A Related work

Our work joins a growing number of studies that leverage computational tools for automated mechanism design [11, 65], the problem of utilizing computational approaches or learning-based techniques for finding revenue-maximizing mechanisms in auction settings. One strand of works [10, 66] in this line of research has focused on using learning approaches for mechanism design where only samples of bidder valuations are used to design revenue-maximizing mechanisms. More recently, deep neural networks has been utilized for the automated design of optimal auctions [20], in which the authors propose multiple neural-network architectures for learning approximately optimal auctions. Several works has extended this study in various applications [27, 31, 13, 42, 61, 58, 45, 12]. Our work differs from this line of works in two ways. First, to the best of our knowledge, we are the first to address the automated information design problem. Second and more importantly, we have incorporated human behavior descriptors in our design, while prior works mostly require standard rationality assumptions.

Our information design formulation builds on top of the seminal work of Bayesian persuasion [39], which initiated a rich theoretical literature on communication games in which a sender can design information to persuade a receiver to take certain actions. Their work has provided theoretical foundations and inspired an active line of research in information design [e.g., see the recent surveys by 38, 4]. Our work builds on top of this line of work through integrating human behavior in the design of information policy, while existing works mostly assume the receiver is Bayesian rational. In particular, our proposed HAIDNet can dynamically adjust to various forms of model-based or data-driven human behavior descriptors. For the model-based receiver behavior, as an example, we have included the probability weighting function [77, 60, 64] for belief updating and the discrete choice model [52, 68, 73] for decision making under uncertainty. Non-Bayesian belief updating in information design also appears in earlier works [14], and the receiver's behavior following the discrete choice model also appears in previous works [70, 26]. Our work generalizes the above in that our framework can adapt to both the above form and the data-driven form of human behavior.

The problem of information design and persuasion has received increasing attention both in research and in practice. For example, researchers have argued that one-quarter of the GDP in the United States is persuasion [51]. Due to its practical relevance, this problem is also getting attention more broadly in the general research community, as demonstrated by the recent papers in machine learning and artificial intelligence venues, studying various problem settings such as in security [79], human language interactions [2], data marketplace design [8], algorithmic recourse [33], online recommendation [25], and market competitions [15]. Our work joins this line of study and aims to develop more efficient approaches for information design under more realistic settings of human behavior.

On a conceptual level, our work is related to the growing attention in understanding, modeling, and accounting for human behavior in computational systems, especially in the context of human-robot or human-AI interactions [9, 67, 43, 7, 63, 54, 55, 74]. Moreover, our work joins the recent research theme that incorporates human models in computational and machine learning frameworks [28, 47, 69, 70, 41, 49, 50, 72, 80]. There have been other lines of research that includes humans in the loop of learning frameworks, such as inverse reinforcement learning [56, 24, 67, 36, 81] that infers the reward functions in Markov decision process through (potentially human) demonstrations. Our work differs in that we focused on the information design problem with realistic human receiver models.

Lastly, in this study, we incorporate insights from human behavior into information design. Extensive literature from psychology and behavioral economics has been devoted to deepen our understanding of human behavior. Examples include studies examining deviations from the standard Bayesian assumption in processing information [57, 40, 3] and the rationality assumption in decision-making [37, 52, 68, 73, 35]. While these classical models, often grounded in human data from behavioral experiments [48, 16, 17], offer interpretable behavioral insights, they tend to lack in terms of predictive accuracy. Recently, given the advancements of machine learning techniques and the avaialability of a larger amount of human data, there has been a growing effort to integrate behavioral insights from these classical models with machine learning techniques to enhance predictive accuracy [5, 59]. These models developed in this line of effort are directly applicable in our framework. Moreover, as outlined in Section 5, integrating human behavioral insights into information design can raise concerns about exploiting human irrationality. One potential solution is to incorporate the concept of differential privacy [22, 21, 71]. This would control the amount of personalized information that can be used, preventing undue exploitation.

B Experimental results and details

In this section, we discuss additional sets of simulation results to highlight the properties and performance of HAIDNet. We also provide details of the optimization process of HAIDNet.

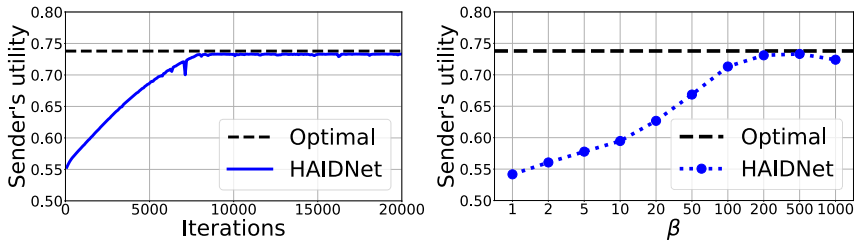
B.1 Additional experiment results

B.1.1 Convergence of training

In this set of simulations, we have examined the convergence of training with respect to the number of training iterations and also with respect to the softmax parameter β when dealing with Bayesian rational receivers. Overall, HAIDNet converges to finding the optimal policy within reasonable setup.

To illustrate the results, here we present the simplest setting with binary actions and binary states, namely, the action space $\mathcal{A} = \{0, 1\}$ and the state space $\Theta = \{0, 1\}$, and observe whether HAIDNet can produce near-optimal information policies. For the sender utility, we adopt a stylized setting where the sender obtains utility 1 when the receiver takes action 1 and utility 0 when the receiver takes action 0. We randomly draw each value in the receiver utility u^R from $[0, 1]$. The prior distribution $\boldsymbol{\lambda}$ is drawn from a Dirichlet distribution. We then simulate data using the setting above and optimize HAIDNet.

We compare the performance of the policy learned by HAIDNet with the closed-form optimal policy. Recall that when the receiver is rational (expected utility maximizer), he chooses the action that maximizes his expected utility given his belief about the state. As introduced in Section 3.2, to enable the gradient-based method in optimizing HAIDNet, we replace this *argmax* operation as *softmax* using a softmax scale parameter β. Therefore, we first examine the impact of this choice of β and the amount of training (# iterations in gradient descent) in optimizing the information policy. As shown in Figure 5, when β is large enough and when we optimize over a large enough number of data batches, the learned information policy from HAIDNet converges to the information policy that achieves near-optimal performance.



(a) Training iterations. (b) β.

Figure 5: The convergence results, with respect to # training iterations and β, of the sender's utility derived from the information policy generated by HAIDNet.

B.1.2 Scalability: Empirical run-time comparison

One of the benefits of HAIDNet is to provide efficient solutions for settings when it is computational challenging to derive the optimal policy exactly (e.g., in settings with multiple receivers).

To demonstrate this benefit empirically, we first record the time for computing the exact optimal policy for a problem instance with K receivers via a Linear Programming approach [19]. As we can see from Table 5, the amount of time to compute the optimal information policy grows significantly (the computational complexity grows exponentially as the number of constraints is exponential in the number of receivers in the linear programming approach) as the number of receiver increases. This reaffirms the computational barriers to computing the exact optimal policy. Note that Xu [78] has shown that it is #P-hard to approximate the optimal sender utility within any constant multiplicative factor. So this computational barrier is backed by theoretical analysis.

For HAIDNet, for each class of problems (i.e., a given number of receivers), we only need to train HAIDNet once. For each new problem instance (different priors, sender/receiver utilities, etc), we only need to make a test-time prediction (one pass of forward propagation) to generate an information

policy. Again, in Table 5, we report the time for training HAIDNet and the time for generating the information policy for each problem instance. To provide the number comparisons, when the number of receivers is 18, traditional linear program method of solving the information policy for a problem instance takes more than 23 hours. On the other hand, for HAIDNet, we only need a little more than 1 hour to train HAIDNet for all problem instances with 18 receivers, and it takes less than 1 second to generate the information policy for each receiver. The reported numbers are performed on the machines with Intel(R) Xeon(R) Gold 6148 CPU (2.40GHz) and a Tesla V100-SXM2-32GB GPU.

Table 5: Comparing run-time between HAIDNet and linear programming methods. K is the number of receivers. The reported run-times are in seconds.

K	Training Time of HAIDNet	Testing time per instance of HAIDNet	Optimal policy per instance via Linear Programming
2	1082	0.184	0.323
3	1291	0.189	0.367
5	1571	0.221	0.371
10	2174	0.270	4.820
15	3284	0.299	235.0
17	3713	0.333	14290
18	4030	0.352	84280

B.1.3 Additional results for a single Bayesian rational receiver

In Section 4.1.1, we compare the performance between the policy from HAIDNet and the optimal policy in the single Bayesian rational receiver setting with an increasing number of states with binary actions, and an increasing number of actions with binary states. To further complete the results, we have also run simulations when we simultaneously increase the number of actions and the number of states at the same time. To put the performance of HAIDNet into context, we also include the performance of random policy, which provides random signals all the time. This random policy serves as the naive baseline setting. The results are shown in Table 6c. The average sender utility obtained by HAIDNet policy is close to optimal policy in both training and testing problem instances (averaged over 1,000 instances) even in cases with large action and state numbers. We also evaluated the model training error for binary action case and binary state case in Table 6, which shows that HAIDNet works well for large-scale problem instances.

B.1.4 Additional results for multiple receivers and non-Bayesian-rational receivers

Due to space constraints, we do not include all results in these two settings in the main paper. Here we provide the full results, including the performance of random baseline as well. The results for settings with multiple receivers are included in in Table 7, and the results for settings with a single non-Bayesian-rational receiver are included in Table 8. HAIDNet performs well in multiple receiver settings and non-Bayesian-rational receiver settings.

Recall that for all the results presented in this work, both the training and testing performance are the average performance for $1,000$ data points sampled from the training and testing datasets.

B.1.5 Varying number of receivers, actions and states

In our main paper, each HAIDNet can accommodate any problem instance (i.e., different specifications of priors, receiver utility, and sender utility) with a fixed number of actions, states, and receivers. It is then natural to wonder whether we can extend the HAIDNet structure so that it can work with varying numbers of receivers, actions, and states. As a proof of concept, in this set of simulations, we attempt to address this question and present an approach that can work with varying numbers of receivers, actions, and states when the numbers are upper bounded.

We first examine the relaxation of a fixed number of receivers. In particular, we can generalize our approach to address varying numbers of receivers when the number of receivers is upper bounded. One straightforward approach is to maintain multiple HAIDNet, one for each fixed number of receivers, for generating the optimal information policy. Another approach is to train a HAIDNet that can generate information policy for the maximum number of receivers. In settings when the

Table 6: Comparing the average sender utility generated by the optimal policy and the policy from HAIDNet in the setting with a single Bayesian rational receiver.

(a) Increase the number of states M with binary actions.

M	Training			Testing		
	Random	HAIDNet	Optimal	Random	HAIDNet	Optimal
2	0.4901	0.7409	0.7498	0.4909	0.7408	0.7451
3	0.5009	0.7737	0.7782	0.4819	0.7598	0.7669
5	0.4898	0.8171	0.8209	0.5227	0.8066	0.8225
10	0.4841	0.8495	0.8699	0.4838	0.8196	0.8686

(b) Increase the number of actions N with binary states.

N	Training			Testing		
	Random	HAIDNet	Optimal	Random	HAIDNet	Optimal
2	0.4901	0.7409	0.7498	0.4909	0.7408	0.7451
3	0.4911	0.7017	0.7214	0.5064	0.7089	0.7227
5	0.4919	0.6906	0.7113	0.5119	0.6690	0.7064
10	0.4907	0.6861	0.7084	0.4861	0.6623	0.6963

(c) Increase both the number of states and actions. $M = N$ represents state number equals action number.

$M = N$	Training			Testing		
	Random	HAIDNet	Optimal	Random	HAIDNet	Optimal
2	0.4901	0.7409	0.7498	0.4909	0.7408	0.7451
3	0.4791	0.7352	0.7679	0.4854	0.7199	0.7587
5	0.5029	0.7771	0.8113	0.5101	0.7755	0.8121
10	0.4799	0.8613	0.8971	0.4872	0.8323	0.8994
50	0.4903	0.9247	0.9550	0.7058	0.9166	0.9545

Table 7: Comparing the average sender utility generated by the optimal policy and the policy from HAIDNet in the setting with K Bayesian rational receivers.

K	Training			Testing		
	Random	HAIDNet	Optimal	Random	HAIDNet	Optimal
2	0.5158	0.7887	0.7934	0.5195	0.7756	0.7873
3	0.5050	0.7508	0.7665	0.4898	0.7379	0.7573
5	0.4920	0.7217	0.7458	0.4980	0.7209	0.7570
10	0.5063	0.6971	0.7152	0.5192	0.6790	0.6966
15	0.5007	0.6553	0.6882	0.4841	0.6621	0.6843
17	0.5037	0.6166	0.6503	0.5004	0.6160	0.6497

Table 8: Comparing the average sender utility by the optimal policy and the policy from HAIDNet in the setting with a non-Bayesian-rational receiver parameterized by β_H.

β_H	Training			Testing		
	Random	HAIDNet	Optimal	Random	HAIDNet	Optimal
1	0.4986	0.5043	0.5051	0.5006	0.5041	0.5060
5	0.4929	0.5512	0.5557	0.5036	0.5506	0.5559
10	0.4904	0.6045	0.6170	0.5064	0.5986	0.6168
50	0.4919	0.7002	0.7134	0.5093	0.6800	0.7081
100	0.4931	0.7187	0.7291	0.5097	0.6964	0.7179
500	0.4946	0.7418	0.7468	0.5096	0.7354	0.7396

number of receivers is less than the maximum number, we can include "null receivers" who always choose action 0 (by setting the receiver utility such that the utility for taking action 0 is always larger than taking other actions in both states). By including this in the training process, we can have a single HAIDNet that can generate policies for a bounded variable number of receivers. As a proof of concept, we have implemented the above approach and trained a HAIDNet that can work with up

to 10 receivers. We then examine its performance when the number of receivers is smaller than 10. As we can see from Table 9, this approach achieves reasonable performance and shows promising results.

Table 9: Comparing the average sender utility by the optimal policy and the policy from HAIDNet in the setting with at most 10 Bayesian rational receivers.

K	Training			Testing		
	Random	HAIDNet	Optimal	Random	HAIDNet	Optimal
2	0.5042	0.7830	0.8018	0.4986	0.7538	0.7921
3	0.5066	0.7337	0.7586	0.4866	0.7139	0.7450
5	0.5032	0.7245	0.7451	0.5071	0.7121	0.7387
10	0.4944	0.6911	0.7118	0.5009	0.6650	0.6901

We now examine whether this approach also works for extending the number of states M and the number of actions N. As a proof of concept, we adopt the same approach above and train a HAIDNet for a maximum of 5 actions and 5 states. We then examine the performance of HAIDNet for problem instances with less or equal to 5 actions or states. As shown in Table 10, this approach also works in addressing varying numbers of actions and states.

Table 10: Comparing the average sender utility by the optimal policy and the policy from HAIDNet in the setting with at most 5 states and 5 actions, for single Bayesian rational receivers.

(M, N)	Training			Testing		
	Random	HAIDNet	Optimal	Random	HAIDNet	Optimal
(2, 3)	0.4994	0.6564	0.7308	0.5276	0.6517	0.7411
(2, 4)	0.4852	0.6450	0.7134	0.5236	0.6535	0.7329
(2, 5)	0.4898	0.6498	0.7042	0.5111	0.6641	0.7258
(3, 2)	0.5094	0.6856	0.7735	0.4731	0.6462	0.7574
(3, 3)	0.5128	0.7072	0.7791	0.4832	0.6689	0.7615
(3, 4)	0.5343	0.7165	0.7729	0.5322	0.6940	0.7672
(3, 5)	0.4798	0.6922	0.7453	0.5308	0.6990	0.7492
(4, 2)	0.4898	0.6849	0.7701	0.5216	0.6922	0.7883
(4, 3)	0.4721	0.7051	0.7761	0.4796	0.6940	0.7844
(4, 4)	0.5032	0.7239	0.7812	0.5143	0.7186	0.7962
(4, 5)	0.4700	0.7347	0.7807	0.5144	0.7421	0.7925
(5, 2)	0.4883	0.7147	0.7915	0.5186	0.7137	0.8038
(5, 3)	0.5394	0.7736	0.8398	0.4928	0.7318	0.8184
(5, 4)	0.4998	0.7810	0.8289	0.4951	0.7494	0.8242
(5, 5)	0.4819	0.7722	0.8159	0.4863	0.7605	0.8079

B.2 Data generation

Here we provide the details in generating data instances for training HAIDNet in our settings.

Single receiver, binary actions and binary states. In the simplest setting with binary actions and binary states, the action space is $\mathcal{A} = \{0, 1\}$ and the state space is $\Theta = \{0, 1\}$. We adopt a stylized setting for binary actions where the sender obtains utility 1 when the receiver takes action 1 and utility 0 when the receiver takes action 0 [39]. The receiver utility u^R is uniformly drawn from $[0, 1]$ and prior distribution is draw from Dirichlet distribution. We filter out trivial problem instances where the receiver will always choose one action whatever the information policy, e.g., the receiver always chooses action 1 when receiver utility $u^R(1, \theta) > u^R(1, \theta), \forall \theta \in \Theta$. Total 102,400 instances are generated for training, 1,000 for validation and 1,000 for testing.

Single receiver, multiple actions, and multiple states. In the setting with N actions and M states, the action space is $\mathcal{A} = \{0, 1, \ldots, N-1\}$ and the state space is $\Theta = \{0, 1, \ldots, M-1\}$. The sender utility is set to $u^S(a, \theta) = \frac{a}{N-1}, \forall \theta \in \Theta$ if $N \geq 3$, and the same as above binary actions if $N = 2$. The receiver utility u^R is uniformly drawn from $[0, 1]$ and prior distribution is drawn from Dirichlet distribution. We also filter out trivial cases where the receiver will always choose one action whatever the information policy is.

Multiple receivers, binary actions, and binary states. The receiver utility and prior distributions are generated in the same way as in the cases of a single receiver, binary actions, and binary states. The sender utility is the fraction of receivers choosing action 1, i.e., her utility is given $\frac{|S|}{K}$ if there are $|S|$ number of receivers choosing action 1 and K is the total number of receivers. We also filter out trivial cases where the receiver will always choose one action whatever the information policy is.

Problem instances in human-subject experiments. In our human-subject experiment, the problem setup is the same as the setting with a single receiver, binary actions, and binary states. To make the setting easier to understand for experiment participants, the receiver utility is drawn from $\{1, 2, 3, 4, 5\}$ when the participant chooses to purchase a good product or chooses to not purchase a bad product, and the participant utility is 0 for other cases. The sender utility is set to 1 when the receiver chooses to buy, and 0 otherwise. The prior distribution is drawn from the Dirichlet distribution, however, we round all probability in the prior distribution and the information policy to the nearest tenth digit, $\{0\%, 10\%, \ldots, 100\%\}$, to make it easier to interpret for human participants.

B.3 HAIDNet optimization procedures

Here we provide more detailed parameter setups for our simulations and human subject experiments.

Optimizing HAIDNet in simulations. We build a 3 fully connected layer neural network with ReLU activation functions as the sender's optimization module in HAIDNet. Network parameters are initialized by Glorot uniform initializer. When optimizing HAIDNet, we use the Adam optimizer and batch gradient descent. Batch size is 1,024, batch number is 100, and maximum training epoch (each epoch contains 100 batches) is 1,000.

The hyperparameters are tuned by using the validation dataset. We then report the performance on the test dataset. The number of nodes for each hidden layers is tuned in the range of $\{64, 128, 256, 512, 1024\}$, and the initial learning rate is tuned in the range of $\{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1\}$. When the human descriptor is Bayesian rational, we use softmax to smoothen the argmax operator. Empirically we find directly assigning large value to β leads to bad performance. So instead, we increase β gradually from 10 to 1000 exponentially in first 100 epochs of training, that is $\beta_i = 10^{1+\frac{i}{50}}$ in i'th epoch, and maintain $\beta = 1000$ for remaining training process. In the setting with multiple actions and multiple states, softmax approximation of Bayesian rational behavior leads to higher errors. We further adapt the approach of iteratively training networks, reweighting data distributions, and aggregate learned neural networks to reduce the error. Empirically, aggregating three models are enough to reach promising performance.

Optimizing HAIDNet in human-subject experiments. After collecting human responses in the first phase of human experiments, we fit TH-Model and train a neural network model for human behavior descriptors. β_H in TH-Model is fitted by minimizing a square error between model prediction and human data, and $\beta_H = 20$ fits the best. For the neural network model, we use a 3 fully connected-layer neural network with ReLU activator. We split the data into training/testing sets, with 80% of the data for training, and 20% of the data for testing. We further split the training dataset and use 25% of the training dataset as a validation set to implement early-stopping during training. The neural network for fitting human behavior is trained with batch number 12, batch size 100, and maximum epoch 100. The number of nodes for each hidden layers is tuned in $\{16, 32, 64, 128, 256, 512, 1024\}$ and the initial learning rate is tuned in $\{0.001, 0.002, 0.005, 0.01\}$. We select the hyper-parameter based on average performance of validation sets. Because of 5-fold splitting, we have 5 trained neural networks for human descriptors, and we train HAIDNet corresponding to each of the human model. The learned policy are close in terms of expected utility, in range of $[0.697, 0.726]$, and we select the model of the highest performance in simulation to design the information policy in the second phase experiment.

C Details of Human-Subject Experiments

We provide more detailed information about our human-subject experiments here. We compare average sender utility of different policies in human-subject experiment in Figure 4, and we also compute the receiver utility in each treatment, included in Table 11. As we can see from the table, while HAIDNet helps find a policy that leads to the highest sender utility, it comes at the cost of reducing the receiver utility, a demonstration of the ethical concerns as discussed in Section 5.

Table 11: Comparing sender and receiver utility of different policies in human-subject experiments.

Information Policy	Random	BR-Policy	TH-Policy	HAIDNet
Sender Utility	0.489	0.524	0.621	0.667
Receiver Utility	0.663	0.634	0.565	0.532

In our experiment setup, given the sender's goal is to have the receiver purchase the products regardless of the product quality, when the sender is more successful, it leads to a lower receiver utility in general and implies the potential negative social impacts.

C.1 Demographic of Workers

We have recruited 300 workers from Amazon Mechanical Turk in total for our experiments. Table 12 contains the demographic information of the 300 workers.

Table 12: Demographic information of the participants in our experiment.

Group	Category	Number
Age	20 to 29	88
	30 to 39	111
	40 to 49	65
	50 to 59	25
	60 or older	11
Gender	Female	131
	Male	168
	Other	1
Race / Ethnicity	Caucasian	240
	Black or African-American	18
	American Indian/Alaskan Native	5
	Asian or Asian-American	22
	Spanish/Hispanic	6
	Other	9
Education	High school degree	12
	Some college credit, no degree	9
	Associate's degree	24
	Bachelor's degree	223
	Graduate's degree	29
	Other	3

C.2 Task Interface and Description

In our human-subject experiments, we simulate the setting with binary actions and binary states. In particular, we present the product purchasing example as we discussed in Section 2. The task interface about our human-subject experiments is shown in Figure 6.

Each human participant is asked to make multiple rounds of purchase decisions. In each round, the participant is presented a product with unknown binary quality (either good product or bad product). The participant is told that a (noisy) inspection has been performed on the product, and is given the conditional distribution associated with the inspection (i.e., the probability to receive a good/bad signal given the product is good/bad). Finally, the participant is given a realization of the inspection signal and is asked to make a binary decision of purchasing or not. The participant's reward depends on both their purchasing decisions and the true product quality. When collecting human response in the first phase, random policy are presented to all participants. In the second phase, different policies are presented: {Random, BR-policy, TH-policy, HAIDNet policy}. The policies are designed with the assumption that the sender is persuading human receivers to purchase the product, and we calculate the probability of participants choosing to purchase and report it as the sender utility to evaluate performance of different policies.

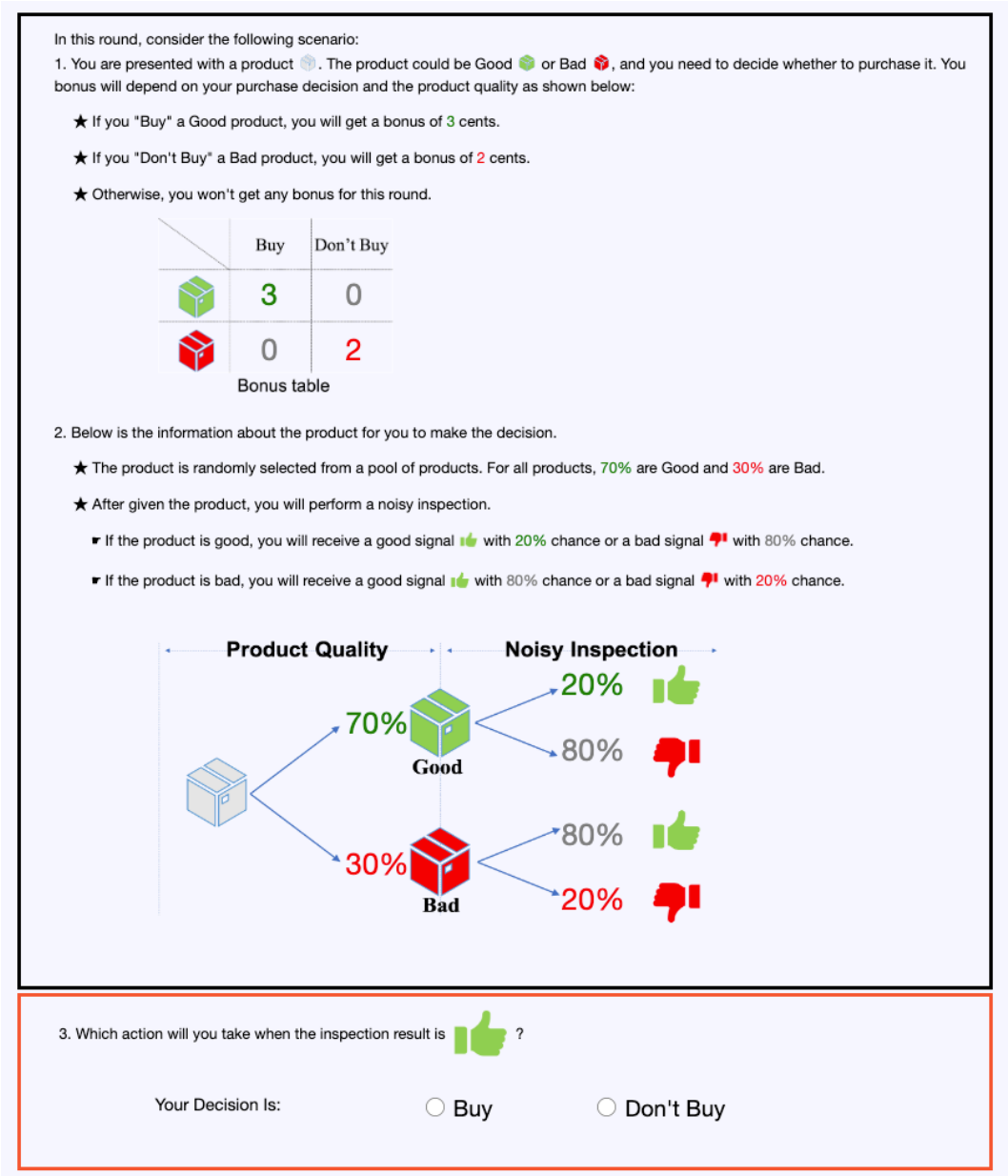


Figure 6: Human experiment interface.