Replicating Human Bias Through Synthetic Data Generation Using Deep Learning

Bryan Brandt

A Thesis Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science

Graduate Program in Computer Science

University of Wisconsin-Whitewater
Whitewater, Wisconsin

December 12, 2023

Graduate Studies

The members of the Committee approve the thesis of

Bryan Brandt presented on 12/01/2023

_____

Dr. Hien Nguyen, Chair

_____

Dr. Prithviraj Dasgupta

_____

Dr. Arnab Ganguly

The author would like to thank the Office of Naval Research for supporting some of the research reported in this thesis through an NREIP summer internship with the Naval Research Laboratory, Washington, DC in summer 2023 and 2024. The research was done as part of the NRL 6.1 Base Funding project titled "Playing Games to Overcome Cognitive Biases in Warfighter Decision Making" (PI: Dasgupta)

The author would also like to thank the graduate school of University of Wisconsin Whitewater for funding this research through the Graduate Research Grant.

Abstract

The growth of Artificial Intelligence (AI) emphasizes the need for greater understanding of human and AI interactions. Human models formed during interactions with AI are subjected to cognitive biases, warranting further study to improve joint performance. Mitigating bias involves real-time bias detection; however, the exigency of extensive training data poses challenges, particularly in research contexts where such data sets may be unavailable. We propose an algorithm that amalgamates reward shaping, reinforcement learning, and imitation learning to emulate human game play data with sparse training data. This algorithm is evaluated in two experiments: the first successfully replicates human game play data utilizing diminutive training data, and the second extends the initial experiment by accurately replicating the anchoring effect cognitive bias from human game play data. Conclusions and suggestions for future research are discussed.

Contents

List of Tables

List of Figures

1 Introduction

The growing accessibility of artificial intelligence underscores the need for a more profound comprehension of the interaction between humans and AI. Trust in AI models is a crucial component for the performance of human-AI teams [1]. In an ideal scenario, trust should align with the reliability of AI systems. However, trust is dynamic, and influenced by various contextual and individual factors [2]. Trust examination is contingent on context, and outcomes vary based on the type of system under scrutiny [3]. Recent research emphasizes enhancing AI accessibility through transparent and interpretable systems, especially in human-AI collaborations [4]. Additionally, by providing details about model predictions aids users in decision-making, fostering a comprehensive understanding of AI operation and training data, shaping user perceptions during decision processes [5]. AI performance explanations without feedback can lead to frustration, highlighting the need for interactive feedback to enhance the user experience [6]. Conversely, the mental models developed by humans in collaboration with AI impact team performance, emphasizing the importance of human-centered training for optimal performance [7, 8, 9]. However, mental models can be subject to human discrepancies, such as cognitive biases [10].

Cognitive biases, predictable deviations from reality, affect trust and performance, particularly in collaborative environments [11, 12, 13, 14]. These biases play a significant role in designing Explainable AI (XAI) methods, and influencing their evaluation. XAI techniques have the potential to mitigate some biases but may inadvertently exacerbate others, emphasizing the need for careful design principles [15]. Detecting and mitigating cognitive biases is pivotal for cultivating trustworthy human-AI collaborations, which highlights the reliability of AI decisions [16, 8]. Particular biases, such as anchoring bias, can detrimentally affect team performance, particularly in cases where AI predictions prove to be incorrect [16]. Therefore, integrating bias mitigation strategies is essential in the design of human-AI interactions.

Mitigating cognitive bias involves various strategies, categorized as training interventions and procedural interventions [17, 18]. Training interventions aim to educate individuals about bias, but their effectiveness is limited. Procedural interventions detect and mitigate bias in real-time, showing promise in various contexts. Research has shown that visual representation of data can trigger cognitive biases, leading to judgment errors [19]. Additionally, modifying interactive visualization tools and presenting prevalence data strategically can mitigate biases such as the attraction effect and anchoring bias [20, 21, 22, 23]. Furthermore, increasing user interaction in machine learning UIs, incorporating domain knowledge, and presenting conflicting information proves valuable in countering biases [24]. However, bias mitigation can only begin when biases are detected.

Detecting and replicating cognitive bias involve various methods and tools. Software interactions have shown to detect and quantify cognitive biases using Markov Decision Processes, Markov Chains, adaptive contextualization methods, and by tracing user interaction histories [25, 26, 27, 28, 29]. Moreover, real-time application of computational metrics on user interaction sequences provides insights into bias using convergence and distribution across data points [30]. However, challenges in machine-centric processes, including data acquisition, diverse variable sampling, and dynamic variable determination, necessitate caution despite advancements in data collection capabilities [31, 32], suggesting the necessity of data scrutinization prior to model training.

The replication of cognitive biases in AI models is intricately tied to the quality and nature of the training data employed [33]. This training data serves as a foundational source that influences how AI models perceive and learn patterns. If the training data inherently contains biases, the model is likely to replicate and even exacerbate these biases in its predictions and decision-making processes [34]. Understanding the origin and composition of training data is essential for comprehending and addressing bias in AI systems. Biases in training data can stem from various sources, including historical disparities, cultural prejudices, and societal inequalities reflected in the data collection process [35, 36]. The challenges lie in acquiring representative data sets that encapsulate the complexity of real-world scenarios and minimize extraneous bias propagation while retaining a singular bias for study. This presents a dilemma in obtaining contextually pertinent data from current data sets, which might already exhibit bias, that compels researchers to procure new data, aligning with their area of bias investigation.

Obtaining sufficient quantities of high-quality data for machine learning (ML) training poses a significant challenge, requiring data to be well-structured, annotated, complete, reliable, and consistent. Many application domains, particularly those relevant to the military and academia, exhibit characteristics such as sparse, unlabeled, and sometimes incomplete data. To address these limitations, researchers have proposed techniques like data augmentation [37, 38] to generate data suitable for ML training and testing. However, existing research on data augmentation primarily focuses on image and tabular data, with lesser attention given to other modalities like text, audio, and human-decision making data. Generating synthetic human interaction data, especially interactions between humans and machines, has received comparatively less attention. Acquiring data related to human actions and behavior introduces additional challenges, including the difficulty of recruiting human participants, the need for extensive training to develop skills for generating reliable data, and the challenge of ensuring the quality of human-generated data without expert inspection and validation. To address these issues, exploring techniques for synthetically generating

data that closely replicates human decision-making behavior, starting with a limited set of human-generated decision-making data, becomes essential.

In this paper, we address existing research gaps by presenting a novel method for synthesizing data through imitation learning within the context of computer game-playing. Our approach involves employing reward shaping [39] to train a reinforcement learning agent, utilizing a limited set of human decisions derived from game-play data. This initial dataset is further refined using the `DAgger` algorithm [40], resulting in the creation of a final set of synthetically generated trajectory data.

Our research hypotheses are structured as follows:

1. **Training Reinforcement Learning Agents with Sparse Human Data:** Can a reinforcement learning agent be effectively trained using a sparse set of human-generated data to accomplish a decision-making task?

2. **Low Divergence Synthetically Generated Trajectories:** Can synthetic trajectories be generated with minimal divergence from human-generated trajectories using the `DAgger` algorithm?

3. **Impact of Human Demonstrator Data on Synthetic Trajectory Quality:** What is the influence of incorporating human demonstrator data into the `DAgger` algorithm on the quality of synthetically generated trajectories?

4. **Eliciting Anchoring Bias in a Computer-Based Game Environment:** Can the anchoring effect, a human bias, be induced within a computer-based game environment?

5. **Replicating Biased Human Gameplay with `DAgger`:** Can the `DAgger` algorithm faithfully replicate biased human gameplay, specifically concerning the anchoring effect?

Through these research inquiries, we aim to contribute valuable insights to the domain of synthetic data generation and its applications in training reinforcement learning agents, addressing challenges and potential biases associated with human-AI interactions in gaming scenarios.

This thesis commences by presenting the literature review in Chapter 2. This section delineates the relevant studies, theoretical background, and rationale for the algorithms utilized, as well as the cognitive bias incorporated in the experiment. Chapter 3 provides an overview of our methodology, encompassing the mathematical framework of our environments and reinforcement learning agents. Additionally, it expounds on the algorithms employed to train our reinforcement learning agents, particularly our adapted `DAgger` algorithm, and elucidates the metrics used for evaluation.

In Chapter 4, we delve into the details of the first experiment, addressing the initial three research questions. This includes a description of the game environments, our procedural approach, and the results concerning synthetic trajectory generation. Chapter 5 extends the exploration with the second experiment, introducing a distinct gaming environment, tailored to elicit a cognitive bias. It investigates whether this bias can be replicated using reinforcement learning, addressing the last two research questions. Finally, Chapter 6 outlines the conclusions drawn from the two experiments, discusses limitations, and proposes avenues for future research.

2 Literature Review

Our study introduces an algorithm that integrates reward shaping with reinforcement learning (RL) to train a Deep Q Network (DQN) agent, aiming to emulate human demonstrator game-play decisions through two experimental iterations. Following this training, the DQN agent serves as an "expert agent" in the `DAgger` imitation learning algorithm, facilitating the generation of synthetic trajectories closely aligned with those produced by human demonstrators. In the second experiment, our objective was to extend the initial experiment by inducing the anchoring effect cognitive bias and assess the algorithm's ability to replicate biased game-play decisions. The subsequent sections provide comprehensive details on the algorithms, mathematical framework, and cognitive bias.

2.1 Data Augmentation

Generating novel data can involve the application of data augmentation methods, which encompass strategies for introducing unobserved data or latent variables into algorithms aimed at improving dataset size and quality [41, 42]. A frequently utilized technique is the implementation of generative adversarial networks (GAN) a type of machine learning model composed of two neural networks: a generator and a discriminator. The generator is responsible for producing data, while the discriminator's role is to distinguish between real and generated data. These networks engage in a competitive training process, where the generator continually improves its ability to create realistic data, and the discriminator enhances its skill in telling real data from fake [43]. A widely used GAN framework for data augmentation is the conditional GAN (cGAN) where the data is input into the generator and discriminator and can generate MNIST digits conditioned on class labels [44]. The cGAN concept has been extended with different techniques including multiple loss functions in the Auxiliary Conditional GAN (ACGAN) [45] and CycleGAN [46], using an autoencoder as generator in Data Augmentation GAN (DAGAN) [47] and Balancing GAN (BAGAN) [48], multiple generators called Siamese networks (Siamese-cGAN) [49] to generate synthetic data. Most data augmentation research has been in the field of computer vision for generating synthetic image data using a base set sampled from public image data sets. Recently, data augmentation techniques have been applied to other data modalities including medical and agricultural imagery, [50, 51, 52], urban vehicle movement data [53], tabular relational data [54, 55], vibration data from a mechanical sensor [56] and cryptography attack data [49]. Nevertheless, in scenarios with sparse training data, relying solely on data augmentation techniques proves inadequate, necessitating the synthesis of novel data.

2.2 Synthetic Data Generation

Synthetic data generation involves replicating data to create novel data sets with statistical properties resembling the original data [57]. In the medical domain, synthetic data is used to model patient health information when direct access is restricted due to privacy protections [58, 59]. It is also employed for analyzing disease risk factors [60] and medical imaging [61]. In the industrial sciences, synthetic data finds applications in various areas, such as corrosion prevention in metal pipes, simulation of electrical load data, and generation of synthetic rainwater data for flood prediction [62, 63, 64]. Synthetic data generation has evolved from the partial generation of data columns [65] to incorporating data descriptions. Techniques, including Gaussian Copula Models extended with condition parameter aggregation, have been employed [54] successfully. Other methods involve estimating frequencies to preserve unconditioned probability distribution or constructing a Bayesian network to model dependencies between attributes [66] and hierarchical autoregressive models [67]. In contrast to the aforementioned data modalities, GAN-based data generation has been relatively less successful for data that has a temporal component such as natural language data [68] and discrete sequence data [69]. Yu et al. [70] proposed SeqGAN whose generative model is composed of RNNs with LSTM cells, and it's discriminator is a convolutional neural network. The generator is updated by a policy gradient and Monte Carlo (MC) search on the expected reward from the discriminator. Additionally, Quant GAN, a financial time series forecaster uses discrete values to predict long range dependencies by employing temporal convolutional networks (TCNs) for it's generator and discriminator [71]. SeqGAN and Quant GAN demonstrated success in generating sequential time-series discrete values, however, it is difficult to evaluate the performance of these time series models without a unified metric [72]. Other GAN frameworks have been utilized for the generation of geo-privacy-protected trajectory data [73], and have been extended by incorporating loss metric functions and Long Short-Term Memory (LSTM) cells [74]. In government sectors, Generative Adversarial Imitation Learning (GAIL) has been applied for traffic volume forecasting. This involves integrating Recurrent Neural Network (RNN) embedding layers within the generator framework [75]. GAN-based synthetic data models require non-trivial quantities of training data and are therefore challenging to use in sparse data environments.

2.3 Reinforcement Learning

Reinforcement learning is a machine learning framework wherein an agent acquires decision-making capabilities through interactions with an environment. The agent is provided feedback in the form of rewards or punishments based on its actions within the environment. The primary objective in reinforcement learning is for the agent to develop a strategy or policy that maximizes cumulative rewards over time. This is accomplished through an iterative process of exploration and

exploitation, enabling the agent to take actions leading to favorable outcomes while adapting to the dynamic nature of the environment [76].

A Deep Q Network (DQN), as introduced by Mnih et al. [77], is a reinforcement learning algorithm rooted in the principles of Q-learning. The core concept of DQN revolves around a deep neural network that takes the current state as input and yields an estimate of the expected cumulative reward that an agent can attain for each possible action available at that state. DQN's training process is marked by its utilization of an experience replay buffer, a memory storage system. In this buffer, state-action-state pairings are collected and saved. The network is updated through a twofold process. First, random batches of experiences are drawn from this replay buffer, providing a diverse set of data for training. Second, the network's parameters are adjusted by computing the disparity between the predicted Q-values and the target Q-values. This process ensures that the neural network learns to approximate the optimal action-value function, Q, by minimizing the discrepancy between its predictions and the desired Q-values, which represent the ideal actions to be taken in different states. Through this reinforcement learning mechanism, DQN enables agents to make informed decisions in complex environments.

2.4 Reward Shaping

Reward shaping entails the process of augmenting the reward associated with a state-action pair by introducing a value derived from the relationship between the current state-action and future state-action pairs [78]. In one of the early studies on reward shaping [39], the authors introduced this value as the difference in a potential function between the next and current states, which they approximated using the negative Manhattan distance. Subsequent researchers have expanded upon this concept by proposing variations of the potential function that take into account both future and past states [79], distances to the nearest states observed in expert demonstrations [80], and even employing a bi-level optimization approach to concurrently optimize potential function values and policy parameters, such as the weights of a policy network, in the learning process [81]. The objective of the aforementioned reward shaping methods is to expedite policy convergence towards a goal. In contrast, our primary aim is to train a policy that rewards agents by elevating the reward values for state-action pairs observed in human demonstrator data. While traditional reward shaping can lead to faster convergence to an optimal policy, our method can potentially transform what may initially be a sub-optimal policy into an optimal one.

2.5 Imitation Learning

Learning from demonstration (LfD) is a concept where a learning process acquires knowledge or new skills by observing and imitating actions or behaviors demonstrated by an expert. Proposed by Schaal [82] in 1997, LfD serves as a method to prime the policy network of a reinforcement learning model using demonstrations for accelerated learning. In this approach, expert demonstrations are transformed into state-action pairings and fed through the reinforcement learning agent, subsequently creating a policy that the agent can optimize through continued exploration and exploitation.

Dataset Aggregation (`DAgger`) [40] is a reinforcement learning algorithm that is used to increase the performance of an existing policy. It begins by collecting a dataset of trajectories through interactions with the expert policy and subsequently employs this dataset to train a new policy. This policy is then sampled to generate a new trajectory. Actions for each state from this trajectory are obtained by querying the expert policy, and the resulting state-action pairings are added to the dataset. The dataset is utilized to train a supervised learning model, enabling it to predict the actions the expert is likely to take for a given state. The ensuing policy obtained through supervised learning is applied in the subsequent iteration, and this cycle repeats until a specified iteration count is reached. Upon completion the algorithm returns an optimal policy that has minimized loss.

2.6 Cognitive Bias

Systematic deviations from established rules are known as cognitive biases. Individuals, are anticipated to adhere to rational maxims, uphold consistent preferences, and aim to maximize benefits in decision-making. Cognitive biases manifest as actions that diverge systematically from established rules, disrupting the expectation that individuals consistently follow rational maxims. These biases unveil intrinsic variations in human decision-making processes, illustrating decisions that defy the principles of rationality [83, 84, 85].

One of these biases, the anchoring effect, manifests when individuals overly depend on the initial information or ”anchor” presented to them during subsequent judgments or estimations [86]. In a study conducted by Valdez et al. [87], the impact of priming and anchoring on perceptual tasks in data visualization was investigated. Participants assessed the visual separability of classes in scatter plots, with one group primed with examples of overlapping scatter plots (indicating no separation) and the other with clearly separated scatter plots. Results revealed participants were anchored to their priming condition, influencing how they perceived ambiguous scatter plots based on whether they were initially primed with separated or overlapping examples. Conversely, in a

study by Cho et al. [88], the anchoring effect was explored in the context of decision-making using real-time tweet analysis through data visualization software. Participants were primed with numerical and visual anchors, with an emphasis on specific features related to their anchor during training. The findings indicated that participants devoted significantly more time to views aligned with their priming condition, suggesting the anchoring effect's extension to visual cues in data analysis software, impacting the decision-making process. These studies collectively underscore the anchoring effect's influence on decision-making processes across diverse contexts, revealing the interplay between priming, anchoring, and subsequent judgments in various cognitive tasks.

3 Method

Our objective is to reproduce human decision-making in gaming environments using deep learning methods with limited training data. This objective is pursued through two experiments. The initial experiment serves as a pilot study, assessing the algorithm's effectiveness. The second experiment expands on the first, duplicating its design and introducing the anchoring effect bias in recruited participants. This extension leverages insights gained from the preliminary experiment. Both experiments involve distinctive game environments organized in a grid-world framework. A small group of human participants navigate and solve these games, resulting in sparse human demonstrator data. Expert demonstrations from these participants guide a reinforcement learning agent employed in an imitation learning algorithm. This process enables the agents to mimic the gameplay behaviors exhibited by human participants, facilitating the generation of synthetic trajectories.

3.1 Mathematical Framework

The algorithms were embedded within the framework of a Markov Decision Process, denoted as $\mathcal{M} = (S, A, T, R, \gamma)$. In this context, S represented the set of states within the grid world environments, A denotes the set of permissible actions for both humans and agents, $T : S \times A \times S \to [0,1]$ defined the transition model governing the dynamics of the environment, $R : S \times A \to \Re$ represented the reward function, and $\gamma \in [0,1]$ served as a discount factor. A trajectory was conceptualized as a sequence of state-action pairings, with the i-th trajectory expressed as $\tau_i = (s_{i,k}, a_{i,k})_{k=0}^{|\tau_i|}$ at time step k.

The collection of human demonstrator trajectories was denoted as $\mathcal{H} = \tau^h$, where each $\tau_j^h = (s_j^h, a_j^h)$.

The human-demonstrated states and actions were represented as $(S^h, A^h) : S^h \subseteq S, A^h \subseteq A$.

The algorithm comprised two main phases. In the first phase, we trained an agent through reinforcement learning and reward shaping, enabling it to produce trajectories closely mirroring those demonstrated by humans. The subsequent phase involved utilizing imitation learning to generate synthetic trajectories. This was achieved by incorporating the reinforcement learning agent, trained with reward shaping, as the expert demonstrator within the algorithm.

3.2 Training Expert Agent via Reward Shaping

Given the sparse nature of our data, we acknowledge the need for a significant amount of data to effectively train the imitation learning algorithm. To address this, we choose to train a RL agent to act as a substitute for this training data, offering the necessary expert demonstrations. The rationale behind training the RL agent with reward shaping is to generate trajectories that closely emulate those of the human demonstrator. Consequently, the trained agent can then serve as the provider of expert demonstrations in the imitation learning algorithm.

Policies acquired by RL agents are generally optimal, yet the state-action pairings within human demonstrator trajectories may occasionally be suboptimal, reflecting the imperfect nature of human decision-making. To address this challenge, we incorporated reward shaping [39]. This technique entails enhancing the rewards associated with state-action pairs in expert agent trajectories by an additional value F. This augmentation is devised to encourage convergence towards state-action pairs observed in the trajectories of human demonstrators, as delineated by Equation 1.

$$F(s,a,s') = \begin{cases} \delta(s), & \text{if } \exists (s^h,a^h) \in (S^h,A^h) : (s,a) = (s^h,a^h) \\ 0, & \text{if } \exists (s^h,a^h) \in (S^h,A^h) : s = s^h, a \neq a^h \\ \gamma\phi(s') - \phi(s), & \text{otherwise} \end{cases} \tag{1}$$

s^G was the target or goal state of the environment, and D_{min} was the Dijkstra's shortest path, and D_{max} was the Dijkstra's shortest path to s^G:

$$\delta(s) = 1 - \frac{D_{min}(s,s^G)}{D_{max}(s,s^G)} \tag{2}$$

$$\phi(s) = 1 - \frac{D_{min}(s,s^h)}{D_{max}(s,s^h)}, s^h \in S^h \tag{3}$$

The function $\phi(s)$ represents the potential indicating the suitability of state s toward the goal, and γ denotes the discount factor. The reward shaping mechanism operates as follows: If the state-action pair (s,a) executed by the agent corresponds to a state-action pair (s^h,a^h) in the human-demonstrated trajectory, the value of F is computed based on the normalized shortest path from the current state s to the goal state s^G. Introducing additional reward at state s provides a higher incentive to replicate the action taken by the human demonstrator. Conversely, if only the current state of the agent matches a state from the human-demonstrated trajectories but not the action, the F value is set to 0, indicating that the reward remains unchanged. Lastly, if the agent's state s is not present in the set of human-demonstrated trajectories $\mathcal{H}$, the F value is determined by the poten-

tial function $\gamma\phi(s') - \phi(s)$, following a similar approach to [39]. Here, the potential function $\phi(s)$ represents the normalized shortest path distance from the agent's current state s to the nearest state s^h in the set of human-demonstrated trajectories, and $s' = \arg\max_{\hat{s}} T(s,a,\hat{s})$ denotes the next state returned by the transition function when taking action a at state s. The updated reward encourages the agent towards the closest state of the human-demonstrator trajectory.

Algorithm 1 outlines the process of training the expert agent using reward shaping based on human trajectories. In this procedure, the agent iteratively generates trajectories through its current policy, applies the reward shaping function defined in Equation 1 to the states within the generated trajectories, and subsequently updates its policy using the modified rewards. In each training iteration, denoted as k, the agent enhances its nominal reward $R(s,a)$ by incorporating an additional term $F(s,s')$, where s' is determined as $s' = \arg\max_{\hat{s}} T(s,a,\hat{s})$. The agent achieves expert status when it consistently produces trajectories with an average length exceeding N_{thresh} over a specific window size W.

Algorithm 1 Training expert agent from human-generated trajectories.

input : S, A, T, R, γ: MDP underlying environment
N_{thresh}: Mean episode length threshold
T_{demo}: Set of human expert trajectories
output: π_{RL}: expert agent policy

Procedure `Train-Expert-Agent` $(S, A, T, R, \gamma, N_{thresh}, RL_{algo}, T_{demo})$
  $k \leftarrow 0$
  while $(\frac{\sum_{k-W}^{k} len(\tau_k)}{W} < N_{thresh})$ **do**
    $\tau_k \leftarrow$ Trajectory generated using current policy π_k^{RL}
    for *each* $(s,a) \in \tau$ **do**
      $R(s,a) \leftarrow R(s,a) + F(s,a,s')$, where $F(s,a,s')$ is given by Equation 1
    end
    $\pi_{k+1}^{RL} \leftarrow$ update policy π_k^{RL} with shaped rewards
    $k \leftarrow k+1$
  end
  return π_{RL}

3.3 Synthetic Trajectory Generation Using Imitation Learning

After establishing the expert agent, we leverage its capabilities to produce synthetic trajectories through an imitation learning methodology, as outlined in Algorithm 2 based on the Dataset Aggregation (DAgger) algorithm [40]. The fundamental idea behind DAgger [40] is to expand the expertise of the expert policy π^{RL} to encompass states it may not have encountered previously. To achieve this, the algorithm combines the expert policy with a policy generated by training a

classifier on states visited by the expert policy, utilizing the mixing parameter β_i (line 6). Trajectories generated by this amalgamated policy have an increased likelihood of exploring states that were previously unvisited by the expert policy (line 8). To mitigate substantial variations arising from actions of the expert policy, the actions in states visited by the mixed policy trajectory are constrained to align with the expert policy (lines 9-13) by querying the expert policy for actions corresponding to given states. These refined trajectories are subsequently utilized to train the classifier for mixing in the next iteration (lines 14-15). The optimal policy derived from the classifier is denoted as π^* and is sampled until either the goal is attained or the time horizon is reached, constituting the foundation of T^{syn}. This trajectory is then presented as a synthetically generated trajectory. This iterative process is reiterated to generate the desired number of synthetic trajectories.

Algorithm 2 Synthetic trajectory creation from human expert-generated trajectories

input : S, A, T, R, γ: MDP underlying environment
N_{train}: `DAgger` training length
π^{RL}: Expert agent policy

output: T_{syn}: set of synthetically generated trajectories

Procedure `Generate-Synthetic-Traj`(S, A, T, R, γ, N_{train}, πRL)

$\pi^{RL} \leftarrow$ `generate-expert-agent()`
$\mathcal{D} \leftarrow T_{demo}$
$\hat{\pi}_1 \leftarrow \pi^{RL}$
for $i = 1$ **to** N_{train} **do**
  $\pi_i \leftarrow \beta_i \pi_{RL} + (1 - \beta_i)\hat{\pi}_i$
  //Sample a trajectory using π_i
  $\tau^{\pi_i} \leftarrow (s_0, \pi_i(s_0), ..., s_{G-1}, \pi_i(s_{G=1}), s_G)$
  //Get actions from expert policy for states visited in τ^{π_i}
  $\mathcal{D}_i \leftarrow \emptyset$
  for *every* $s_j \in \tau^{\pi_i}$ **do**
    $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup (s_j, \pi^{RL}(s_j))$
  end
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$
end
$\pi^* \leftarrow$ Policy giving highest return in validation
$T^{syn} \leftarrow (s_1, \pi^*(s_1), ..., s_H, \pi^*(s_H))$
return T^{syn}

3.4 Evaluation Metrics

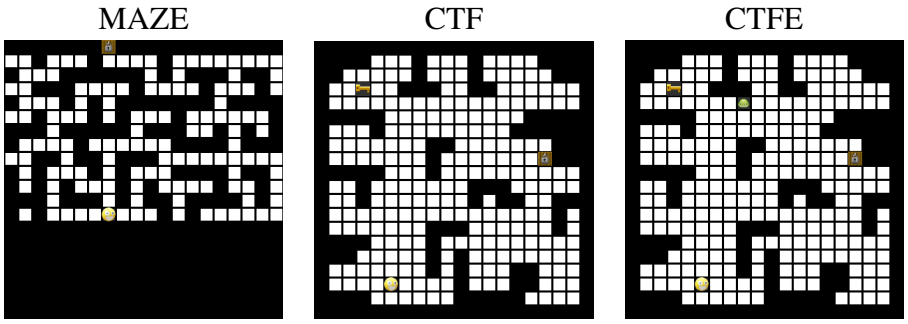
To evaluate the efficacy of our proposed approach, we utilized the METEOR score [89]. Originally developed as a similarity measure for machine-based language translation, the METEOR score

has been repurposed as a distance metric for comparing vehicle trajectories in autonomous driving scenarios [53]. We selected the METEOR score as our primary metric due to its consideration of the order of state-action pairings and their frequency of occurrence, providing a more comprehensive assessment compared to one-to-one comparison metrics. The METEOR score for a sentence translated from a reference language is computed as follows:

$$\text{METEOR score} = \frac{\text{\#mapped words in translation}}{\text{\#words in translation}} \times \frac{\text{\#mapped words in ref.}}{\text{\#words in reference}} \tag{4}$$

A score is determined by multiplying the precision of a translation, which is the total number of mapped words divided by the total number of words in the translation, by the recall, which is the total number of mapped words divided by the total number of words in the reference. This product is then divided by the parameterized harmonic mean of precision and recall.

For our specific problem, a trajectory, whether human-demonstrated or synthetic, is represented as a string by converting each state-action pairing into words of the form "`observation-id, action-id`". The human-demonstrated trajectory string serves as the reference, while the synthetic trajectory string is regarded as the translation. The METEOR score for a synthetic trajectory concerning a human-demonstrated trajectory is computed using Equation 4. METEOR scores fall within the range of [0, 1], with 0 indicating no match and 1 representing a perfect match.



(a) (b) (c)

Figure 1: Maps of the different games used for experiments

4 Pilot Study

The initial experiment's objective was to confirm the effectiveness of our suggested approach for generating synthetic trajectories. This approach is designed to replicate sequential human decision-making during the game play of three uncomplicated maze navigation computer games. Through our experiment, we sought to address the following set of three research inquiries:

1. Can DQN agents be trained using reward shaping from a sparse set of human-generated data to complete simple decision-making tasks at different levels of difficulty?

2. Can synthetic trajectories with low divergence from human-generated trajectories be generated from human generated trajectories using the imitation learning technique in Algorithm 2?

3. What is the effect of the imitation learning in Algorithm 2, used either without and with human data, on the quality of generated trajectories?

4.1 Game Environment

In our experiments, we employed three maze navigation games, each incrementally more complex, to capture human decision-making. These games were used solely as a platform to observe the choices made by individuals while controlling a character within the game. It's important to note that we did not impose any criteria such as rewarding players for selecting more efficient navigation routes during game play. Likewise, the agents employed in our synthetic trajectory generation algorithm did not utilize search or path planning algorithms to optimize the length of their navigation paths while playing the games. Here is a description of the three maps:

4.1.1 Maze Navigation Game

Figure 1(a) provides an overview of the maze game environment (MAZE). The game map comprises a grid consisting of 20 columns and 13 rows of cells. In this depiction, dark squares represent obstacles, while light squares represent empty, navigable cells. The player's character, indicated by the yellow face, is positioned within one of the empty cells near the bottom of the map. The objective of the player is to guide the character to reach the goal location, marked by a brown cell with a lock icon. The character's movement is constrained to four cardinal directions: up, right, down, or left. If there is an unoccupied cell in the direction of movement, the character moves into that cell. Conversely, if an obstacle blocks the path in the chosen direction, the character remains in its current cell. The arrangement of obstacles is such that there is only one viable path leading from the starting point to the goal state, with all other routes leading to dead ends. These routes are all one cell wide. Upon successfully reaching the goal, the player is awarded a score of $+1000$, while all intermediate moves within the game carry a score of 0. The rewards assigned to the learning agent corresponded to the same values as the human scores upon reaching the goal state. For states other than the goal, the agent's rewards were determined as a value proportionate to the state's suitability in terms of progressing toward the key or goal. This suitability was defined using the equation provided below

$$R = \begin{cases} 100, & \text{if agent reaches goal} \\ 1 - \frac{D_{min}(s,s^{goal})}{D_{max}(s,s^{goal})}, & \text{otherwise} \end{cases}$$

4.1.2 Capture-The-Flag (CTF) Game

Figure 1(b) displays the map utilized in our Capture The Flag (CTF) game. This game is set within a grid of dimensions 20 columns by 20 rows, which introduces obstacles to challenge both human and agent movements. However, this environment is not as restrictive as the maze map. Here, paths can be wider than a single cell, allowing for more open areas that facilitate maneuvering across the map. At the outset of the game, the player's character is positioned in the bottom left corner of the map. The initial objective for the player is to guide the character through a maze to reach a key located near the top left corner of the map and collect it. Subsequently, after collecting the key, the player must navigate to the goal, represented by a brown cell with a lock icon, situated near the middle of the map to complete the game. The player receives a score of $+100$ for collecting the key and $+1000$ for reaching the goal with the key in their possession. Similar to the MAZE game, all intermediate moves within the CTF game carry a score of 0. For the learning agent, we established reward values based on the agent reward structure from the MAZE game. However, for non-goal states, the agent's rewards are determined proportionally to the agent's distance from its

current objective (either the key or the goal), in accordance with the equation provided below:

$$R = \begin{cases} 100, & \text{if agent collects key} \\ 1000, & \text{if agent reaches goal with key} \\ 1 - \frac{D_{min}(s,s^{key})}{D_{max}(s,s^{key})}, & \text{if agent does not have key} \\ 1 - \frac{D_{min}(s,s^{goal})}{D_{max}(s,s^{goal})}, & \text{if agent has key} \end{cases}$$

4.1.3 Capture-The-Flag With Enemy (CTFE) Game

In Figure 1(c), we present the CTFE (Capture The Flag with Enemy) game. This game mirrors the CTF game, with the introduction of an additional element: a roaming enemy that moves horizontally back and forth just below the key. If the player comes within a distance of 1 cell from the enemy, it is captured by the enemy, resulting in a game loss with a score of 0. In all other scenarios, the same scoring system for both players and agents as in the CTF game was retained. The agent's reward function for the CTFE game is detailed below:

$$R = \begin{cases} 100, & \text{if agent collects key} \\ 1000, & \text{if agent reaches goal with key} \\ 1 - \frac{D_{min}(s,s^{key})}{D_{max}(s,s^{key})}, & \text{if agent does not have key} \\ 1 - \frac{D_{min}(s,s^{goal})}{D_{max}(s,s^{goal})}, & \text{if agent has key} \\ 0, & \text{if captured by enemy} \end{cases}$$

4.2 Software and Hardware

4.2.1 Software

Game environments for each of the three maps were coded using the AI Gym [90] framework. Each map consisted of a multi discrete value for the observation space. The first value indicated the state number the agent was in, the second value indicated the target the agent was working towards, the key or the exit, and a third boolean value, available on the CTF and CTFe maps only, indicated if the agent was in possession of the key, 1, or not 0. The action space was a discrete value numbered between 0 and 3 that corresponded to movement actions, up, left, right, and down. Stable Baselines 3 [91] was the library used to train the DQN agents and the Imitation library [92] was utilized for implementing the DAgger algorithm. To compute path lengths denoted as D_{min} and D_{max} in Equations 1 and 5, we relied on the all-pair shortest paths calculated through Dijkstra's algorithm, which was executed on a graph representation of the environment. This graph representation was created using the NetworkX [93] library. Lastly, for the METEOR score implementation, we utilized the

DQN hyper-parameter	MAZE	CTF	CTFE
Exploration Fraction	0.8	0.8	0.99
Exploration Initial EPS	0.9	0.9	0.9
Exploration Final EPS	0.1	0.1	0.001
γ (discount factor)	0.999	0.999	0.999
Learning Starts	1×10^5	3×10^5	3×10^5
Learning Rate	1×10^{-4}	1×10^{-4}	1×10^{-4}
Training Time-steps	2×10^6	2×10^6	1.8×10^6
Algo. input or parameter			
No. human demo trajectories	5	5	5
Avg. human traj. length	35	35	34
N_{thresh} (Algo. 1)	55	40	42
W (Algo. 1)	10	10	10

Table 1: DQN algorithm hyper-parameters and Algorithms 1 and 2 parameters in the MAZE, CTF, and CTFE games.

NLTK Toolkit [94].

4.2.2 Hardware

DQN agents were trained on a Google Colab server with 26GB of RAM, four dual-core Intel Xeon CPUs clocked at 2.30 GHz, each featuring a 46MB cache. Additionally, the server was equipped with an NVIDIA Tesla T4 GPU with 16GB of RAM and operated using Chromium 13 as the operating system. In contrast, DAgger algorithms were trained on a laptop with 8GB of RAM, four dual-core AMD Ryzen CPUs running at 2.1 GHz, with a 6MB cache per CPU core. This laptop also included an integrated AMD Radeon Vega-8 GPU and operated on the Microsoft Windows 10 operating system.

4.3 Procedure

One participant completed each of the three maze maps 5 times: MAZE, CTF, and CTFE for a total of 15 trajectories. These human trajectories were collected from the game play and the state and action pairings were extracted. The average human trajectory lengths for each respective game can be found in Table 1. Furthermore, Table 1 lists information about the specific hyperparameters employed in the training of the DQN agent as outlined in Algorithm 1. It's worth noting that while certain hyperparameter settings remained in accordance with the default values set by Stable-

Baselines3, others were subject to fine-tuning aimed at optimizing the DQN agents' performance.

4.4 Results

4.4.1 Can DQN agents be trained using reward shaping

To answer the first research question, we trained a DQN agent on each of the maps: Maze, CTF, and CTFE, using reward shaped values derived from human demonstrator trajectories. In the case of the maze map, the parameter N_{thresh}, as specified in Algorithm 1, was set to a value of 55. This setting is notable, because it significantly exceeded the average of 35 steps taken by humans to successfully complete the map, suggesting the DQN agent could not converge to the average length the human demonstrator could. This behavior was primarily attributed to action oscillations between successive states, which caused the agent to move back and forth between neighboring states or engage in repetitive action sequences within the same state to maximize the reward. Notably, it's important to emphasize that such behavior did not manifest when the trained model was deployed for navigating the maze map during similarity evaluations.

Results of the episode length during training are depicted in Figure 2. All trained models successfully completed the task after training. However, there were variations in the convergence time steps. Maze DQN and CTFE DQN models required 6000 time steps to reach their convergence, while the CTF DQN model reached convergence in 720 time steps. These disparities can be attributed to differences in map structure and task complexity. The maze map presents limited movement options, with a higher likelihood of the agent becoming "stuck" in a corner, leading to navigational challenges. Similarly, the CTFE map introduces added complexity by involving multi-point navigation tasks and the presence of an enemy that can lead to the agent's destruction. This can result in a scenario where an agent successfully reaches and retrieves the key but then gets destroyed by the enemy, effectively getting trapped in a constrained state-action sequence akin to the maze map.

4.4.2 Can low-divergence synthetic trajectories be generated using Imitation Learning

Two variations of the `DAgger` algorithm were created to examine the influence of the human demonstrator trajectories on the imitation learning algorithm. `DAgger-E` did not initially add human demonstrated trajectories to the dataset $\mathcal{D}$ as outlined on line 3 of Algorithm 2. Instead, a trajectory was obtained from the expert DQN agent and used to initialize the dataset: $\tau^{RL} \leftarrow \pi^{RL}$, $\mathcal{D} \leftarrow \tau^{RL}$. The second algorithm `Dagger+E` utilized the 5 human demonstrator trajectories to initialize the dataset $\mathcal{D}$ as outlined on 3 of Algorithm 2.

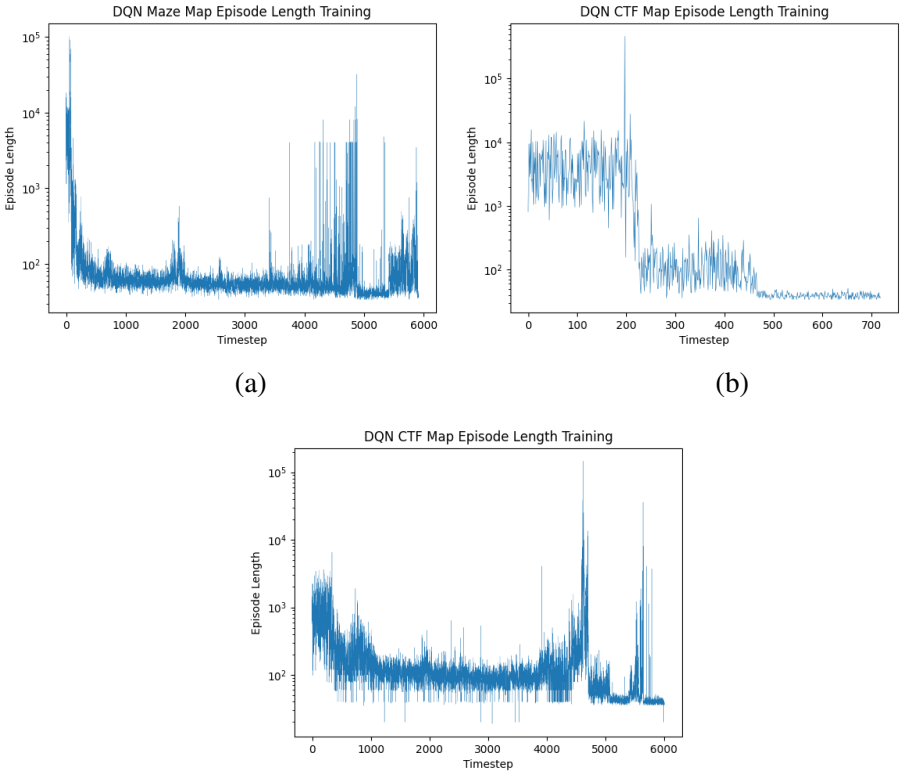


Figure 2: DQN episode length during training for different games.

Algorithm	Expert1	Expert 2	Expert 3	Expert 4	Expert 5
MAZE					
DQN	0.70	**0.97**	.72	0.93	0.89
DAgger-E	0.70	**0.99**	0.72	0.95	0.90
DAgger+E	0.73	**0.97**	0.75	0.93	0.89
CTF					
DQN	**0.75**	0.74	0.62	0.35	0.74
DAgger-E	**0.75**	**0.75**	0.63	0.35	**0.75**
DAgger+E	0.71	0.73	0.63	0.41	**0.75**
CTFE					
DQN	0.17	0.18	0.18	0.18	0.17
DAgger-E	0.69	**0.78**	**0.78**	0.72	0.63
DAgger+E	0.70	**0.78**	0.76	0.72	0.63

Table 2: Average METEOR Scores for each game and agent compared to human experts demonstrators 1-5.

A total of 1000 trajectories were generated from the expert DQN, DAgger-E, and Dagger+E. Meteor scores for each of these trajectories were calculated by comparing them to the human demonstrator trajectories. Figure 3 displays the METEOR scores for each game and algorithm, while Table 2 presents the average METEOR score calculated from 1000 generated trajectories.

In the MAZE game, the three agents exhibited the highest similarity to human demonstrator trajectory 2. DAgger-E achieved the highest average METEOR score of 0.99, while both DQN and DAgger+E scored an average of 0.97. For the CTF game, the results were more diverse, with each algorithm replicating different human trajectories. DQN closely resembled human demonstrator 1 with a score of 0.75, while DAgger+E closely resembled human demonstrator 5, also with a score of 0.75. DAgger-E was an exception, as it exhibited similarity to human demonstrators 1, 2, and 5, each with a score of 0.75. The similarity scores were comparatively lower than those of the MAZE game, likely due to the less constrained layout of the map, which allows for more varied movement compared to the more limited maneuverability in the MAZE game. In the CTFE game, DQN failed to replicate any of the human trajectories, achieving a maximum METEOR score of 0.19. DAgger-E closely resembled human demonstrator trajectories 2 and 3, while DAgger+E exhibited the highest similarity to human demonstrator 2, both achieving scores of 0.78. In summary, these findings support the generation of synthetic trajectories with minimal divergence from human trajectories using the proposed technique in this paper.

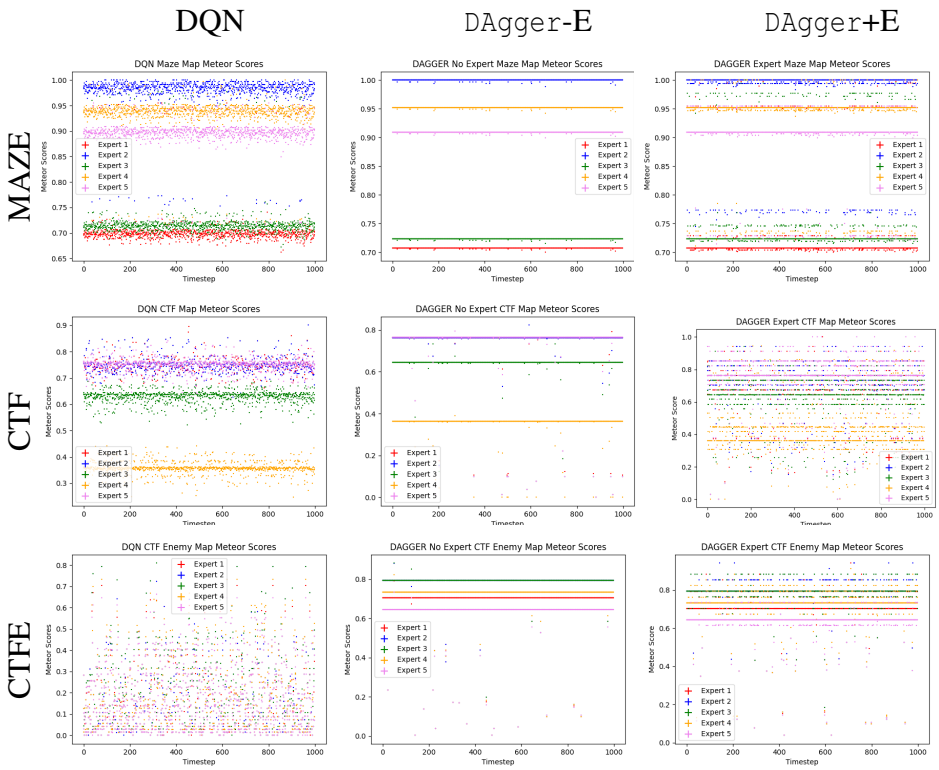


Figure 3: METEOR scores for all three games in our experiments.

4.4.3 What effect does the use of human data have on the quality of trajectories

To assess the performance variations and the DAggger algorithms and to ascertain the impact the integration of human trajectory data during the creation of synthetic trajectories a one-way ANOVA test was conducted between the meteor scores of the DQN expert, DAgger-E, and DAgger+E. These results are summarized in Table 3.

In summary, the statistical analysis revealed significant differences in METEOR scores between DQN, DAgger-E, and DAgger+E across the three maps, with one exception in the CTF map. To gain a deeper understanding of these differences, we examined how often DAgger-E and DAgger+E outperformed DQN. The results, presented in Table 4, indicate that both DAgger-E and DAgger+E frequently achieved METEOR scores above the average DQN score across all human demonstrator trajectories. This suggests that DAgger algorithms can generate synthetic trajectories more similar to human trajectories than DQN.

Furthermore, when comparing the frequency counts between DAgger+E and DAgger-E, it becomes evident that incorporating human expert trajectories within the DAgger algorithm significantly increased the similarity to human demonstrator trajectories. DAgger+E consistently sur-

Game	METEOR 1	METEOR 2	METEOR 3	METEOR 4	METEOR 5
MAZE	102.64	85.18	111.12	66.90	46.30
	$(< .001)$	$(< .001)$	$(< .001)$	$(< .001)$	$(< .001)$
CTF	40.12	5.48	1.76	175.36	2.93
	$(< .001)$	$(< .01)$	(0.17)	$(< .001)$	$(< .05)$
CTFE	7505	8137	7852	7504	6474
	$(< .001)$	$(< .001)$	$(< .001)$	$(< .001)$	$(< .001)$

Table 3: P and Significance Values for One-Way ANOVA

passed `DAgger-E` in terms of METEOR scores, implying that the inclusion of human expert data diversifies the synthetic trajectories, making them more aligned with various human trajectories.

	DAgger-E above DQN Avg.	DAgger+E above DQN Avg.	DAgger+E above DAgger-E
MAZE			
Expert 1	992	903	856
Expert 2	1000	901	696
Expert 3	992	913	901
Expert 4	1000	901	736
Expert 5	1000	901	809
CTF			
Expert 1	973	592	592
Expert 2	973	684	684
Expert 3	976	719	719
Expert 4	972	848	848
Expert 5	972	778	778
CTFE			
Expert 1	987	984	954
Expert 2	988	984	895
Expert 3	988	984	793
Expert 4	987	984	911
Expert 5	987	984	847

Table 4: Frequency of METEOR Scores above METEOR Average

5 Human Bias Experiment

The second experiment aimed to extend the achievements of the initial trial by introducing a more complex game environment and enlarging the participant pool. It specifically targeted the elicitation and replication of human cognitive bias, focusing on the anchoring effect and employing Algorithm 2. The research questions being addressed are as follows:

1. Can the anchoring effect bias be elicited within computer-based gaming environments?
2. Can biased human game play be replicated, if it exists, using Algorithm 2?

5.1 Game Environment

The study's participant conditions were implemented in a game environment using Python's Pygame library [95]. Open-source graphic libraries provided the visuals for the game. At the lower part of the game screen, a bar displayed the total game score, and the remaining ammunition the player possessed. The game maps were positioned above this information bar during the different experimental conditions.

The study comprised several phases outlined below, commencing with the tutorial condition, followed by the demographics condition, the bias conditions, and concluding with the Game Experience Questionnaire. To prevent participants from encountering conditions in the same order, the sequence of bias conditions was randomized using counterbalancing. Throughout all these conditions, various performance metrics, including the onset time of each condition, the timing and types of keystrokes, the participant's location on the map, the current score, the number of enemies defeated, the retrieval of ammunition, the map's duration, and observations of the map's state, were collected by the computer. All data generated during the study were stored in a local log file, and this information was linked solely to a participant's unique ID number to protect their anonymity.

5.1.1 Tutorial

Figure 4 illustrates the the tutorial condition, where participants learned the game controls. They were instructed that their main task throughout the game was to navigate a "drone swarm" across diverse maps in different environments, using only the keyboard's arrow keys. They were guided to move the drone swarm in all directions (left, right, up, and down) to ensure their ability to perform these basic maneuvers. After the navigation test, participants were shown a map exit icon on the screen, and they were informed that reaching this icon would conclude each map after fulfilling the

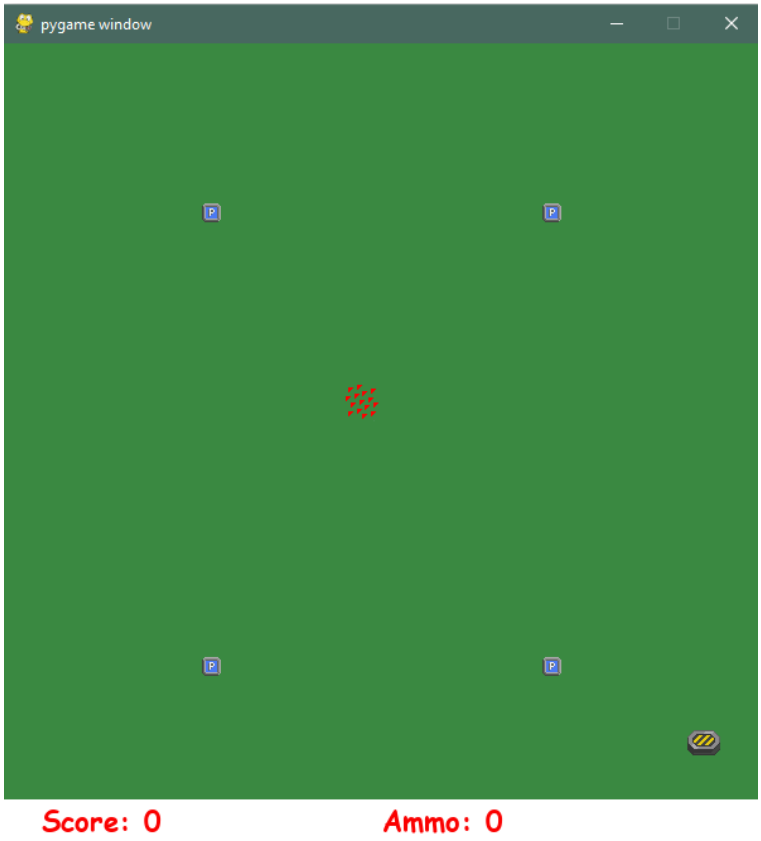


Figure 4: Tutorial Condition Game Screen

required tasks. Subsequently, participants were informed that certain tasks would involve engaging with other elements on the map in combat. They learned that they could initiate attacks on enemy elements by pressing the space bar to fire their weapon. To confirm their understanding, participants engaged with several enemies positioned at varying distances, helping them become familiar with the weapon's range and demonstrating their competence in destroying map elements. Finally, participants were informed that some tasks required them to retrieve items, so they could reload their weapons. They were tasked with a multi-point navigation challenge, involving navigating to various points on the map to obtain weapon refills.

5.1.2 Demographics

Subsequently, the software prompted participants to provide demographic information, encompassing age, gender, college major, and the number of years enrolled in college. Following this, the computer requested participants to assess their self-rated experience level in playing video games, their comprehension of the navigation instructions, and their confidence in completing forthcoming tasks, utilizing a Likert scale ranging from 1 to 10.

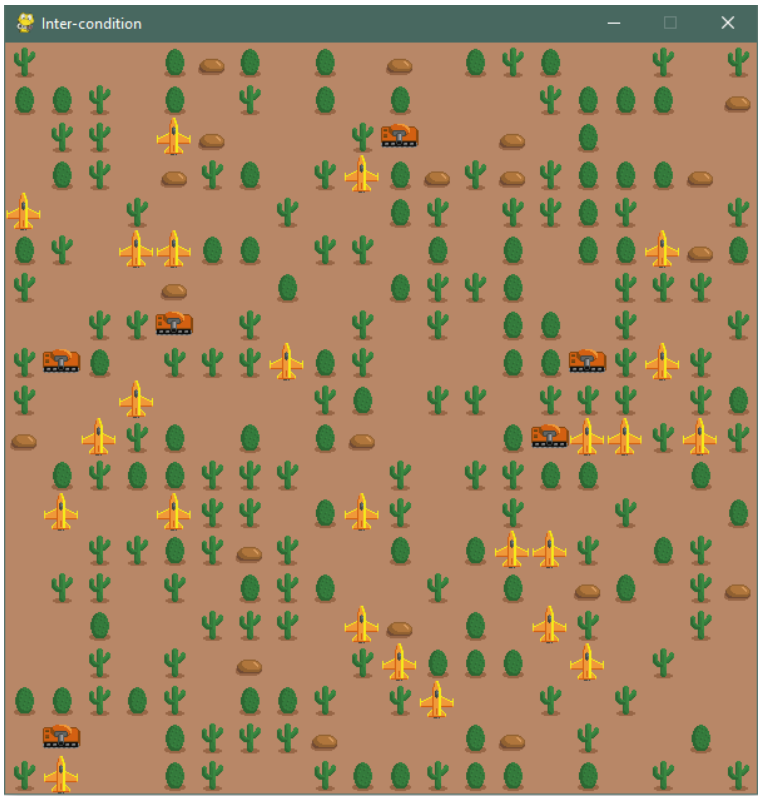


Figure 5: Inter-Bias Condition Game Screen

5.1.3 Inter-bias Masking Condition

Between each condition, there was an inter-bias masking condition aimed at minimizing potential carry-over effects between different conditions as illustrated in Figure 5. In this condition, participants were presented with a desert map containing various elements, such as tanks, drones, vehicles, and buildings. They were instructed to count either the number of tanks or planes, which one was determined by a random number generator. After 15 seconds to complete the counting task, the map vanished, and participants were prompted to enter the total count of either tanks or drones observed on the map. Feedback on the accuracy of their counts was provided, however, they were informed that performance on this task did not contribute to their total score.

5.1.4 Anchoring Bias Condition

The aim of the anchoring bias condition was to explore whether participants demonstrated a bias towards a particular feature on the map, specifically the number of tanks, by anchoring them to a specific map type. Two maps were employed: an urban map depicting a small town with houses, roads, and trees, and a rural map featuring trees, bushes, and rocks in a grassy area. To induce anchoring bias, participants were primed with the number of tanks on each map, earning a point

for each tank destroyed.

The anchoring bias condition commenced with the fulfillment of a baseline (`BLAnchor`) condition, outlined in Figure 6(a). In this condition, the game map was divided into two sections: the top and bottom halves. The top half randomly featured either the rural or urban map, with the selection being counterbalanced across all participants, while the bottom half contained the map not chosen for the top half. For example, if the top half displayed the rural map, the bottom half exhibited the urban map, and vice versa. Each half had an exit for navigation. A wall separated the two halves of the map, compelling the participant to choose an exit through either the rural or urban map. In the baseline condition, the goal was to establish a baseline measurement of the participant's repetitive navigation choices. The task involved navigating to one of the maps and destroying the tanks present. Tanks were not visible at the start of the map and only appeared when the participant moved within the "radar" box, extending 10 cells around the tank. Each half contained an equal number of tanks: 5. Importantly, the participant could only destroy tanks from the chosen map, as navigation to the other half of the map was disabled. This meant that if the participant navigated to the top half of the map, they could not turn around and navigate to the bottom half, and vice versa. Participants completed the baseline condition five times before advancing to the training condition.

In the training condition (`BLTrain`), depicted in Figure 6(b), participants were exposed to either the rural or urban map. The selection was determined through a pseudo-random process with counterbalancing. However, this choice remained constant throughout all training cycles. The primed map contained a larger number of tanks to be destroyed (20) compared to the other map (5), and all tanks were visible. The order of the maps was randomized and counterbalanced. The training task required participants to navigate the chosen map and destroy all visible tanks. After completing this objective, participants could exit the map..

A masking condition (`BLMask`), illustrated in Figure 6(c), followed each map to prevent carry-over effects. The masking condition entailed a simple navigation task where participants navigated to the ammo icons and then proceeded to the exit. During each training cycle, participants completed the rural map, followed by a masking condition, and then the urban map, also followed by a masking condition. Once both maps and their corresponding masking conditions were completed, participants repeated the anchoring baseline condition to assess for bias. This process concluded one training cycle, and participants undertook five training cycles to complete the anchoring bias condition.

Figure 6: Maps of the Anchor Bias Conditions

5.1.5 Game Experience Questionnaire

The Game Experience Questionnaire (GEQ) [96] is a self-report survey designed to evaluate different aspects of a player's digital gaming experience. It consists of questions covering various facets of the gaming experience, such as immersion, enjoyment, challenge, frustration, and more. Participants use a rating scale to provide quantitative data about their gaming experience. The GEQ is commonly used to assess and compare different games, game design elements, or game versions. In this study, we utilized the core component of the Game Experience Questionnaire. The "Competence" item assesses a player's self-assessment of their gaming skill and ability, focusing on their perception of performance and confidence in their gaming capabilities. "Sensory and Imaginative Immersion" gauges a player's sense of full engagement and absorption in the game world. "Flow" measures the player's experience of being in a state of flow during gameplay. "Tension" quantifies the level of tension or stress experienced by the player while playing. "Challenge" evaluates the degree of challenge presented by the game. "Negative Affect" measures the player's experience of negative emotions or feelings during gaming, while "Positive Affect" assesses their experience of positive emotions or feelings during gameplay.

5.2 Software and Hardware

5.2.1 Software

The game the participants played was programmed using the pyGame library [95]. Forms that the participants filled out were designed using pyQT. Reinforcement learning environments were designed using Stable Baselines 3 [91] and NetworkX [93]. Imitation Learning algorithms were coded using the Imitation Learning library [92]. Meteor scores were calculated using the meteor score from the NLTK library [94].

5.2.2 Hardware

DQN agents and Imitation learning algorithms were trained on the same Google Collab servers that were used in Experiment 1. Participants completed experimental conditions on a laptop with 8GB of RAM, four dual-core AMD Ryzen CPUs running at 2.1 GHz, with a 6MB cache per CPU core running the Windows 10 operating system.

5.3 Participants

5.3.1 Demographics

Participants were recruited from the University of Wisconsin-Whitewater via departmental emails and campus flyers. Interested individuals contacted the investigator through email. Those participants that met requirements of being actively enrolled as a full time student at the university and could speak English were allowed to proceed. In total 10 participants completed the study. Their demographic information is outlined in Table 5

Table 5: Participant Information

Participant Number	Age	Gender	Major	Years in College
1	20	Male	CompSci	3
2	19	Male	Finance	1
3	22	Male	Media Arts Game Design	3
4	19	Female	CompSci / Business	1
5	21	Male	CompSci	4
6	20	Female	CompSci	3
7	21	Male	CompSci	3
8	21	Male	Cybersecurity	4
9	21	Male	CompSci	3
10	18	Female	CompSci	1

5.3.2 Confidence Assessment

Following the input of demographic information, participants underwent a confidence assessment. They were required to rate, on a scale of 1 to 10 (with 1 indicating no experience and 10 indicating absolute experience), their proficiency in playing video games, understanding game controls, and confidence in completing the game's tasks. Figure 7 displays the results for each participant. The y-axis represents the Likert scale rating given by the participant, while the x-axis represents the participant number. Navy blue bars indicate ratings for gaming experience, grey bars indicate ratings for understanding game controls, and light blue bars indicate confidence in completing tasks. The average rating for the game experience question was 7.5, indicating that the majority of participants

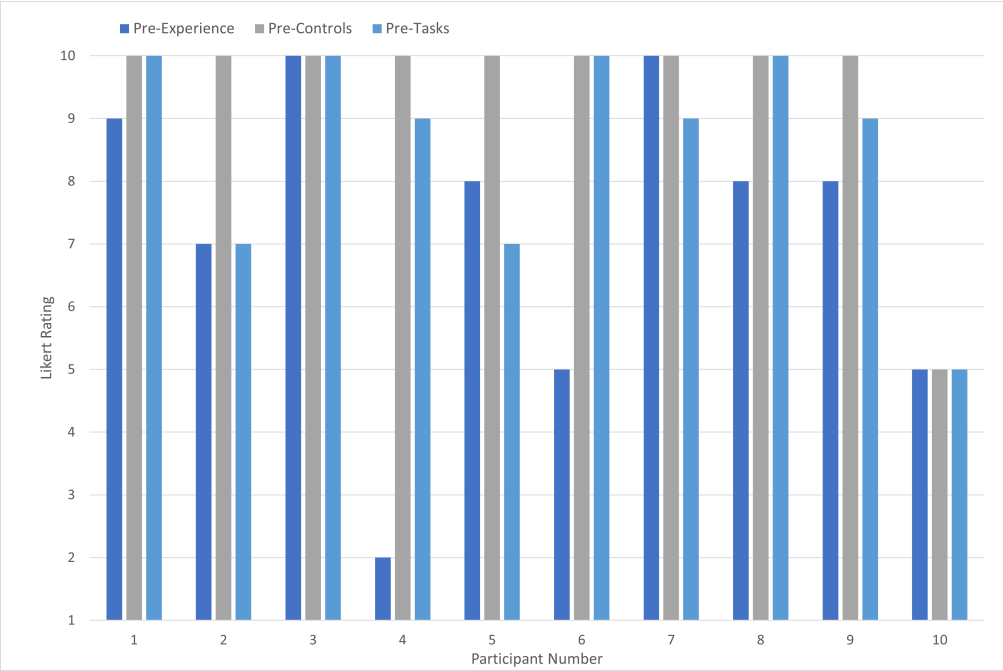


Figure 7: Self ratings of participants

were familiar with playing video games. The average rating for understanding game controls was 9.5, suggesting that the tutorials effectively taught game controls, and participants comprehended how to play the game. The average confidence rating for participants moving forward was 8.6, indicating that participants felt prepared to complete the game after finishing the tutorial condition.

5.3.3 Game Experience Questionnaire

Upon completion of the game conditions, participants assessed their gaming experiences by completing the core component of the Game Experience Questionnaire, as shown in Figure 8. The average competence rating was .61, representing a decrease from the previous rating of 8.6 using the competence questionnaire. This suggests that participants, after interacting with the game, were less confident in their performance during task completions. The average rating for sensory and imagination was .33, indicating that the game did not offer a particularly stimulating experience for the participants. Given that many participants had prior gaming experience, this suggests that the game's graphics and gameplay did not captivate them as much as those found in commercial games. The average flow rating was .52, suggesting that participants were engaged with the game's dynamics but not to a significant extent. The average tension rating was .01, indicating that the game's environment did not induce any degree of stress or tension among the participants. The average challenge rating was .08, implying that the game was not particularly challenging for the participants. This rating can be attributed to the simple nature of the multi-point navigation tasks

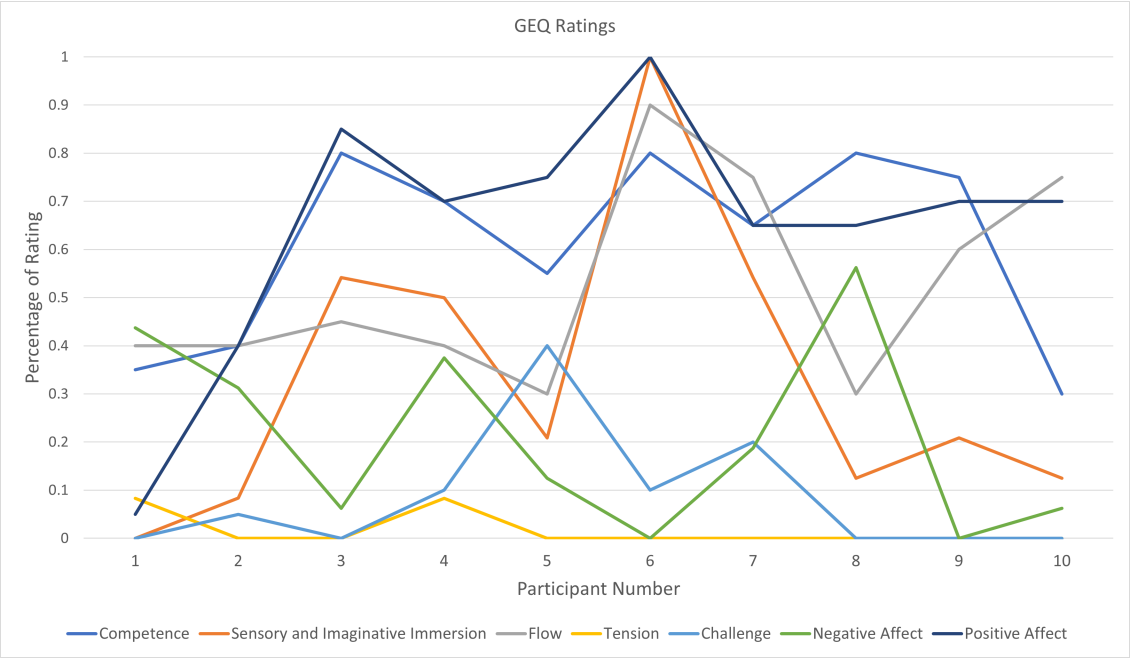


Figure 8: Game Experience Questionnaire Results

within the game and the repetitive exposure to the same conditions. The average negative affect rating was .21, suggesting that the game did not evoke negative emotions among the participants. Conversely, the average positive affect rating was .64, indicating that the game elicited slightly positive emotions in the participants.

5.4 Procedure

The research study took place at the University of Wisconsin-Whitewater in the McGraw building. Upon a participant's arrival, the primary contact reviewed the study details, obtained consent from each participant, addressed any inquiries, and supervised the research study as participants engaged in the game. Each participant received a participant ID number to protect their anonymity throughout the study. All subsequently collected data were linked to this identification number, with the only document connecting participants to their ID being the informed consent form. Participants were informed that they would be playing a game involving various tasks that needed completion. They were assured that these tasks were not challenging, and they could earn points for accomplishing certain objectives. These earned points could later be converted into compensation, up to a maximum of 50, for their participation. The study commenced by seating a participant at a table equipped with a laptop. The investigator entered the participant number into the study's software program, after which the participant progressed through the study conditions. Following the completion of the game, participants were informed that their in-game performance did not affect their

compensation, and they would receive the full participation compensation regardless.

5.5 Results

5.5.1 Can the anchoring effect bias be elicited within computer-based gaming environments?

To address the initial research question, we collected data on map and route choices made by each participant across all the maps within the anchoring bias condition. Initially, we determined whether a participant was anchored to the urban or rural map. Subsequently, we documented the participant's map selections during all the `BLAnchor` conditions. A participant was classified as biased toward the anchoring map if they consistently chose the map they were anchored to more frequently during training conditions compared to baseline conditions. Table 6 provides a summary of the results from the anchoring bias condition.

Table 6: Anchoring Bias Baseline Results U=Urban, R=Rural

Participant Number	1	2	3	4	5	6	7	8	9	10
Anchored To	R	R	U	R	R	U	R	U	R	U
BL1	U	U	U	U	R	U	U	U	R	R
BL2	R	R	U	U	R	U	R	R	U	U
BL3	U	U	U	U	R	U	U	U	R	U
BL4	R	R	U	R	R	U	R	U	U	U
BL5	U	U	U	U	R	U	U	U	R	U
BL6	U	R	U	R	R	R	U	R	R	U
BL7	R	R	U	U	U	U	R	R	U	R
BL8	R	U	U	R	R	U	U	R	R	R
BL9	R	R	U	U	R	U	U	U	R	U
BL10	R	U	U	R	R	R	U	U	U	R

The first row in the table displays the participant numbers, while the row beneath it indicates whether the participants were anchored to the rural (denoted as R) or urban (denoted as U) conditions. Out of the 10 participants, six were anchored to the rural condition, and four to the urban condition. The following five rows represent the outcomes of the initial baseline conditions, which served as a reference for map choices. Participant numbers 1, 2, 7, and 9 displayed an even distribution between urban and rural map choices. Conversely, the remaining participants exhibited a preference, either towards urban or rural maps. It's important to emphasize that these baseline conditions occurred prior to any bias-related training phases. Therefore, any existing bias within these

baseline choices can be attributed to factors like map preferences or misunderstandings. Some participants consistently selected the same map throughout the conditions either due to personal preferences or a misconception that certain map features, like roads, were impassable. The final five rows in the table correspond to the baseline test conditions conducted during the anchoring baseline training phase. Of the four participants, only participant 1 consistently demonstrated a bias toward the map they were anchored to. Participant 2, although initially biased toward the rural condition, did not consistently choose rural maps during the training phase, hence they weren't considered biased. Participant 7, with a bias towards the rural condition, favored urban maps more often during training. Lastly, participant 9, who was initially biased toward the urban condition, chose rural maps more frequently during the training conditions. In total, only one participant, participant 1, showed a clear bias toward the map they were anchored to.

5.5.2 Can a RL agent replicate human bias game play

To answer the second research question, we used the information from participant 1 `BLAnchor` condition to train Algorithms 1 and 2. Participant 1 was the only participant that exhibited anchoring bias during anchoring conditions. To accomplish this we began by extracting the action information that the participant undertook for each map in `AnchorBL` training conditions. This data was subsequently translated into state-action pairs and saved in a file. While conducting this process, it was observed that participants executed numerous actions within the same states. The majority of these actions led off the map or directed them to states that were impassable, like trees or rocks. To enhance the effectiveness of training reinforcement learning agents and replicate the human navigation task accurately, these state-action pairings were eliminated. They did not hold any significance and only hindered convergence during the training process. Additionally, because of an issue with the game where two tanks next to each other could be destroyed with one shot, the state-action data for the fifth anchoring baseline condition was not used. Therefore only information on the first four conditions were used to train the algorithms.

Due to the presence of multiple multi-point objectives within the maps and the human demonstrator's tendency to revisit states and take different actions during each visit, the state-action pairings of the human demonstrator were dissected into sub-paths, each corresponding to the goals of destroying the 5 tanks on each map and reaching the exit:

$$(s_1, a_1) \rightarrow (s_2, a_2) \rightarrow \ldots \rightarrow (s_{n-1}, a_{n-1}) \rightarrow (s_n, a_n) \rightarrow s^g$$

For example, if the human participant targeted the closest tank to its location as its first goal, only states and actions would be rewarded that the human participant exhibited towards that specific

goal. If the agent engaged in state-action pairings that led to a different goal, they were not rewarded, until the first goal was reached. We consider the subset of states that lead to goal number k as S^{g_k} and the actions that lead to goal number k as A^{g_k}. For each condition, a total of 6 sub paths were created, 5 directing them to each of the 5 tanks on the map, and 1 directing them to the exit.

Four DQN agents underwent training using algorithm 1. During training, reward shaping was omitted because we observed that agent convergence was impeded when it was applied, mainly due to oscillations between neighboring states. As a result, the training process employed the following reward function:

$$\mathcal{R} = \begin{cases} 1, & \text{if } \exists (s^h, a^h) \in (S^{g_k}, A^{g_k}) g : (s,a) = (s^{g_k}, a^{g_k}) \\ -.01, & \text{otherwise} \end{cases} \tag{5}$$

If the agent was in state that the human demonstrator was in leading to the current goal, s^{g_k}, and it undertook the same action, then the agent received a reward of 1. For any other condition, the agent received a reward of $-.01$. The introduction of the negative reward value was to encourage the agent to complete the goal quickly. We found that if we did not introduce this value, the agent could converge, however it would not meet the required trajectory length threshold W from 1. Additionally, the agent received a reward of 1 for destroying a tank, and a reward of 1 for reaching the exit.

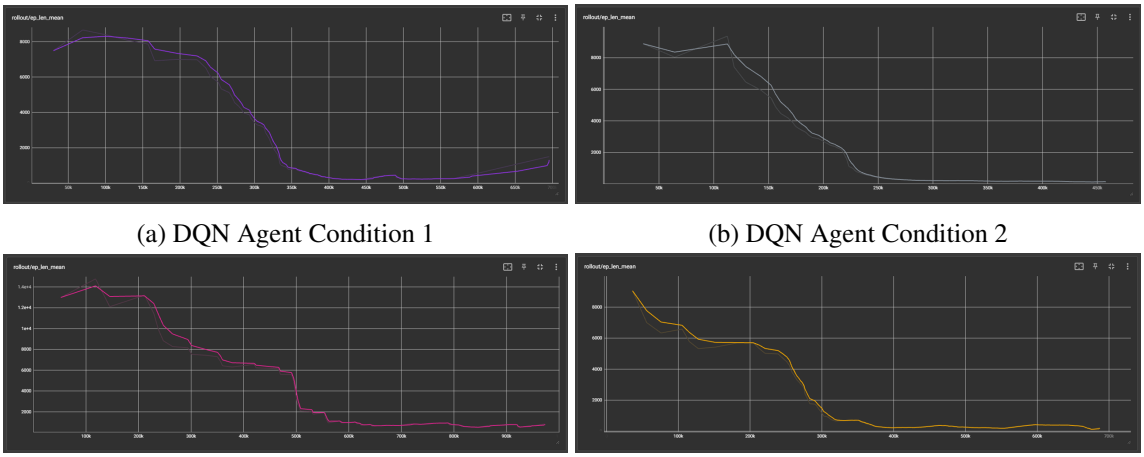
The OpenAI gym environment had an observation space comprising 3 discrete values. The first value represented the agent's current state, the second number indicated the state value of the current objective, and the third value denoted the state value of the exit. The environment's action space consisted of 5 discrete values, where zero represented a rightward movement, 1 indicated an upward movement, 2 signified a leftward movement, 3 represented a downward movement, and 4 denoted the action of firing the weapon.

Table 7 illustrates the hyper-parameters used for training the DQN agents on the human demonstrator data.

In total, all 4 DQN agents were effectively trained, and they all exhibited successful completions of the maps for conditions $1-4$ of the `BLAnchor` conditions. Figure 9 presents a visual representation of the mean episode lengths observed during the training process. The x-axis represents the number of time steps, while the y-axis represents the mean episode length. Specifically, the DQN agent for condition 1 reached the episode length threshold at $690,000$ timesteps, as depicted in Figure 9(a). Similarly, the DQN agent for condition 2 achieved the episode length threshold at $455,000$

DQN hyper-parameter	Condition 1	Condition 2	Condition 3	Condition 4
Episode length Threshold	1000	1000	1000	1000
γ (discount factor)	0.3	0.1	0.1	0.1
Training Time-steps	2×10^6	2×10^6	2×10^6	2×10^6
Algo. input or parameter				
No. human demo trajectories	1	1	1	1
Avg. human traj. length	164	122	137	117
N_{thresh} (Algo. 1)	220	220	220	220
W (Algo. 1)	5	5	5	5

Table 7: DQN algorithm hyper-parameters and Algorithms 1 and 2 parameters in the `BLAnchor` conditions



(a) DQN Agent Condition 1

(b) DQN Agent Condition 2

(c) DQN Agent Condition 3

(d) DQN Agent Condition 4

Figure 9: Episode Length Data for DQN Agents during `BLAnchor` conditions

timesteps, as illustrated in Figure 9(b). For condition 3, the DQN agent reached the episode length threshold after 975,000 timesteps, as shown in Figure 9(c). Finally, the DQN agent for condition 4 accomplished the episode length threshold at 690,000 timesteps, as indicated in Figure 9(d).
After training DQN agents with human demonstrator data, the agents were employed as experts within the `DAgger-E` and `DAgger+E` algorithms. Subsequently, 1000 trajectories were produced using the DQN agent, `DAgger-E`, and `DAgger+E` for the `BLAnchor` conditions from 1 to 4. Meteor scores were computed based on these trajectories to assess their resemblance to human demonstrator trajectories. The results are presented in Figure 10.

In Figure 10, the y-axis represents the METEOR score ranging from 0 to 1. The green data corre-

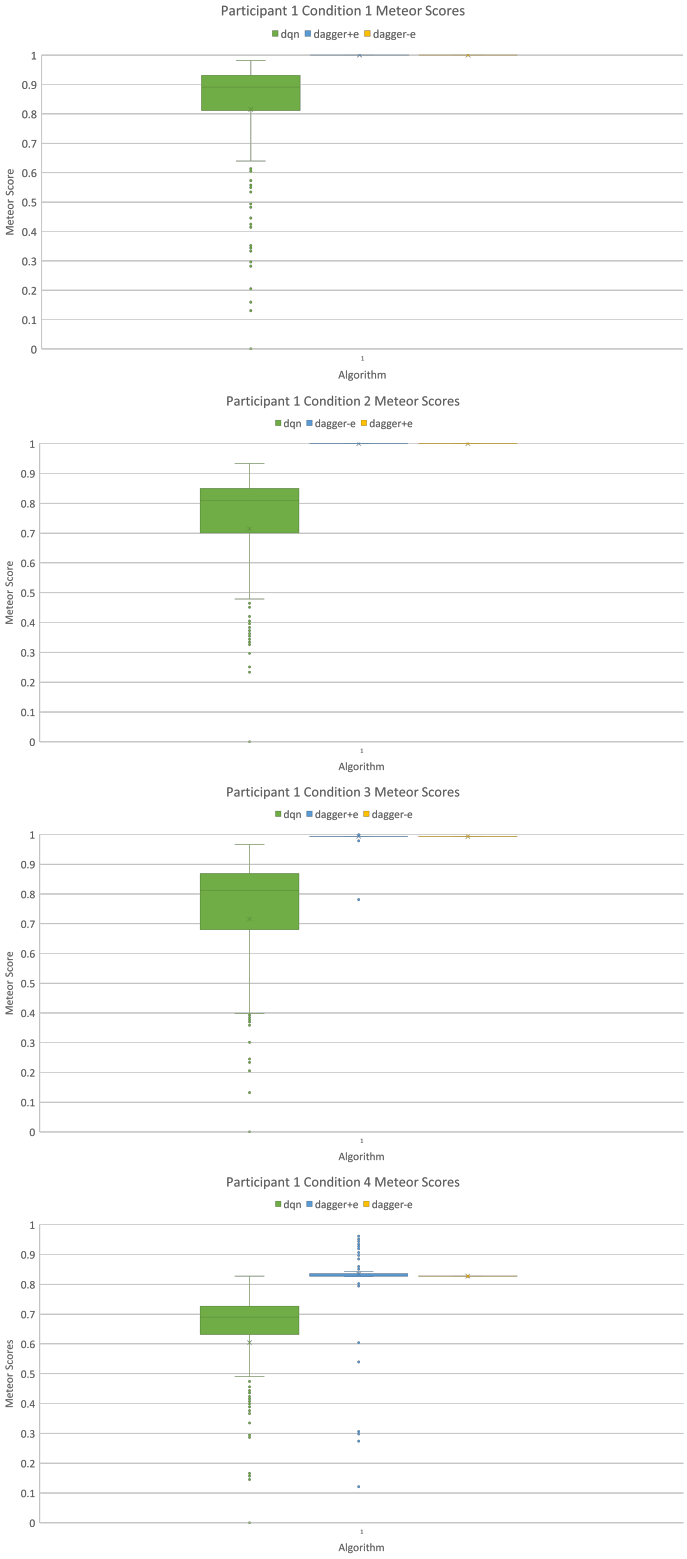


Figure 10: Meteor scores for the `BLAnchor` conditions

sponds to the METEOR score of the DQN agent, the blue data represents the DAgger+E algorithm, and the orange data represents the DAgger-E algorithm. Across all four conditions, both DAgger+E and DAgger-E generated trajectories more similar to human trajectories compared to the DQN agent, as indicated by their higher METEOR scores.

In conditions 1, 2, and 3, the DAgger algorithms achieved METEOR scores of 1, signifying perfect replication of human trajectories. For condition 1, the DQN had an average METEOR score of 0.815 with a standard deviation of 0.22 and a score range from 0 to 0.98. In condition 2, the DQN agent achieved an average METEOR score of 0.7 with a standard deviation of 0.24 and a range from 0 to 0.93. Condition 3 displayed the DQN agent with an average METEOR score of 0.71, a standard deviation of 0.23, and a score range between 0 and 0.96. This indicates the DQN agent's ability to generate versatile trajectories, some of which closely resemble human demonstrator data.

In condition 4, the DQN agents averaged a METEOR score of 0.60, with a standard deviation of 0.22, reflecting reasonable but somewhat variable performance in this condition, ranging from 0 to 0.82. Notably, the DAgger-E algorithm excelled, achieving an average score of 0.82 with a low standard deviation of 2.53×10^{-15} and a score range of 0.826 to 0.827. DAgger+E also delivered strong results, averaging a score of 0.83, with a standard deviation of 0.05 and a score range of 0.12 to 0.96. These findings indicate that DAgger-E and DAgger+E algorithms outperformed the DQN agents in condition 4, with DAgger-E demonstrating exceptional consistency and DAgger+E a strong overall performance, albeit with some variability.

To assess performance disparities among DQN, DAgger-E, and DAgger+E, we performed an analysis of variance (ANOVA) on their meteor scores across all four conditions, as specified in Table8. Our analysis revealed substantial performance distinctions between the DAgger algorithms and the DQN agents. This indicates that the DAgger algorithms significantly surpassed the DQN agents in emulating human bias during game play. They achieved this by generating synthetic trajectories that closely resembled the human player's actions more than the DQN agent.

	Condition 1	Condition 2	Condition 3	Condition 4
F statistic	649.41	1383.61	1343	985.69
P Value	4.93×10^{-235}	0.0	0.0	0.0

Table 8: ANOVA Conducted on the Meteor Scores Across all 4 conditions

6 Conclusion

6.1 Conclusions

This research encompassed two experiments aimed at examining efficacy of our proposed algorithm in synthesizing trajectories that emulate the sequential decision-making processes of human players during game play. The central focus of the study revolved around addressing five research inquiries: Can DQN agents be effectively trained through reward shaping? Can imitation learning yield synthetic trajectories characterized by minimal divergence? What impact does the infusion of human-generated data have on the quality of synthetic trajectories? Can the anchoring effect cognitive bias be induced within computer-based gaming environments? Lastly, can a reinforcement learning agent faithfully replicate biased human game play?

The successful training of DQN agents was achieved through the utilization of reward shaping derived from human demonstrator data, in experiment 1, and without reward shaping in experiment 2. Notably, all DQN agents achieved task completion, exhibiting trajectory lengths that surpassed those of human demonstrators. While this suggests the ability of DQN agents to replicate task completion utilizing human demonstrator trajectories, it was observed that they executed a greater number of actions compared to human participants.

Subsequently, these trained DQN agents were integrated as expert demonstrators within an imitation learning framework, facilitating the generation of trajectories with low divergence from human demonstrator trajectories. Evaluation through Meteor scores confirmed the successful creation of synthetic trajectories exhibiting a high degree of similarity to human demonstrator trajectories, thereby affirming the algorithm's capability to generate synthetic trajectories even with sparse data.

Of particular significance was the consistent augmentation of similarity to human demonstrator trajectories within the DAgger+E framework, wherein human expert data played a pivotal role. This highlights the algorithm's capacity to consistently enhance the fidelity of synthetic trajectories through the inclusion of human expert data. The algorithm was then generalized to a larger map entailing multiple objectives and a larger state space to test performance. Human participants were recruited to complete a study protocol devised of a multiple tasks whose function was to elicit cognitive biases. Out of the 10 participants who completed the study, 1 exhibited a high degree of cognitive bias, suggesting that computer game based environments can evoke the anchoring effect cognitive bias, albeit in limited capacity. This participant's game play information was utilized to train a DQN agent across 4 map conditions, using state-action information in the reward function effectively replacing the utilization of reward shaping. DQN agents trained across these conditions

completed the same game maps as the human demonstrator using a higher frequency of actions when compared to human demonstrator data, effectively replicating DQN training from the first experiment. These DQN agents were employed as expert demonstrators within the `DAgger` algorithm and generated trajectories that were a perfect replication of human demonstrator trajectories, a marked improvement over the quality of synthetic trajectories generated during the first experiment.

The divergence in algorithmic performance between the two experiments can be attributed to several factors. Primarily, the initial experiment incorporated reward shaping, whereas the subsequent one did not. In scenarios with an expanded state space, fewer movement constraints, and an increased number of multi-point objectives, the creation of states with multiple visitations resulted in diverse actions. This complexity hindered the convergence of DQN agents, as each state, regardless of its presence in human demonstrator data, bore a high reward due to the gradients introduced by the reward shaping function (F value). This phenomenon introduced numerous local minima, impeding convergence, as participants extensively explored the majority of the map, leaving minimal room for unvisited states. Despite being infrequently visited, these states retained high reward values due to their proximity to visited states. As a consequence, reward maximization did not align with map completion but rather manifested as oscillations around states within these local minima. This challenge prompted the adoption of a reward function based on sub-paths leading to each objective in the second experiment. However, even with this modified reward function, some agents failed to complete maps, encountering local minima due to states involving multiple actions. Agents tended to maximize rewards by converging on a single state and repetitively invoking $2-3$ actions yielding high rewards. To address this issue, human demonstrator data underwent "pruning," wherein states with multiple actions were scrutinized, and actions leading to untraversable or off-the-map states were removed, as they did not impact the participant's "intention." With these adjustments, agents successfully achieved convergence. The significance of an efficient reward function becomes evident when considering the higher quality of synthetic trajectories produced by DQN agents in the second experiment, as indicated by the Meteor scores in Figure 10 compared to those in Figure 3 from the first experiment. Additionally, the use of reward shaping appeared more warranted in the first experiment which encompassed long horizon rewards with sparse intermediate rewards for movements, as opposed to the second experiment which entailed a reward structure that rewarded the same for movement and goal completion. This suggests that reward shaping could have a more prominent role in shaping RL agents when sparse rewards are available.

6.2 Limitations

A principal constraint in this study pertains to the limited manifestation of bias during the game play conditions of the second experiment. Among the ten recruited participants, only one exhibited a discernible bias towards the map type they were anchored to, and, notably, out of the four conditions displaying such bias, merely four proved usable. Several factors contributed to this outcome. Firstly, a significant portion of participants exhibited a lack of engagement with the game, as evident in their self-reported values on the game questionnaire. Participants perceived the map levels as non-challenging, lacked immersion in the gaming experience, and reported low flow ratings. This collectively suggests a general lack of investment in the game, despite relatively higher scores of positive affect. The repetitive nature of training and testing tasks, coupled with the less engaging graphics of the game, likely contributed to this disposition. Notably, the majority of participants were computer science majors with substantial experience in video games, and it is plausible that a more graphically captivating environment could have induced a higher degree of bias. Larger incentives for completing the game or a greater clarification on the exchange of points for compensation could motivate future researchers to complete game requirements, resulting in more bias elicitation. Participants did demonstrate a bias towards other map features without being conditioned on them such as roads or map preference. Future research should include neutral map features to separate conditions. This could include using novel interference features that could serve as alternative interference mechanisms on maps instead of the utilization of houses, rocks, trees. Furthermore, maps could be distinguished not by terrain features and instead by color features. For example, instead of having maps during anchoring conditions of rural and urban, maps could be distinguished by neutral colors such as red or blue. Alternatively, preference assessments could be conducted before map conditions ensuring preliminary biases are not present to confound experimental conditions.

Secondly, the experimental design involved a degree of deception to motivate participants, informing them that points earned during game play could be exchanged for compensation of up to 50 dollars. This strategy aimed to incentivize efficient task completion and encourage a more serious approach to the game. However, a stronger emphasis on the correlation between points earned and compensation could potentially have heightened participants' motivation to fully engage.

Lastly, participants entered certain anchoring conditions with pre-existing biases towards specific features on the map. Attempts to mitigate these biases through inter-bias masking in prior conditions proved ineffective. Consequently, carry-over effects from earlier stages of the study, confusion regarding instructions, or graphic selections may have contributed to these biases. For instance, one participant reported avoiding roads, assuming they were impassable, leading to a preference for ru-

ral maps. However, a review of bias condition ordering did not show any patterns that could have contributed to this.

This study is centered around the evaluation of the proposed algorithm's effectiveness in generating trajectories within the confined scope of computer-based gaming environments. The principal aim is to replicate human decision-making processes within the context of gaming. However, it is essential to acknowledge the existence of alternative modalities, such as the utilization of data visualization methods [30], which also serve as means to elicit and study human decision-making patterns. While our primary focus was on emulating discrete action information in the form of trajectories within a gaming framework, the algorithm's potential applicability to other contexts remains uncertain. Unlike certain methodologies that are explicitly versatile across various domains, the algorithm's generalization beyond gaming scenarios is challenging to hypothesize. This raises questions about its adaptability and efficacy in capturing diverse decision-making dynamics across different applications. The study revealed the algorithm's success in generating trajectories with minimal divergence during the second experiment. However, the observed variability in algorithmic performance between experiments highlights the sensitivity of the approach to specific factors, particularly those associated with reward shaping. This sensitivity suggests that the algorithm's effectiveness may be influenced by contextual nuances and specific algorithmic parameters.

6.3 Future Work

In future research, consideration should involve broadening the participant pool to encompass a more diverse demographic. The goal is to include individuals with varied backgrounds, experiences, and distinct gaming preferences, aiming to amplify the applicability and broader relevance of the study's findings. This expansion can be achieved by intentionally recruiting participants from different academic disciplines or those with varying degrees of familiarity with video games. It's worth noting that the current study grappled with limitations due to its reliance on a relatively small participant pool, predominantly comprising individuals already acquainted with computer games and their associated preferences. This limited pool may not fully capture the spectrum of responses and biases that could emerge in a more heterogeneous participant group.

To address this limitation and ensure a more comprehensive understanding of bias elicitation, future research should actively seek participants with diverse backgrounds. In particular, including individuals beyond the university system can provide valuable insights into how biases manifest in a broader context, transcending the specific demographics prevalent within academic settings. Expanding the participant pool in this manner serves not only to bolster the external validity of the

findings but also facilitates a more nuanced exploration of biases across various social, cultural, and experiential contexts. Such an approach aligns with the imperative to ensure that research outcomes reflect the diversity inherent in real-world populations, ultimately fortifying the robustness and relevance of the study's conclusions.

In future research methodologies, it is imperative to explore alternative strategies for inducing cognitive bias in participants. This necessity becomes apparent when considering the observed limitations in the current study, where the employed bias conditioning method proved ineffective in eliciting bias from all participants. This highlights a potential shortfall in the efficacy of the current approach, prompting a more nuanced exploration of alternative techniques. While the method for bias elucidation in this study drew inspiration from established practices in previous research, its limitations indicate room for improvement. In addressing this, future studies might consider exploring different anchoring bias techniques or integrating elements from applied behavioral sciences. Such an approach holds promise for uncovering more effective methods to induce and measure cognitive biases, providing deeper insights into the mechanisms underlying biased decision-making. This diversification of investigative approaches not only aims to refine bias induction methodologies but also contributes to a broader understanding of the complex processes involved in bias formation and expression. This exploration aligns with ongoing efforts in the field to continually enhance and expand research methodologies, fostering a more comprehensive and accurate understanding of cognitive biases across diverse contexts.

Additionally, future research should enhance this algorithm by integrating real-time strategies for mitigating bias. The ability to replicate bias through agents offers a unique pathway to utilize this information for identifying and addressing bias as it occurs. By integrating mechanisms that recognize and counteract bias in real time, the algorithm holds the potential to significantly improve decision-making processes, making them more dynamic and responsive. Furthermore, this avenue of exploration could be extended by investigating biases within human-AI teams. This entails a focus on understanding biases that may arise in collaborative settings involving both human and artificial intelligence entities. Examining strategies to mitigate bias within these teams has the potential to lead to more nuanced and effective interventions, ultimately contributing to the development of fair and unbiased decision-making frameworks in intricate, interactive environments.

Bibliography

[1] A. Jacovi, A. Marasović, T. Miller, and Y. Goldberg, "Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in ai," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, (New York, NY, USA), p. 624–635, Association for Computing Machinery, 2021.

[2] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 46, no. 1, p. 50–80, 2004.

[3] O. Vereschak, G. Bailly, and B. Caramiaux, "How to evaluate trust in ai-assisted decision making? a survey of empirical methodologies," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, oct 2021.

[4] I. Seeber, E. Bittner, R. O. Briggs, T. de Vreede, G.-J. de Vreede, A. Elkins, R. Maier, A. B. Merz, S. Oeste-Reiß, N. Randrup, and et al., "Machines as teammates: A research agenda on ai in team collaboration," *Information amp;amp; Management*, vol. 57, no. 2, p. 103174, 2020.

[5] V. Lai, C. Chen, Q. V. Liao, A. Smith-Renner, and C. Tan, "Towards a science of human-ai decision making: A survey of empirical studies," 2021.

[6] A. Smith-Renner, R. Fan, M. Birchfield, T. Wu, J. Boyd-Graber, D. S. Weld, and L. Findlater, "No explainability without accountability: An empirical study of explanations and feedback in interactive ml," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, (New York, NY, USA), p. 1–13, Association for Computing Machinery, 2020.

[7] G. Bansal, B. Nushi, E. Kamar, W. Lasecki, D. Weld, and E. Horvitz, "Beyond accuracy: The role of mental models in human-ai team performance," 10 2019.

[8] G. Bansal, T. Wu, J. Zhou, R. Fok, B. Nushi, E. Kamar, M. T. Ribeiro, and D. Weld, "Does the whole exceed its parts? the effect of ai explanations on complementary team performance," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, (New York, NY, USA), Association for Computing Machinery, 2021.

[9] G. Bansal, B. Nushi, E. Kamar, E. Horvitz, and D. S. Weld, "Is the most accurate ai the best teammate? optimizing ai for teamwork," 2021.

[10] Y. Zhong, "Study on cognitive decision support based on learning and improvement of mental models," *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, 2008.

[11] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *Science*, vol. 185, no. 4157, p. 1124–1131, 1974.

[12] R. Pohl, *Cognitive illusions: Intriguing phenomena in judgement, thinking and memory*. Routledge, 2022.

[13] R. Bromme, F. W. Hesse, and H. Spada, *Barriers and Biases in Computer-Mediated Knowledge Communication: And How They May Be Overcome*. Springer Publishing Company, Incorporated, 1st ed., 2010.

[14] J. Janssen and P. A. Kirschner, "Applying collaborative cognitive load theory to computer-supported collaborative learning: Towards a research agenda," *Educational Technology Research and Development*, vol. 68, no. 2, p. 783–805, 2020.

[15] A. Bertrand, R. Belloum, J. R. Eagan, and W. Maxwell, "How cognitive biases affect xai-assisted decision-making: A systematic review," AIES '22, (New York, NY, USA), p. 78–91, Association for Computing Machinery, 2022.

[16] C. Rastogi, Y. Zhang, D. Wei, K. R. Varshney, A. Dhurandhar, and R. Tomsett, "Deciding fast and slow: The role of cognitive biases in ai-assisted decision-making," 2022.

[17] D. R. Kretz, "Experimentally evaluating bias-reducing visual analytics techniques in intelligence analysis," *Cognitive Biases in Visualizations*, p. 111–135, 2018.

[18] R. J. Heuer, *Psychology of intelligence analysis*. Center for the Study of Intelligence, Central Intelligence Agency, 1999.

[19] N. Boukhelifa, C. R. Johnson, and K. Potter, "Visualization and decision making design under uncertainty," *IEEE Comput. Graph. Appl.*, vol. 43, p. 23–25, sep 2023.

[20] A. Küper, G. Lodde, E. Livingstone, D. Schadendorf, and N. Krämer, "Mitigating cognitive bias with clinical decision support systems: An experimental study," *Journal of Decision Systems*, p. 1–20, 2023.

[21] E. Dimara, G. Bailly, A. Bezerianos, and S. Franconeri, "Mitigating the attraction effect with visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, p. 850–860, 2019.

[22] J. M. Echterhoff, M. Yarmand, and J. McAuley, "Ai-moderated decision-making: Capturing and balancing anchoring bias in sequential decision tasks," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[23] M. Pohl, "Cognitive biases in visual analytics—a critical reflection," *Cognitive Biases in Visualizations*, p. 177–184, 2018.

[24] T. Kliegr, Bahn´ik, and J. Fürnkranz, "A review of possible effects of cognitive biases on interpretation of rule-based machine learning models," *Artificial Intelligence*, vol. 295, p. 103458, 2021.

[25] C. North, R. Chang, A. Endert, W. Dou, R. May, B. Pike, and G. Fink, "Analytic provenance: Process+interaction+insight," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, (New York, NY, USA), p. 33–36, Association for Computing Machinery, 2011.

[26] J. A. Cottam and L. M. Blaha, "Bias by default?," *Cognitive Biases in Visualizations*, p. 43–58, 2018.

[27] E. Wall, A. Arcalgud, K. Gupta, and A. Jo, "A markov model of users' interactive behavior in scatterplots," *2019 IEEE Visualization Conference (VIS)*, 2019.

[28] D. Gotz, S. Sun, and N. Cao, "Adaptive contextualization: Combating bias during high-dimensional visualization and data selection," in *Proceedings of the 21st International Conference on Intelligent User Interfaces*, IUI '16, (New York, NY, USA), p. 85–95, Association for Computing Machinery, 2016.

[29] E. Wall, A. Narechania, A. Coscia, J. Paden, and A. Endert, "Left, right, and gender: Exploring interaction traces to mitigate human biases," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 966–975, 2022.

[30] E. Wall, L. Blaha, C. Paul, and A. Endert, "A formative study of interactive bias metrics in visual analytics using anchoring bias," *Human-Computer Interaction – INTERACT 2019*, p. 555–575, 2019.

[31] E. Wall, L. M. Blaha, L. Franklin, and A. Endert, "Warning, bias may occur: A proposed approach to detecting cognitive bias in interactive visual analytics," *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2017.

[32] D. Streeb, M. Chen, and D. A. Keim, "The biases of thinking fast and thinking slow," *Cognitive Biases in Visualizations*, p. 97–107, 2018.

[33] S. Leavy, B. O'Sullivan, and E. Siapera, "Data, power and bias in artificial intelligence," 2020.

[34] J. Silberg and J. Manyika, "Notes from the ai frontier: Tackling bias in ai (and in humans)," *McKinsey Global Institute*, Jun 2019.

[35] J. Foerderer, "Should we trust web-scraped data?," 2023.

[36] R. Schwartz, A. Vassilev, K. Greene, L. Perine, A. Burt, and P. Hall *Towards a standard for identifying and managing bias in artificial intelligence*, 2022.

[37] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[38] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for nlp," *arXiv preprint arXiv:2105.03075*, 2021.

[39] A. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, 1999.

[40] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," 2010.

[41] D. A. van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, p. 1–50, 2001.

[42] "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, p. 139–144, 2020.

[44] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[45] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International conference on machine learning*, pp. 2642–2651, PMLR, 2017.

[46] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[47] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.

[48] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, "Bagan: Data augmentation with balancing gan," *arXiv preprint arXiv:1803.09655*, 2018.

[49] N. Mukhtar, L. Batina, S. Picek, and Y. Kong, "Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices." Cryptology ePrint Archive, Paper 2021/991, 2021. https://eprint.iacr.org/2021/991.

[50] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[51] S. Motamed, P. Rogalla, and F. Khalvati, "Data augmentation using generative adversarial networks (gans) for gan-based detection of pneumonia and covid-19 in chest x-ray images," *Informatics in Medicine Unlocked*, vol. 27, p. 100779, 2021.

[52] Y. Lu, D. Chen, E. Olaniyi, and Y. Huang, "Generative adversarial networks (gans) for image augmentation in agriculture: A systematic review," *Computers and Electronics in Agriculture*, vol. 200, p. 107208, 2022.

[53] S. Choi, J. Kim, and H. Yeo, "Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning," *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103091, 2021.

[54] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 399–410, 10 2016.

[55] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," in *Advances in Neural Information Processing Systems*, 2019.

[56] S. Shao, P. Wang, and R. Yan, "Generative adversarial networks for data augmentation in machine fault diagnosis," *Computers in Industry*, vol. 106, pp. 85–93, 2019.

[57] S. I. Nikolenko, "Synthetic data for deep learning," 2019.

[58] M. Hernandez, G. Epelde, A. Alberdi, R. Cilla, and D. Rankin, "Synthetic data generation for tabular health records: A systematic review," *Neurocomputing*, vol. 493, p. 28–45, 2022.

[59] A. Torfi, E. A. Fox, and C. K. Reddy, "Differentially private synthetic medical data generation using convolutional gans," *Information Sciences*, vol. 586, p. 485–500, 2022.

[60] A. Chen, "A novel graph methodology for analyzing disease risk factor distribution using synthetic patient data," *Healthcare Analytics*, vol. 2, p. 100084, 2022.

[61] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," *Medical Image Analysis*, vol. 58, p. 101552, dec 2019.

[62] Z. He and W. Zhou, "Generation of synthetic full-scale burst test data for corroded pipelines using the tabular generative adversarial network," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105308, 2022.

[63] B. Yilmaz and R. Korn, "Synthetic demand data generation for individual electricity consumers : Generative adversarial networks (gans)," *Energy and AI*, vol. 9, p. 100161, 2022.

[64] S. Welten, A. Holt, J. Hofmann, L. Schelter, E.-M. Klopries, T. Wintgens, and S. Decker, "Synthetic rainfall data generator development through decentralised model training," *Journal of Hydrology*, vol. 612, p. 128210, 2022.

[65] D. B. Rubin, "Multiple imputation for nonresponse in surveys," *Wiley Series in Probability and Statistics*, 1987.

[66] H. Ping, J. Stoyanovich, and B. Howe, "Datasynthesizer: Privacy-preserving synthetic datasets," in *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, SSDBM '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[67] B. Theodorou, C. Xiao, and J. Sun, "Synthesize high-dimensional longitudinal electronic health records via hierarchical autoregressive language model," *Nature Communications*, vol. 14, no. 1, 2023.

[68] D. Alvarez-Melis, V. Garg, and A. T. Kalai, "Why gans are overkill for nlp," 2022.

[69] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative adversarial networks in time series: A systematic literature review," *ACM Comput. Surv.*, vol. 55, feb 2023.

[70] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," 2017.

[71] M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer, "Quant GANs: deep generation of financial time series," *Quantitative Finance*, vol. 20, pp. 1419–1440, apr 2020.

[72] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative adversarial networks in time series: A systematic literature review," *ACM Comput. Surv.*, vol. 55, feb 2023.

[73] X. Liu, H. Chen, and C. Andris, "trajgans : Using generative adversarial networks for geo-privacy protection of trajectory data (vision paper)," 2018.

[74] J. Rao, S. Gao, Y. Kang, and Q. Huang, "Lstm-trajgan: A deep learning approach to trajectory privacy protection," 2020.

[75] K. Zhu, S. Zhang, J. Li, D. Zhou, H. Dai, and Z. Hu, "Spatiotemporal multi-graph convolutional networks with synthetic data for traffic volume forecasting," *Expert Systems with Applications*, vol. 187, p. 115992, 2022.

[76] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. The MIT Press, 2020.

[77] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[78] M. Grzes, "Reward shaping in episodic reinforcement learning," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017* (K. Larson, M. Winikoff, S. Das, and E. H. Durfee, eds.), pp. 565–573, ACM, 2017.

[79] E. Wiewiora, G. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, p. 792–799, AAAI Press, 2003.

[80] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, p. 3352–3358, AAAI Press, 2015.

[81] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: A new approach of reward shaping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15931–15941, 2020.

[82] S. Schaal, "Learning from demonstration," 1997.

[83] E. Dimara, S. Franconeri, C. Plaisant, A. Bezerianos, and P. Dragicevic, "A task-based taxonomy of cognitive biases for information visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 2, p. 1413–1432, 2020.

[84] D. Kahneman, *Thinking, fast and slow*. Farrar, Straus and Giroux, 2015.

[85] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *Science*, vol. 185, no. 4157, pp. 1124–1131, 1974.

[86] A. Furnham and H. C. Boo, "A literature review of the anchoring effect," *The Journal of Socio-Economics*, vol. 40, no. 1, p. 35–42, 2011.

[87] A. C. Valdez, M. Ziefle, and M. Sedlmair, "Priming and anchoring effects in visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, p. 584–594, 2018.

[88] I. Cho, R. Wesslen, A. Karduni, S. Santhanam, S. Shaikh, and W. Dou, "The anchoring effect in decision-making with visual analytics," *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2017.

[89] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, (USA), p. 228–231, Association for Computational Linguistics, 2007.

[90] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[91] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[92] A. Gleave, M. Taufeeque, J. Rocamonde, E. Jenner, S. H. Wang, S. Toyer, M. Ernestus, N. Belrose, S. Emmons, and S. Russell, "imitation: Clean imitation learning implementations." arXiv:2211.11972v1 [cs.LG], 2022.

[93] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11 – 15, 2008.

[94] Biard, Steven, E. Loper, and E. Klein, *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[95] P. Community, "Pygame: Python game development," 2000–present.

[96] K. Poels, Y. de Kort, and W. IJsselsteijn, *D3.3 : Game Experience Questionnaire: development of a self-report measure to assess the psychological impact of digital games*. Technische Universiteit Eindhoven, 2007.