced by fiat at the task level.

There are two differences between this experiment and the bandit experiments. First, the adversary determines observations (go vs. no-go) rather than learner rewards. Second, we considered an open-loop adversary—that is, one that did not receive the state information x_t^n on trial t and indeed knew nothing about how a particular subject responded.

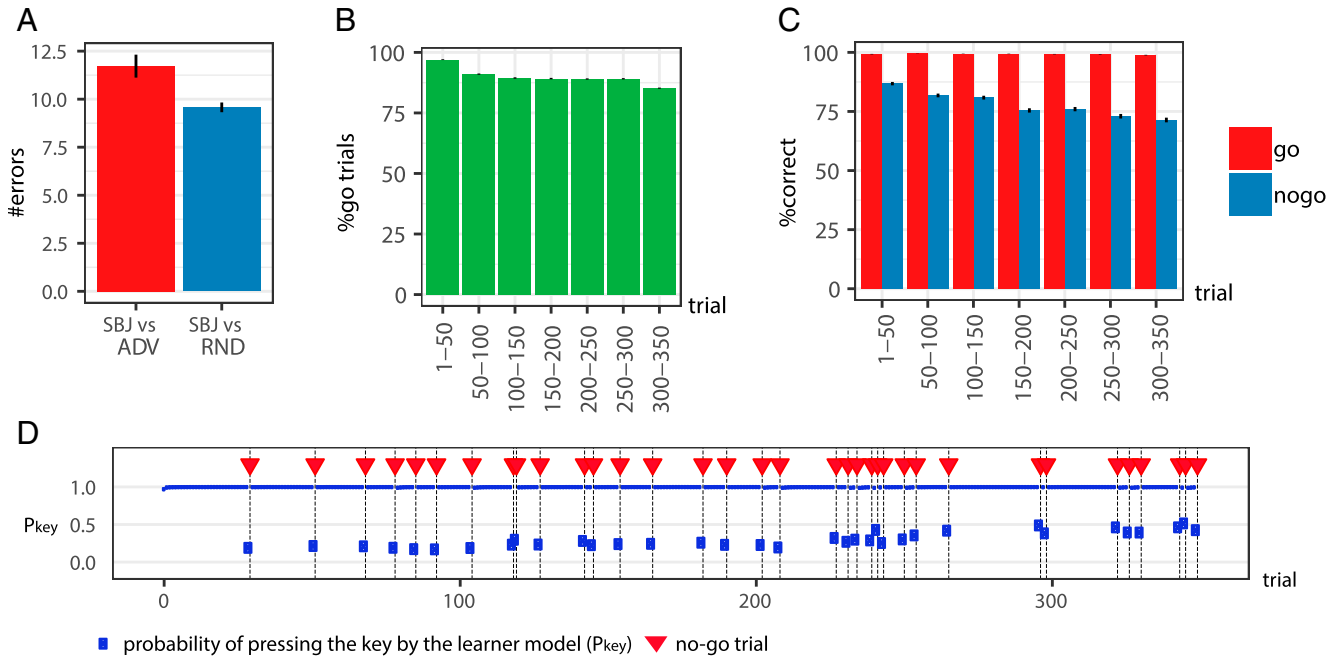

Fig. 4. Adversarial strategy in the go/no-go task. (*A*) The number of errors made by the subjects when no-go trials are delivered at random (SBJ vs ADV) and when no-go trials are delivered by the adversary (SBJ vs ADV). (*B*) The distribution of the go trials over the task period delivered by the adversary. (*C*) Percentage of the correct responses by the subjects in go and no-go trials when the no-go trials are delivered at random. (*D*) A sample simulation. Red triangles indicate no-go trials. The blue shapes indicate the probability of pressing the key assessed by the learner model. Error bars represent 1 SEM.

We started by training the learner model using the data generated in the random case ($N = 770$ subjects collected using Amazon Mechanical Turk were included in the analysis). Next, we trained the adversary using the learner model, but unlike the previous experiment the adversary did not receive the state of the learner model. The trained adversary was then used to collect data from humans using Amazon Mechanical Turk ($N = 139$). The results are shown in Fig. 4*A*. Subjects on average made 11.7 errors when playing against the adversary and 9.5 errors when no-go trials are distributed randomly (Wilcoxon rank-sum test; $P < 0.001$). Therefore, the adversary was successful in finding a state distribution which induces extra errors. The number of excess errors may seem modest, but the task is extremely austere, and so any significant change is an achievement.

To elucidate the adversary's strategy, Fig. 4*B* shows the ratio of go trials allocated by the adversary across the task. The adversary allocates more no-go trials toward the end of the task. This may be since subjects (in the training data in which the no-go trials were randomly distributed) were more likely to make errors in the no-go condition later in the task (Fig. 4*C*). However, the adversary faces a challenging problem, since assigning all of the no-go condition in a short period at the end of the task will likely have the opposite effect, since the subjects will have to stay alert only in a short period, and therefore a trade-off between these factors is required.

Fig. 4*D* shows an example simulation. Red triangles indicate no-go trials and blue symbols indicate the probability of pressing the key (space bar) according to the learner model. Consistent with the data, the probability of making an error in the no-go trials increases over trials; thus, the adversary spread the no-go trials across the task with a bias toward the end of the task to induce more errors.

Multiround Trust Task. In the third experiment we evaluated our framework on the multiround trust task (MRTT). MRTT is a social exchange task for two players, whose roles are called "investor" and "trustee" (9, 12). The task involves 10 sequential rounds. In each round the investor receives an initial endowment of 20 monetary units. The investor can share a portion (or all) of this endowment. The shared amount is tripled by the experimenter and sent to the trustee. Then, the trustee can send back any portion of this amount to the investor, hereafter called repayments. The total amount earned by each player in the task is the sum of what they earned in each round.

In our framework humans play the role of the investor and the adversary plays the role of the trustee. The actions of the adversary correspond to the proportions that the investor sends back to the trustee (discretized to five actions corresponding to 0, 25, 50, 75, and 100% repayments). The goal of the adversary is to make repayment choices that persuade the investor to make payments that meet adversarial objectives. We trained two adversaries based on two different objectives: 1) a MAX objective, in which the aim of the adversary is to gain the most over the 10 rounds, and 2) a FAIR objective, in which the aim of the adversary is to balance the total earnings of the trustee and investor over the whole task. Comparing the two objectives shows the extent to which the adversary can adapt to the problem is faces; the FAIR objective allows us to evaluate the framework on motivating humans to make cooperative rather than competitive adversarial choices.

Note that the pattern of play can be substantially influenced by the number of remaining rounds (for instance, a MAX-trained trustee has no incentive to repay anything on the last round; the prospect of this noncooperation can make investors cautious) (13). This induces dependencies across rounds, making the task a sequential decision-making problem. This is a significant difference from the bandit setting of the first experiment.

We first collected data on a random investor (RND condition, that is, the investor selects action uniformly at random) using Amazon Mechanical Turk ($n = 232$). Fig. 5*A* shows the effect of the repayment amount on the investment by the subjects in the next round (marginalizing over the round number). Subjects generally invested more if a higher portion had been repaid to them in the previous trial. This implies that in the case

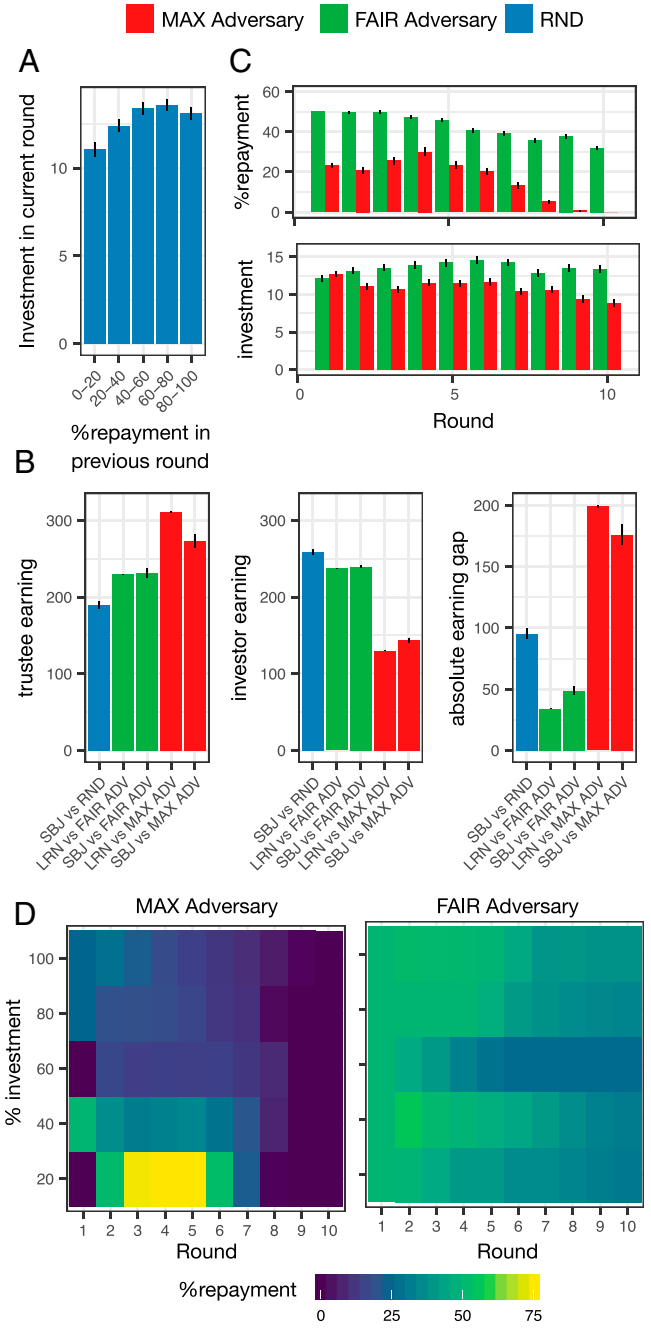

Fig. 5. MRTT. (*A*) The investment in a round as a function of the proportion of the investment repaid by the trustee in the previous round, for the data collected in the random condition. (*B*) The total amount earned by the trustee (adversary) and investor (subjects) and the absolute gap between them in different conditions. "LRN" refers to the learner model. "SBJ" refers to subjects. "RND" refers to the random investor. (*C*) The percentages of investment and repayment in each round for MAX and FAIR adversaries. (*D*) Percentage of repayment by the adversary by the adversary in each round as a function of the percentage of investment.

of MAX objective the adversary should aim to increase future investments by making high repayments and building trust, but it also needs to share as little as possible to profit from the investments it receives. This is a nontrivial decision-making problem for the adversary.

We used the data collected in the random condition to train the learner model and used this learner model to train two adversaries based on the above objectives. Each adversary was tested on the data collected using Amazon Mechanical Turk ($n = 155$ for FAIR adversary and $n = 209$ for MAX adversary). Fig. 5*B* shows the performance of the adversaries tested on subjects and also their performance in simulations using the learner model. As the figure shows, the gains of the MAX adversary are significantly higher than the RND investor and FAIR adversaries (Wilcoxon signed-rank test, $P < 0.001$). On the other hand, the absolute earning gap (i.e., the absolute difference between the earning of the trustee and the investor over the whole task) is lower for the FAIR adversary than for both RND and MAX adversaries (Wilcoxon signed-rank test, $P < 0.001$). Therefore, the adversaries were successful in guiding subjects' actions toward their objectives. The performance of the adversaries tested on the subjects is slightly worse than the performance on the simulations against the learner model, which might partly be due to the differences in subjects' pools used for training the learner models and testing the adversaries.

Fig. 5*C* shows the percentage of repayments and investments in each round. As expected, the FAIR adversary repays more (Fig. 5*C*, *Top*), which makes the subjects increase their repayments over the rounds (Fig. 5*C*, *Bottom*). The MAX adversary repays less, and, again as expected, repays nothing in the last round.

Fig. 5*D* shows the adversary strategy in more details and as a function of the investments. The MAX adversary starts with high repayments to gain the investors' trust but then sharply decreases the repayments to exploit their cooperativity. This pattern depends on the investment amount: If the investment is very low (20%), in the early trials the adversary tries to persuade the subjects to increase their investment by making large repayments (up to 75%). This strategy is different from the FAIR adversary: In this case if the investment is around 50% (10 units), the adversary returns around 30%, which makes each player earn 20 units. With investments less than 5, no matter what action the adversary takes there will be a gap between the earnings. As such, similar to the MAX case the adversary aims to build trust by making high payments in early rounds but later on the repayments become proportional to the investments. Note that the adversary is not making repayments to balance the gains at each round independently, in which case its strategy should be same in all of the rounds, but it is adjusting repayments to guide the investors' actions to high values so that the gains can be balanced with appropriate repayments.

Discussion

We have provided a general framework for generating adversaries to elicit target behavior in a wide variety of human decision-making processes. In three experiments, we showed that the framework is effective in inducing target behaviors and also interpretable using simulations. We also showed that the framework can be used in settings that the target behavior is nonadversarial, such as inducing fairness in two-player games.

Cognitive Biases. The strategies used by the adversaries are driven by the choice characteristics embedded in the structure and weights of the learner model. Such choice characteristics, when exploited by the adversary, can lead to irrational and suboptimal actions by the learner model. In this respect, they can be seen as distilled, implicit generalizations of the traditional cognitive biases which underlie different sorts of deviations from normative accounts of choice (14). Exploring the relationship between adversarial strategies and traditional cognitive biases is a direction for future research.

Batch RL. An alternative to the framework developed here is using batch RL algorithms, which are able to learn from a precollected dataset without interacting with the environment (15).

There would then be a parallel with the learning to reinforcement learn (16) or RL^2 (17) frameworks. One advantage of the current approach over these methods is that the policy of the adversary can be interpreted with respect to the behavior of the learner model which has been used to train it.

Experiment Design. Although we considered one-step adversarial scenarios, in principle the same framework can be used for multistep experimental design (see also ref. 18). Say, for example, that an experimenter desires the subjects to exhibit a specific pattern of behavior, but the experimental parameters (e.g., probabilities, delays, etc.) that yield the pattern are unknown. Following the framework here, the experimenter can train a learner model and use that learner model to train an open-loop adversary which determines the optimal set of parameters for obtaining the desired behavior. The obtained parameters then can be tested to see whether they make the subjects exhibit the desired behavior; if they did not, the learner model can be retrained using the new dataset and this process can be iterated until the desired behavior is obtained. We conjecture that the procedure will often converge.

One reason that the subjects might not exhibit the desired behavior in early iterations of this procedure is that the method depends on the generalization of the predictions of the learner model from nonadversarial regimes to the adversarial regimes. The violation of this assumption implies that the adversary pushes the learner model in the parts of the state-space which have not been visited in the nonadversarial training and therefore the approximations of the human behavior in those regions will be poor. This covariate shift phenomenon is known as "extrapolation error" in the batch reinforcement learning literature (19) and several solutions have been suggested (see also ref. 20 for batch supervised learning), which can be applied to the current framework. Here, we used RNNs with a relatively small number of cells to avoid this issue, but the extension of the framework to address extrapolation error can be an interesting future step.

Materials and Methods

Training the Learner Model. The learner model was trained using the objective function $\mathcal{L}(\Theta) = -\sum_{n=1\ldots N}\sum_{t=1\ldots T^n}\log\pi_t^n(a_t^n)$, in which Θ refers to the free parameters of the model, N is the total number of subjects, and T^n is the number of trials completed by subject n. Note that $\pi_t(a_t)$ depends on the history of inputs, which are omitted for simplicity. Optimization was based on the Adam optimizer (21). The learner model was implemented using Tensorflow (22) and the gradients were calculated using automatic differentiation. The RNN in the learner model was based on gated recurrent unit architecture (23). The optimal number of training iterations (early stopping) and the number of cells for each experiment were determined using 10-fold cross-validation. The optimal number of iterations and cells were then used to train a learner model using the whole dataset for each experiment. The number of cells considered was 3, 5, 8, and 10 cells, as in previous work (4). The optimal number of cells and training iterations for each experiment is presented in *SI Appendix*, Table S1. Note in the case of MRTT experiment we discretized the investments to five actions corresponding to ranges 0 . . . 4, 5 . . . 8, 9 . . . 12, 13 . . . 16, and 17 . . . 20.

Training the Adversary. The DQN algorithm was used for training the adversary in the bandit experiments (11). The inputs to the closed-loop adversary were the internal state of the learner model (x_t), its policy vector (probability of taking each action by the learner model), the trial number, and the number of rewards so far assigned to each action. Note that the internal state of the learner model embeds other elements such as policy vector, but we also fed these elements explicitly to the adversary to speed up training. The output of the adversary was the value of each of the four actions corresponding to the combination of assigning reward/no-reward to each of the choices available to the learner model.

The adversary neural network had three fully connected layers with 128, 128, and 4 units, with ReLU, ReLU, and linear activation functions. Replay buffer sizes of 200,000 and 400,000 were considered. The $\epsilon-$greedy method was used for exploration with $\epsilon \in \{0.01, 0.1, 0.2\}$. Learning rates $\{10^{-3}, 10^{-4}, 10^{-5}\}$ were considered for training the adversary using the Adam optimizer. These performance of these 18 combinations was evaluated after $\{1, 2, 3, 4, 5, 6, 7, 8, 9\} \times 10^5$ training iterations. For the performance evaluation, the adversary was simulated against the learner model 2,000 times and the average bias was calculated. The adversary with the highest average bias was used. For the humans/Q-learning experiments, the highest bias was achieved with buffer size 400,000/400,000, $\epsilon = 0.1/0.01$, and learning rate $0.0001/0.001$

For the go/no-go experiment, since the task was longer (350 trials) we used a policy-gradient method and advantage A2C algorithm (10) for training the open loop adversary. We also used an additional entropy term to encourage sufficient exploration (24). The input to the adversary was the current trial number and the total number of go/no-go states assigned. The output of the adversary was the policy (i.e., the probability that the next trial is go or no-go). Note that the policy is stochastic. The network did not receive the learner model's internal state as input since we sought an open-loop adversary. Both the value and policy networks for the adversary had three layers, with 256, 256, and 1 unit(s) in the value layer and 256, 256, and 2 units in the policy layer. The activation functions were ReLU, ReLU, and linear, respectively, in each layer. We considered two values for the weight of entropy $\{0.01, 0.5\}$ and the adversary with 0.01 entropy weight achieved a higher performance against the learner model (in terms of the average number of errors made by the learner model in 1,500 simulation). The adversaries were implemented in Tensorflow and trained using the Adam optimization method (21).

For the MRTT experiment, we used DQN as for the bandit experiment. The inputs to the adversary were the internal state of the learner model (x_t), its policy vector (probability of taking each action by the learner model), the action taken by the investor (learner model), and the trial (round) number. The output of the network was the action values of each of the five actions corresponding to different proportions of repayments. In the case of the MAX adversary, the reward delivered to the adversary in each round was the amount earned ($3 \times$ investment $-$ repayment). In the case of FAIR adversary, the reward in each round was zero except for the last round in which the reward was the negative absolute difference between the gains of trustee and investor over the whole task.

The adversaries were neural networks with three fully connected layers with 128, 128, and 4 units, with ReLU, ReLU, and linear activation functions. Replay buffer sizes of 200,000 and 400,000 were considered. The $\epsilon-$greedy method was used for exploration with $\epsilon \in \{0.01, 0.1, 0.2\}$. Learning rates $\{10^{-3}, 10^{-4}, 10^{-5}\}$ were considered for training the adversaries using the Adam optimizer. The performance of the 18 combinations was evaluated after $\{2, 5, 10\} \times 10^5$ training iterations. For the performance evaluation, the adversaries were simulated against the learner model for 15,000 times and the average objectives were calculated.

Data Collection. The study was approved by the Commonwealth Scientific and Industrial Research Organisation (CSIRO) ethics committee (Ethics Clearance 102/19). Subjects agreed to an online consent form prior to each task for their participation in the study. The data were collected using Amazon Mechanical Turk. In all experiments, the participants received \$0.4. In the bandit task, they also could earn \$0.01 for each smiley face. In MRTT subjects received \$0.01 for each monetary unit earned in the task.

For the bandit and MRTT tasks, during the data collection the adversaries ran on a back-end Python server communicating with the task running in the web browser by receiving the action information from the task and sending the assigned rewards back to the test for the next trial. For the go/no-go experiment, since the delay between the trials was short it was not feasible to run the adversary on the back-end due to the communication lag. Instead, the adversary was exported to Javascript and ran in the web browser using Tensorflow.js framework.

For the case of the go/no-go task, we selected subjects with performance in the 75th percentile (which corresponded to less than 32 errors) and used their data for training the learner model. We then collected the data in the adversarial conditions and used the same threshold (fewer than 32 errors) to select the subjects who were included in the analysis. In the bandit and MRTT tasks, all of the subjects were included in the analysis.

Data Availability. Anonymized csv data have been deposited in GitHub (https://github.com/adezfouli/decision_adv).

ACKNOWLEDGMENTS. We are grateful to Yonatan Loewenstein for discussions. P.D. was funded by the Max Planck Society and the Humboldt Foundation. This research was funded partially by CSIRO's Machine Learning and Artificial Intelligence Future Science Platform.

1. O. Dan, Y. Loewenstein, From choice architecture to choice engineering. *Nat. Commun.* **10**, 2808 (2019).
2. A. Dezfouli, R. W. Morris, F. Ramos, P. Dayan, B. W. Balleine, "Integrated accounts of behavioral and neuroimaging data using flexible recurrent neural network models" in *Advances in Neural Processing Systems 31*, S. Bengio et al., Eds. (Curran, Red Hook, NY, 2018), pp. 4233–4242.
3. A. Dezfouli *et al.*, "Disentangled behavioral representations" in *Advances in Neural Processing Systems 32*, H. Wallach et al., Eds. (Curran, Red Hook, NY, 2019), pp. 2243–2252.
4. A. Dezfouli, K. Griffiths, F. Ramos, P. Dayan, B. W. Balleine, Models that learn how humans learn: The case of decision-making and its disorders. *PLoS Comput. Biol.* **15**, e1006903 (2019).
5. C. Szegedy *et al.*, Intriguing properties of neural networks. arXiv:1312.6199 (21 December 2013).
6. Y. C. Lin *et al.*, Tactics of adversarial attack on deep reinforcement learning agents. arXiv:1703.06748 (8 March 2017).
7. K. S. Jun, L. Li, Y. Ma, J. Zhu, "Adversarial attacks on stochastic bandits" in *Advances in Neural Processing Systems 31*, S. Bengio et al., Eds. (Curran, Red Hook, NY, 2018), pp. 3644–3653.
8. I. W. Eisenberg *et al.*, Uncovering the structure of self-regulation through data-driven ontology discovery. *Nat. Commun.* **10**, 2319 (2019).
9. B. King-Casas *et al.*, Getting to know you: Reputation and trust in a two-person economic exchange. *Science* **308**, 78–83 (2005).
10. V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning" in *Proceedings of the 33rd International Conference on Machine Learning*, M. F. Balcan, K. Q. Weinberger, Eds. (Curran, Red Hook, NY, 2016), Vol. 48, pp. 1928–1937.
11. V. Mnih *et al.*, Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
12. K. A. McCabe, M. L. Rigdon, V. L. Smith, Positive reciprocity and intentions in trust games. *J. Econ. Behav. Organ.* **52**, 267–275 (2003).
13. A. Hula, P. R. Montague, P. Dayan, Monte Carlo planning method estimates planning horizons during interactive social exchange. *PLoS Comput. Biol.* **11**, e1004254 (2015).
14. A. Tversky, D. Kahneman, Judgment under uncertainty: Heuristics and biases. *Science* **185**, 1124–1131 (1974).
15. S. Lange, T. Gabel, M. Riedmiller, "Batch reinforcement learning" in *Reinforcement Learning*, M. Wiering, M. van Otterlo, Eds. (Adaptation, Learning, and Optimization Series, Springer, 2012), Vol. 12, pp. 45–73.
16. J. X. Wang *et al.*, Learning to reinforcement learn. arXiv:1611.05763 (17 November 2016).
17. Y. Duan *et al.*, RL^2: Fast reinforcement learning via slow reinforcement learning. arXiv:1611.02779 (9 November 2016).
18. J. H. Bak, J. Y. Choi, A. Akrami, I. Witten, J. W. Pillow, "Adaptive optimal training of animal behavior" in *Advances in Neural Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, R. Garnett, Eds. (Curran, Red Hook, NY, 2016), pp. 1947–1955.
19. S. Fujimoto, H. Van Hoof, D. Meger, "Addressing function approximation error in actor-critic methods" in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy, A. Drause, Eds. (Curran, Red Hook, NY, 2018), Vol. 80, pp. 1582–1591.
20. Z. Cranko *et al.*, "Monge blunts Bayes: Hardness results for adversarial training" in *Proceedings of the 36th International Conference on Machine Learning* (Curran, Red Hook, NY, 2019), Vol. 81, pp. 2523–2543.
21. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. arXiv:1412.6980 (22 December 2014).
22. M. Abadi *et al.*, TensorFlow: Large-Scale machine learning on heterogeneous systems. arXiv:1603.04467 (14 March 2015).
23. K. Cho *et al.*, Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078 (3 June 2014).
24. Z. Ahmed, N. Le Roux, M. Norouzi, D. Schuurmans, "Understanding the impact of entropy on policy optimization" in *Proceedings of the 36th International Conference on Machine Learning* (Curran, Red Hook, NY, 2019), Vol. 81, pp. 151–160.