# Creating a Privacy-Preserving Recommendation System Through Adversarial Training in Federated Reinforcement Learning

**Kory W. Rosen**
Department of Computer Science
Tulane University
New Orleans, LA 70118
`krosen5@tulane.edu`

**Keith J. Mitchell**
Department of Computer Science
Tulane University
New Orleans, LA 70118
`kmitchell2@tulane.edu`

## Abstract

Reinforcement learning is often used as a powerful tool for optimizing user engagement and decision making in recommendation systems; however, the susceptibility of these systems to adversarial manipulation remains a widely understudied area, particularly in federated learning environments where privacy and decentralized architectures present unique challenges. This study investigates the impact of adversarial perturbations on recommendation systems based on federated reinforcement learning, focusing on training mechanisms to limit rewards. We propose an adversarial framework that employs the Fast Gradient Sign Method (FGSM) to introduce perturbations into the agents' state space during training. This approach minimizes the reward by creating adversarial states after a predefined number of episodes and challenges the agents' ability to adapt under malicious conditions. The framework incorporates metrics for both clean and adversarial state evaluations, enabling a comprehensive analysis of the perturbation's impact on Q-value estimation and overall agent performance. Unlike traditional methods emphasizing defensive mechanisms, this work explores how adversarial training can manipulate reward dynamics to highlight vulnerabilities in reinforcement learning systems. We implement a federated setting where multiple agents interact with a shared recommendation system. Experimental results demonstrate a significant degradation of reward optimization under adversarial influence, offering critical insights into the interplay between federated reinforcement learning and adversarial perturbations. This study lays the groundwork for understanding adversarial risks in decentralized learning environments and informs the development of future recommendation system strategies more resilient to adversarial exploitation.

## 1 Introduction

In recent years, recommendation systems have seen tremendous growth, playing a crucial role in personalizing user experiences across a variety of industries. From e-commerce platforms to media services, the ability to suggest relevant content is a major part of user engagement and satisfaction. However, the centralization of user data, often required by traditional models, has raised significant concerns regarding privacy and security. To address these issues, federated learning offers a promising solution that enables the development of models that can be trained without the need for raw user data to leave their devices. This decentralized approach helps protect user privacy, but introduces its own set of challenges, including adversarial vulnerabilities and the potential for biased model updates. To overcome these, there is a need for robust, privacy-preserving federated recommendation systems that can learn from adversarial training while working towards long-term reward optimization.

## 1.1 Background

As the digital world becomes more sophisticated in its offerings, the need for personalized experiences has grown exponentially. Recommendation systems, powered by machine learning, have become a critical tool in achieving this personalization. They leverage large volumes of user interaction data to predict and suggest content that matches user preferences. Traditionally, these systems have relied on centralized data architectures, which raise significant privacy and security concerns due to the aggregation of sensitive user information in one location. In response, federated learning has emerged as a viable alternative, offering a way to train models collaboratively across multiple devices without centralizing data. However, while federated learning addresses privacy concerns, it also introduces new challenges, such as adversarial attacks. These challenges must be addressed to create effective and secure recommendation systems.

### 1.1.1 Recommendation Systems

Recommendation systems are essential for providing personalized content to users across a variety of digital platforms, including streaming services and social media platforms. These systems analyze historical user data, such as clicks, purchases, interactions, and other forms of engagement, to predict and make recommendations that align with individual preferences. Over time, recommendation systems have become more sophisticated by integrating complex algorithms that take into account both user behavior and the broader context of the content being recommended. Traditional approaches to building these systems involve collecting and storing vast amounts of data on centralized servers, allowing models to be trained on all user interactions in a single location. This requires users to share sensitive data, making them vulnerable to data breaches and potential misuse. This has led to the demand for privacy-preserving approaches, such as federated learning, which allow for the development of recommendation systems while mitigating privacy risks.

### 1.1.2 Federated Learning

Federated learning is a machine learning approach that enables the training of models across multiple agents or decentralized data sources without the need to centralize the data. Instead of sharing said data, federated learning facilitates the exchange of model updates between agent devices and a central server. This approach not only preserves privacy by keeping sensitive data at the local level but also allows for efficient, large-scale training of models. Federated learning has gained considerable attention in recent years, particularly in the context of recommendation systems, as it enables the creation of models that respect privacy laws and regulations. Despite its advantages, federated learning comes with its own set of challenges, such as dealing with data across devices, ensuring the integrity of model updates, and managing the computational complexity of local training. These challenges become even more pronounced when considering adversarial threats that can compromise model performance and security.

### 1.1.3 Adversarial Training

Adversarial training is a defense strategy that involves exposing a model to adversarial perturbations during training to enhance its robustness. These perturbations, generated by methods like the Fast Gradient Sign Method (FGSM), simulate attacks on the model's input states to identify vulnerabilities. In the context of federated recommendation systems, adversarial training is critical in ensuring that the model can operate effectively under malicious conditions. This technique not only tests the model's resilience but also helps adapt its behavior to minimize the impact of adversarial states, aligning with long-term reward optimization.

## 1.2 Identification of the Problem

As the demand for personalized services continues to grow, so does the complexity of building scalable and privacy-preserving recommendation systems. In particular, adversarial vulnerabilities in federated systems pose a significant risk to the integrity of model training, as attackers can manipulate the model by introducing adversarial updates. This calls for the development of robust federated recommendation systems that are resilient to adversarial attacks and capable of delivering accurate, privacy-preserving recommendations across multiple platforms.

#### 1.2.1 Justification of the usage of Reinforcement Learning

Reinforcement learning (RL) offers a natural framework for recommendation systems, as it focuses on learning a sequence of actions that maximize long-term rewards. In the context of recommendation systems, RL allows the agents involved to continuously adjust their recommendations based on user feedback or engagement rates, optimizing the recommendation policy over time. The key advantage of RL in these systems lies in its ability to dynamically adapt to changes in behavior and preference. Furthermore, the RL framework is well suited for federated learning, as it enables agents on different devices to act independently while optimizing a common objective. This makes RL an ideal approach for training decentralized recommendation models that not only learn from user engagement, but also mitigate adversarial risks and privacy concerns.

#### 1.2.2 Example of Potential Challenges

Federated RL for recommendation systems presents several key challenges. One major issue is adversarial attacks, in which malicious agents or states attempt to manipulate the model by introducing perturbations, which can degrade the performance and security of the model. Federated RL systems also face difficulties in maintaining consistency with decentralized agents, as interactions and user feedback are not uniformly distributed. These challenges necessitate the development of robust training strategies that can handle adversarial disruptions, ensure model stability, and optimize performance across distributed agents.

## 2 Related Works

The development of robust and efficient federated recommendation systems has had a considerable growth in attention over recent years. Due to this rise, researchers have been exploring various approaches to address challenges such as privacy preservation, adversarial robustness, and optimization of long-term rewards. This section highlights key contributions in this domain that directly relate to the solution proposed. This includes topics of Slate-Q, FedSlate, FGSM, and the introduction of adversarial perturbations or attacks in RL settings.

### 2.1 Slate-Q

Slate-Q is a reinforcement learning algorithm designed specifically for recommendation systems that operate on slates, or ordered lists of items, rather than single-item recommendations. Unlike traditional methods, Slate-Q focuses on the relationships between items within a slate and aims to optimize the collective reward of the entire slate rather than treating each item independently. This allows the algorithm to account for user behavior, such as interactions with multiple items in a list or the impact of slate order on user preferences. The algorithm employs a Markov Decision Process (MDP) framework to model user interactions and utilizes Q-learning to optimize the recommendation policy. Slate-Q's ability to evaluate the utility of entire slates makes it a powerful tool for building recommendation systems that prioritize user engagement over a long time period. It does, however, have a heavy reliance on centralized data collection which poses challenges for privacy preservation. This makes it a less viable option for federated learning contexts without modifications. [3]

### 2.2 FedSlate

FedSlate extends the principles of Slate-Q to a federated learning setting, addressing privacy concerns by decentralizing the training process. Instead of relying on a central repository of user data, FedSlate enables collaborative training of models across distributed devices, ensuring that data never leaves the user's device. By incorporating federated learning techniques, FedSlate maintains user privacy while optimizing slate-wise recommendations. FedSlate incorporates privacy-preserving mechanisms like differential privacy and secure aggregation to further protect sensitive user information. Despite these advancements, FedSlate remains vulnerable to adversarial attacks, which can exploit the decentralized nature of federated learning to compromise model performance. [2]

### 2.3 Adversarials in RL

Adversarial attacks in RL present a significant challenge by targeting the integrity of the learning process. These attacks are perturbations, designed to mislead the agent's decision-making either by manipulating the observations received from the environment or by altering the agent's policy directly. [5] In recommendation systems, such adversarial influences can lead to suboptimal or even harmful recommendations, undermining user trust and the system's effectiveness.

In the context of federated RL, adversarial attacks pose unique risks due to the decentralized nature of data and learning. Privacy-preserving constraints in federated setups often obscure the full state information, making the detection and mitigation of adversarial impacts even more complex. [1] Attacks such as Fast Gradient Sign Method (FGSM) exploit model vulnerabilities by introducing imperceptible yet impactful perturbations, causing agents or the federated server to deviate from optimal behavior.

Addressing these challenges requires integrating adversarial training techniques into RL models. This involves simulating adversarial states during training to improve the agent's or the server's robustness and resilience. By incorporating such methods, federated RL systems can enhance their reliability, extending to take into account adversarially compromised environments. [4]

## 3 Problem Formulation

### 3.1 Objective

The primary objective of our research is to train a federated recommendation system to robustly withstand adversarial attacks while preserving user privacy. This involves optimizing the system to deliver high-quality recommendations, even in the presence of perturbations introduced by adversarial states. The federated system must learn from the perturbed recommendations and adversarial states to attempt to maximize rewards while the adversarial states seek to minimize the rewards.

### 3.2 Environment: Recsim

The simulation environment for this study is based on RecSim, a highly customizable platform for modeling and analyzing recommendation systems. RecSim provides a realistic framework for testing and evaluating recommendation policies under dynamic user interactions. By simulating user behaviors, feedback loops, and slate-wise interactions, RecSim serves as an ideal environment for investigating the robustness of federated recommendation systems.

For this research, RecSim is extended to incorporate federated learning mechanisms and adversarial state training, enabling a comprehensive study of privacy, robustness, and recommendation quality in a distributed setting. The integration of adversarial states into the environment further facilitates the evaluation of system resilience under compromised conditions. [2]

### 3.3 Challenges

Several key challenges define the problem space:

Adversarial Vulnerabilities: Adversarial perturbations, such as those generated through the use of FGSM, can mislead the recommendation system, degrading its performance and reliability.

Privacy Constraints: Federated setups restrict direct data sharing among agents, creating partial observability that complicates robust policy learning.

Scalability: Ensuring efficient training and deployment across distributed environments with limited communication and computational overhead is critical.

Reward Optimization: Federated RL systems must optimize slate-wise rewards while addressing conflicting priorities, such as robustness and recommendation quality.

### 3.4 Framework and Architecture

The framework for this study integrates federated reinforcement learning with adversarial training mechanisms. It involves the following components:

Agents: Collaborative agents that operate within the federated setup, each learning to optimize recommendations based on local data and shared policy updates.

Adversarial Training: FGSM is employed to generate adversarial states during training, exposing vulnerabilities and strengthening system robustness.

Federated Learning Protocol: A decentralized protocol coordinates agents without direct data sharing, ensuring privacy and compliance with federated learning principles.

Evaluation Metrics: Metrics such as slate-wise reward, adversarial robustness, and privacy compliance measure the effectiveness of the proposed system.

#### 3.4.1 User Preferences within the Environment

This recommendation system operates within an environment where user preferences are based on two key attributes: kaleness and chocolate. Kaleness represents the "healthiness" of a document, with values ranging from 0 to 1. A higher kaleness value indicates a more "healthy" document, which aligns with the user's preference for health-conscious recommendations. Chocolate, on the other hand, represents the user's preference for indulgent or "sweet" recommendations, with higher values reflecting a stronger inclination toward chocolate-like preferences. These two attributes combine to form a user's overall preferences, which the system uses to rank documents and generate recommended slates. The system also incorporates a time budget, simulating the user's patience while interacting with the recommendation system. As the user engages with the system, their time budget decreases, imposing a constraint on how many recommendations they can evaluate. This adds complexity to the recommendation process, as the system must not only align with the user's preferences but also respect their limited time. The time budget is an essential factor in designing the recommendation system, as it ensures that recommendations are both engaging and time-efficient.

### 3.5 Markov Decision Process (MDP)

The federated recommendation system operates through two distinct Markov Decision Processes (MDPs), tailored for agents Alpha ($\alpha$) and Beta ($\beta$). Each MDP captures the respective agent's learning framework, interactions with the federated server, and involvement with adversarial states.

#### 3.5.1 Agent $\alpha$

**State Space ($S_\alpha$)** The state space for Agent $\alpha$ consists of the following components:

- **User Features ($\mathbb{R}^{d_u}$)**: Encodes the user's profile, preferences, and demographic information.
- **Document Features ($\mathbb{R}^{d_d}$)**: Represents the attributes of $N$ candidate items in the recommendation pool.
- **Engagement History ($\mathbb{R}^{d_e}$)**: Tracks user interactions over the last $k$ sessions, reflecting behavioral patterns.

The overall state dimension is:

$$S_\alpha = d_u + N \times d_d + k \times d_e$$

**Action Space ($A_\alpha$)** Agent $\alpha$ selects slates of items from the set of candidates ($I$), where each slate contains $k$ items:

$$A_\alpha = \{a \mid a \subseteq I, |a| = k\}$$

**Transition Probability ($P_\alpha$)** The transition dynamics are defined as:

$$P(s' \mid s, a) = \sum_{i \in \alpha} P(i \mid s, a) \times P(s' \mid s, i)$$

Where:

- $s$: Current state.
- $s'$: Next state with the same dimensionality as $s$.
- $a$: Selected slate of size $k$.
- $i$: Selected item from the slate, including the null selection ($\perp$).

**Reward Function** ($r_\alpha$) Agent $\alpha$'s reward function combines user engagement and adversarial robustness:

$$r_\alpha(s, a) = \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] - \lambda(Q_{f\alpha} - Q^*_\alpha)^2$$

Where:

- $R_t$: User engagement for the selected item.
- $Q_{f\alpha} - Q^*_\alpha$: Deviation of attacked Q-values from clean Q-values, penalized by $\lambda$.

### 3.5.2 Agent $\beta$

**State Space** ($S_\beta$) Agent $\beta$'s state includes:

- **User Features** ($\mathbb{R}^{d_u}$)
- **Document Features** ($\mathbb{R}^{d_d}$)
- **Engagement History** ($\mathbb{R}^{d_e}$)

The state dimension is reduced to:

$$S_\beta = d_u + N \times d_d + d_e$$

**Action Space** ($A_\beta$) Agent $\beta$ selects slates similarly:

$$A_\beta = \{a \mid a \subseteq I, |a| = k\}$$

**Transition Probability** ($P_\beta$) Defined similarly to $P_\alpha$:

$$P(s' \mid s, a) = \sum_{i \in \beta} P(i \mid s, a) \times P(s' \mid s, i)$$

**Reward Function** ($r_\beta$) Agent $\beta$'s reward incorporates cross-platform influences:

$$r_\beta(s, a) = \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a]$$

Where $R_t$ is inferred from rewards generated by Platform A.

These MDPs for Agents $\alpha$ and $\beta$ enable the system to adapt to adversarial threats and optimize user engagement within a federated learning environment. Both agents utilize a discount factor ($\gamma$) to weigh immediate vs. future rewards. This factor reflects the agents' long-term optimization goals, ensuring that recommendations not only generate immediate user satisfaction but also contribute to sustained engagement over time.

## 3.6 Federated Q-Value Sharing

Our federated Q-value sharing mechanism addresses the challenge of collaborative learning across platforms while preserving privacy. Traditional federated learning approaches typically share model parameters or gradients, potentially leaking sensitive information. Our approach innovates by sharing only Q-values, creating a stronger privacy barrier while enabling effective knowledge transfer. The architecture balances collaboration and privacy using three components: Agent $\alpha$, Agent $\beta$, and a federated "server" (AgentFed). The asymmetric design allows agent $\alpha$ to maintain a rich state representation incorporating user features ($d_u$), document features ($d_d$), and engagement history ($d_e$) while letting Agent $\beta$ operate with a more straightforward state space that excludes historical data. This ensures privacy while creating complementary perspectives on user behavior.

The federation process works through the following Q-value sharing mechanism:

$$Q_{fed}(s, a) = F(Q_\alpha(s, a), Q_\beta(s, a))$$

This allows knowledge transfer without direct data exchange. $Q_{fed}(s, a)$ represents a synthesized recommendation value for state $s$ and action $a$, while $Q_\alpha$ and $Q_\beta$ represent each agent's individual assessments. The federation function $F$ combines these individual assessments as follows:

$$F(Q_\alpha, Q_\beta) = \phi(W[Q_\alpha \| Q_\beta] + b)$$

Here, $\phi$ represents a non-linear activation function that helps capture complex relationships between agents' perspectives, $W$ represents the weight matrix learning the optimal way to combine these perspectives, and $[Q_\alpha \| Q_\beta]$ represents their concatenated wisdom. The bias term $b$ helps adjust the baseline of these combined recommendations.

Each agent employs different TD-learning processes tailored to their roles. Agent $\alpha$ performs TD updates specifically focused on clicked recommendations:

$$Q_\alpha(s, a) \leftarrow Q_\alpha(s, a) + \alpha[R + \gamma \max_{a'} Q_\alpha(s', a') - Q_\alpha(s, a)]$$

The update process identifies relevant user interactions through:

$$\text{selected\_items} = \text{batch\_actions} \odot \text{batch\_clicks}$$

where $\odot$ represents element-wise multiplication. This ensures updates occur only for recommendations that received explicit user engagement.

On the other hand, Agent $\beta$ employs a slate-wise TD learning approach that considers the entire recommendation set:

$$Q_\beta(s, a) \leftarrow \mathbb{E}_{slate}[\sum i \in slate p(i|s, a) Q_\beta(s, i)]$$

For each experience batch, the slate-wise update computes:

$$\text{greedy\_Q} = \sum_{i \in slate} p(i|s, \text{slate}) Q_\beta(s, i)$$

where $p(i|s, \text{slate})$ represents the probability of selecting item $i$ from the recommended slate in state $s$.

To maintain stable learning, we implement a periodic synchronization protocol:

$$\theta_{target} \leftarrow \tau \theta_{online} + (1 - \tau)\theta_{target}$$

Here, $\theta_{target}$ represents the stable, consensus understanding, while $\theta_{online}$ represents new insights. The parameter $\tau$ $(0 < \tau \leq 1)$ controls how quickly we incorporate new knowledge, preventing any sudden shifts that could destabilize the system.

The system's learning objective balances individual and collective goals:

$$\mathcal{L} = \mathcal{L}_\alpha + \mathcal{L}_\beta + \lambda \mathcal{L}_{fed}$$

While $\mathcal{L}_\alpha$ and $\mathcal{L}_\beta$ ensure each agent maintains its expertise, $\mathcal{L}_{fed}$ promotes effective collaboration. The parameter $\lambda$ controls how much we prioritize collaborative learning vs. individual specialization.

We maintain strong privacy guarantees while enabling rich knowledge transfer by sharing only processed Q-values rather than raw data or direct model parameters.

## 4 Preliminary Solutions

In this section, we present our approaches to implementing adversarial attacks within a federated RL recommendation system and discuss challenges encountered during implementation.

### 4.1 Environment Implementation Challenges

Our first initial challenge was addressing compatibility issues with the RecSim environment. As TensorFlow evolved and deprecated the TensorFlow estimator framework, we found that RecSim's reliance on this outdated framework created significant obstacles. To overcome this, we revised key components of the environment's package, updating dependencies and ensuring compatibility with the latest TensorFlow versions. This allowed us to preserve the functionality of RecSim while ensuring its proper operation within a modern TensorFlow setup.

### 4.2 Adversarial Training with FGSM

We chose FGSM as our adversarial attack due to its suitability for reinforcement learning, as it uses gradient-based methods to generate adversarial examples, making it adaptable to state-action environments. Our implementation evolved through several stages to fit our federated reinforcement learning framework. Initially, we tried torchattacks but encountered compatibility issues, particularly with the continuous state-space data used in our environment. Torchattacks, like many adversarial attack frameworks, is optimized for discrete or image-based inputs and struggled to adapt to the continuous states of reinforcement learning tasks. Following this, we explored CleverHans, but it also fell short due to its primary design focus on supervised learning tasks. Neither framework was equipped to handle the complexities of a federated reinforcement learning model like ours.

With the help of our teaching assistant, Xiaolin Sun, we adapted a version of the Fast Gradient Sign Method (FGSM) initially designed for state-adversarial Deep Q-Networks (DQN). This version of FGSM's objective involves perturbing the input state $s$ to maximize an adversarial loss, forcing the Q-network to produce incorrect predictions. Our attack objective was reformulated to:

$$L_{\text{attack}} = -Q(s, a),$$

where $Q(s, a)$ represents the Q-value for state $s$ and action $a$. By maximizing $L_{\text{attack}}$, the adversary minimizes $Q(s, a)$, disrupting the agent's ability to recognize/execute optimal actions. This negative loss functions as a "gain" for the adversary, as reducing the Q-value undermines the agent's policy.

FGSM uses the gradient of $Q(s, a)$ with respect to the state $s$ to craft adversarial states. The gradient of the adversarial objective is expressed as:

$$\nabla_s L_{\text{attack}} = -\nabla_s Q(s, a).$$

Therefore, the adversarial state is generated as:

$$s_{\text{adv}} = s + \epsilon \cdot \text{sign}(\nabla_s Q(s, a)),$$

where $\epsilon$ controls the perturbation magnitude. This introduces subtle but deliberate changes to the state $s$ to mislead the Q-network. For instance, if $s = [0.5, 0.2, 0.8]$ and $\nabla_s Q(s, a) = [0.3, -0.1, 0.5]$, the adversarial state would be $s_{\text{adv}} = [0.5, 0.2, 0.8] + \epsilon \cdot [1, -1, 1]$. These perturbations can cause significant deviations in the Q-value predictions.

To apply FGSM in our federated reinforcement learning context, we had to adapt the methodology to align with our Q-network's dynamic state-action architecture. Unlike supervised learning, where FGSM optimizes a fixed loss function $L(X, y)$, reinforcement learning requires an adversarial objective based on the state-action pair. Xiaolin identified a supervised-learning-based FGSM implementation that we adapted to handle the dynamic states of reinforcement learning. This required transitioning from a static loss formulation to a dynamic Q-value-based adversarial objective. Additionally, the federated structure of our model posed unique challenges, as perturbations in one agent could propagate through shared Q-values, amplifying their effects across the system.

We implemented a delayed attack start mechanism to ensure initial training stability. Q-learning began after 5,000 steps, and adversarial perturbations were introduced only at 200 episodes. This approach allowed the model to establish a policy baseline before encountering the effects of adversarial states.

In our setup, only Agent Alpha is subjected to adversarial perturbations. Adversarial states disrupt Alpha's Q-network in two critical ways. First, they lower the Q-value for optimal actions,

$Q(s_{\text{adv}}, a_{\text{optimal}})$, making the agent less likely to choose the best actions. Simultaneously, they can increase the Q-values for suboptimal actions, $Q(s_{\text{adv}}, a_{\text{suboptimal}})$, misleading Alpha into favoring poor decisions. This destabilizes Alpha's decision-making process, reducing the effectiveness of its policy.

The impact of these adversarial perturbations extends to the performance metrics of the system itself. While Beta is not directly attacked, the Q-values shared between Alpha and Beta in the federated learning framework mean that perturbations to Alpha's Q-network can propagate to Beta, amplifying the system-wide degradation. Errors introduced by the adversarial states in Alpha influence the shared Q-values, leading to reduced recommendation quality, lower engagement rewards, and poorer user interactions for both agents. This cascading effect highlights the vulnerabilities introduced by the federated structure, where even a single attacked agent can compromise the system as a whole.

### 4.3 Project Scope Refinements

Following guidance from Professor Zizhan Zheng, we made two simplifications to reduce the overall complexity of the project and maintain focus on our core research objectives. First, we eliminated the dedicated security state tracking mechanism to streamline the system and better focus on the direct interactions between adversarial perturbations and federated Q-learning. Second, we removed the double Q-learning defense mechanism, simplifying the analysis of adversarial effects within our federated learning system. These changes were made to avoid overcomplicating the project, as incorporating additional techniques, such as using Opacus (a library that is used to train PyTorch models with Differential Privacy), would have introduced unnecessary complexity.

## 5 Evaluations

### 5.1 Experimental Details

Our experimental evaluation aims to assess the impact of adversarial training on our federated recommendation system. We compare two main configurations: our model that trains on both adversarial and clean data against a baseline model without adversarial training.

#### 5.1.1 Shared Hyperparameters

In both models, the Q-networks use five hidden layers with 4096 units each and Mish activation functions between layers. The level of depth in the architecture allows the networks to learn complex patterns in user behavior. The Mish activation provides smooth gradients that help stabilize training. A final Tanh activation constrains Q-values between -1 and 1, preventing value explosions that could destabilize the learning process. Both models employ a federation network with 2048 units per layer to combine Q-values from the agents. A smaller architecture is used since the federated network focuses on analyzing already-processed Q-values rather than learning raw state-action relationships.

Both models use an Adam optimizer with a learning rate of 0.01 and momentum parameters $\beta_1$ = 0.9, $\beta_2$ = 0.999. These parameters control the model's adaptation to new information, with $\beta_1$ for short-term memory and $\beta_2$ for long-term memory. The experience replay system stores 2000 experiences and trains on batches of 32 samples. Both models begin learning after collecting 2000 initial experiences and perform updates every 3 steps to efficiently use computational resources.

The reinforcement learning parameters remain identical across both models. The discount factor ($\gamma$) being 0.9 creates a balanced focus between immediate and future rewards. The target network updating every 500 steps helps stabilize learning by providing consistent targets. Both use epsilon-greedy exploration starting at 1.0 (pure exploration) and decaying by 0.99995 per step until reaching 0.0 (pure exploitation). This slow decay ensures a thorough exploration of possible recommendations before settling into exploiting learned knowledge.

#### 5.1.2 FGSM Hyperparameters

The perturbation magnitude ($\epsilon$) of 0.01 emerged from extensive experimentation since it was large enough to create meaningful adversarial examples but small enough to maintain learning stability. We begin adversarial training after 200 episodes of clean training, allowing the system to establish baseline performance before introducing adversarial challenges. The attack penalty weight ($\lambda$) of

0.1 balances between two competing objectives: building adversarial robustness and maintaining recommendation quality. When generating adversarial states, we clip perturbations to the range [-1, 1] to ensure they remain within our state space's bounds. This clipping prevents extreme perturbations that could break the learning process while still allowing meaningful adversarial examples.

### 5.2 Evaluation Metrics

We track four key types of metrics to understand how adversarial training affects performance.

Our first comparison examines rewards, which in our recommendation system represent user engagement. For our adversarially-trained model, we measure engagement rewards in both clean states ($R_{\alpha_{clean}}$, $R_{\beta_{clean}}$) and adversarial states ($R_{\alpha_{adv}}$, $R_{\beta_{adv}}$). We compare these against the baseline model's engagement rewards ($R_\alpha$, $R_\beta$), which only sees normal recommendations. This comparison helps us understand whether making our system more robust through adversarial training affects its ability to maintain high user engagement under normal conditions.

Our second comparison looks at exploration patterns through epsilon decay curves. Both models need to balance between trying new recommendations (exploration) and sticking to what works (exploitation). By comparing how quickly each model transitions from pure exploration ($\epsilon$ = 1.0) to pure exploitation ($\epsilon$ = 0.0), we can see whether adversarial training has any effect on the recommendation system's ability to efficiently learn which content users enjoy.

Our third comparison examines loss values, which indicate how well each model is learning to predict user engagement. For the adversarially-trained model, we track prediction errors in both clean states ($\mathcal{L}\alpha_{clean}$, $\mathcal{L}\beta_{clean}$) and adversarial states ($\mathcal{L}\alpha_{adv}$, $\mathcal{L}\beta_{adv}$), comparing them against the baseline model's prediction errors ($\mathcal{L}\alpha$, $\mathcal{L}\beta$). Lower loss values indicate better prediction accuracy, which is crucial for making recommendations that users will engage with.

Our fourth comparison looks at Q-values, which represent each model's estimated long-term value of recommending particular content to users. We track how these estimates develop in our adversarially-trained model under both normal conditions ($Q_{\alpha_{clean}}$, $Q_{\beta_{clean}}$) and when facing adversarial perturbations ($Q_{\alpha_{adv}}$, $Q_{\beta_{adv}}$), comparing them against the baseline model's estimates ($Q_\alpha$, $Q_\beta$). This comparison reveals whether adversarial training affects our system's ability to accurately judge which recommendations will lead to sustained user engagement over time.

### 5.3 Results

#### 5.3.1 Rewards

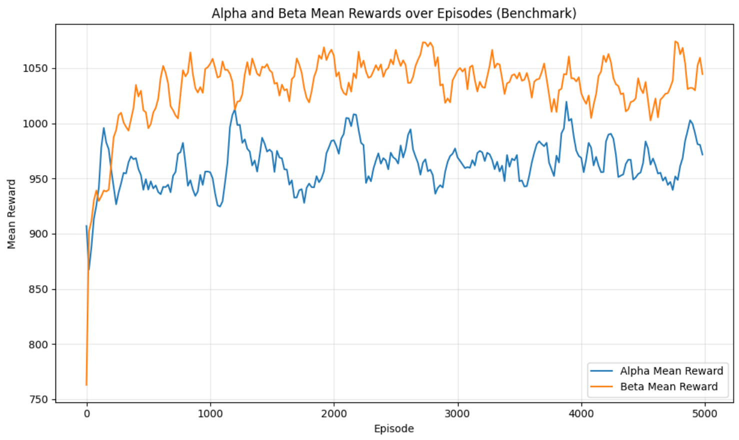

Figure 1: Mean reward for the existing framework.

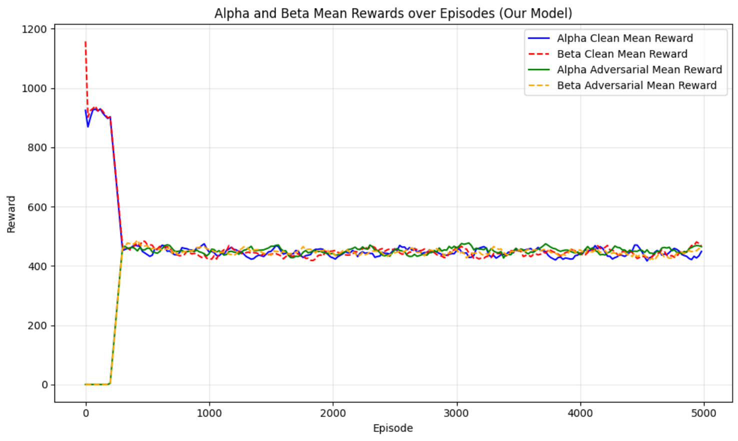

Figure 2: Mean reward for our model.

The reward patterns reveal a critical limitation in our adversarially-trained model. While both models show initial volatility during the burn-in period, their subsequent behaviors highlight important differences in learning capability. The baseline model demonstrates continuous learning and adaptation, maintaining higher reward levels throughout training (Beta: ∼1050, Alpha: ∼950). This suggests it successfully learns to optimize user engagement over time. More importantly, it maintains this higher performance level, indicating sustained learning and adaptation to user preferences. In contrast, our adversarially-trained model shows concerning behavior. After an initial spike to 1200 during early learning, the rewards drop and stabilize around 450 after adversarial training begins - less than half of the baseline model's performance. Most troublingly, these rewards never recover or improve. While our model achieves consistency across clean and adversarial conditions, it does so by converging to a strategy that underperforms.

### 5.3.2 Epsilon Decay

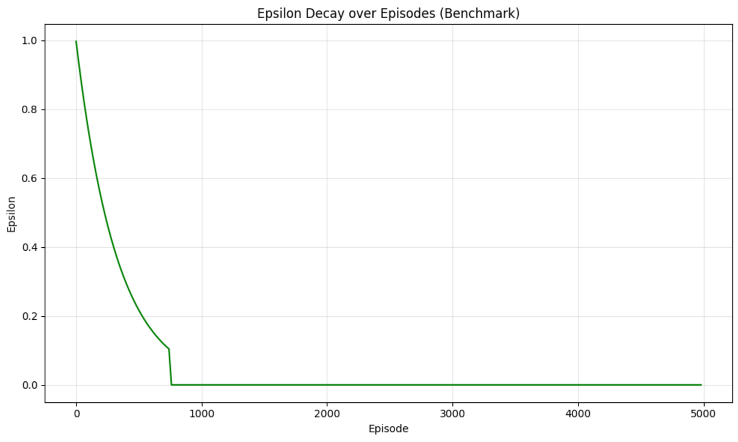

Figure 3: Epsilon Decay for the existing framework.

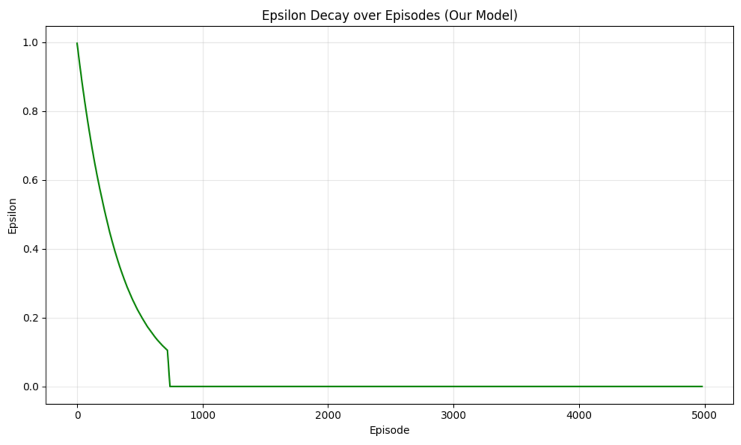

Figure 4: Epsilon Decay for our model.

Both models exhibit similar epsilon decay patterns, starting with high exploration ($\epsilon$ near 1.0) and transitioning to exploitation ($\epsilon$ approaching 0.0). The benchmark model maintains non-zero epsilon values for 760 episodes, while the adversarially-trained model transitions to $\epsilon = 0$ slightly earlier, around 740 episodes. This suggests the adversarially-trained model adapts quicker, likely due to its design for adversarial perturbations and robust learning.

### 5.3.3 Loss

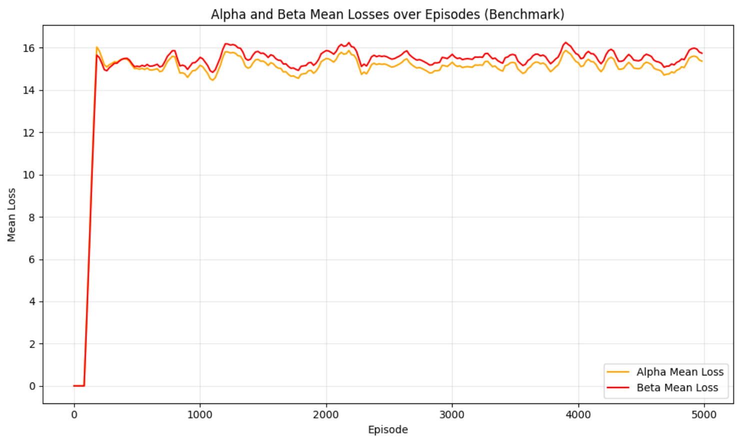

Figure 5: Mean Loss for the existing framework.

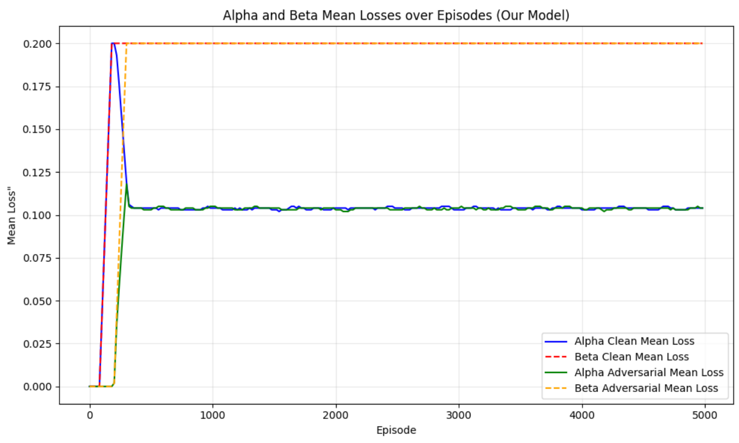

Figure 6: Mean Loss for our model.

In the benchmark model, Agent $\beta$'s mean loss stabilizes around 16, while Agent $\alpha$'s loss remains slightly below that, indicating that both agents are learning but with high residual loss levels. This points to suboptimal convergence, as the losses are consistent but not effectively minimized.

In the adversarially-trained model, during the burn-in phase (first 500 episodes), both agents exhibit a sharp initial spike in loss, reflecting the collection of diverse experiences. Once learning and perturbations begin, Agent $\alpha$'s clean mean loss (blue) and adversarial mean loss (green) stabilize around 0.1, demonstrating effective learning and robust performance under both clean and adversarial conditions. Meanwhile, Agent $\beta$'s clean mean loss and adversarial mean loss converge closely, stabilizing around 0.2, indicating limited improvement compared to $\alpha$.

### 5.3.4 Q-Values

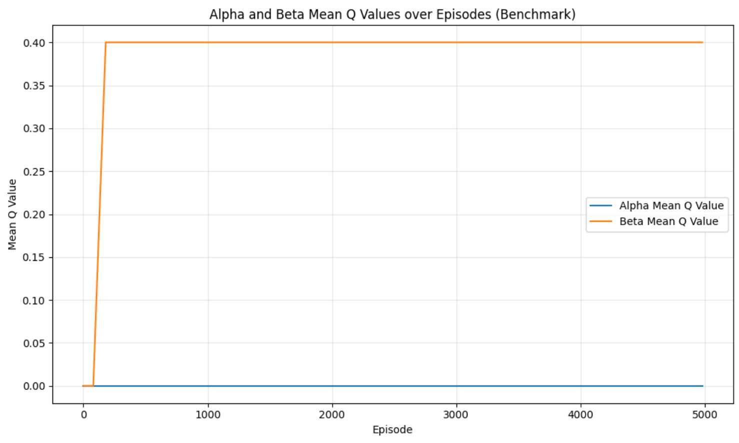

Figure 7: Mean Q-Values for the existing framework.

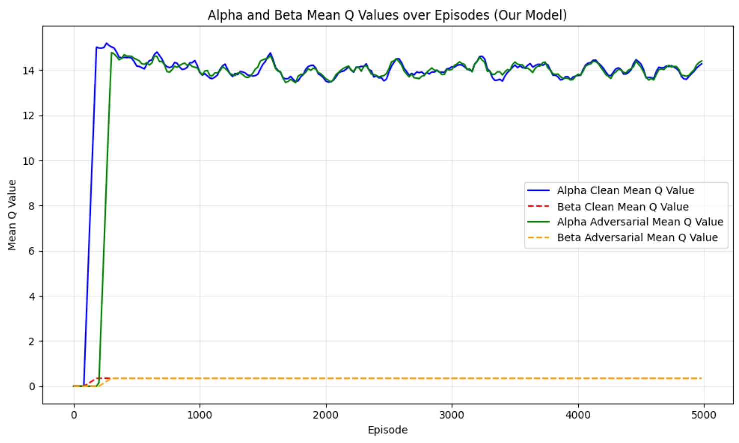

Figure 8: Mean Q-Values for our model.

In the baseline model, Agent $\beta$'s Q-values stabilize around 0.4, while Agent $\alpha$'s remain near zero, showing that $\beta$ learns marginally but $\alpha$ fails entirely to develop meaningful value estimates. In the adversarially-trained model, after learning begins, Agent $\alpha$'s clean Q-values spike sharply to $\sim$14, reflecting effective learning in clean conditions. In contrast, Agent $\beta$'s clean Q-values (red) plateau around 0.35, showing limited learning despite the clean environment. After adversarial training begins (post-episode 2000), clean Q-values remain stable: Agent $\alpha$ maintains values around $\sim$14, while Agent $\beta$ remains near $\sim$0.35, consistent with its earlier performance. For adversarial Q-values, Agent $\alpha$'s values closely track its clean values at $\sim$14, demonstrating strong robustness. However, Agent $\beta$'s adversarial Q-values (yellow) settle slightly above their clean values ($\sim$0.36). This reveals that adversarial training enables Agent $\alpha$ to maintain consistent value estimates across both clean and adversarial conditions, showcasing its robustness. However, Agent $\beta$ remains largely stagnant, with no significant improvement in learning or adaptability. This stagnation aligns with expectations since Agent $\beta$ was not perturbed during training.

## 6 Conclusion

This work explores the vulnerabilities of federated reinforcement learning-based recommendation systems when subjected to adversarial attacks. Our contributions include the development of an adversarial training framework leveraging FGSM to introduce perturbations and evaluate their impact on Q-value estimations and overall system performance. Our proposed approach shows that adversarial training improves robustness against malicious states but can also lead to significant trade-offs in reward optimization, especially under clean conditions. The findings reveal that adversarial training enhances robustness, with Agent Alpha maintaining consistent Q-value performance even in adversarial environments. However, this robustness comes at the cost of reduced overall reward performance, as the model tends to converge to suboptimal strategies in both adversarial and clean scenarios. Despite these challenges, the federated Q-value sharing mechanism effectively preserved privacy and facilitated knowledge transfer between agents.

Several limitations emerged during this study. The focus on a single adversarial attack method, FGSM, limits the generalizability of the findings to other types of attacks. Simplifications, such as omitting double Q-learning and security state tracking, while practical, may restrict the evaluation of broader defense strategies. Future work aims to address these challenges by incorporating additional defense mechanisms and diverse attack strategies, such as reward hacking or poisoning. Additionally, introducing differential privacy mechanisms entices us, as it could enhance data security while mitigating adversarial risks. Finally, efforts to counteract performance trade-offs would focus on improving reward optimization while maintaining adversarial resilience. By bridging the gaps in robustness, privacy preservation, and performance optimization, this research contributes to building trustworthy systems that can adapt to evolving adversarial threats while respecting user privacy.

## Acknowledgments

We would like to thank Xiaolin Sun, the course TA, for his invaluable help with understanding and applying adversarial attacks throughout this project.

## Code

The code used in this paper is available at https://github.com/rosenkor/CMPS6740Project. The repository includes the baseline implementation (Baseline_Model.ipynb) and our model (finalversion.ipynb).

## References

[1] Aqeel Anwar and Arijit Raychowdhury. Multi-task federated reinforcement learning with adversaries, 2021.

[2] Yongxin Deng, Xiaoyu Tan, Xihe Qiu, and Yaochu Jin. Fedslate:a federated deep reinforcement learning recommender system, 2024.

[3] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, Jim McFadden, Tushar Chandra, and Craig Boutilier. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology, 2019.

[4] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks, 2017.

[5] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations, 2021.