Auto.gov: Learning-based Governance for Decentralized Finance (DeFi)

Jiahua Xu, Yebo Feng, Daniel Perez, and Benjamin Livshits

***Abstract*—Decentralized finance (DeFi) is an integral component of the blockchain ecosystem, enabling a range of financial activities through smart-contract-based protocols. Traditional DeFi governance typically involves manual parameter adjustments by protocol teams or token holder votes, and is thus prone to human bias and financial risks, undermining the system's integrity and security. While existing efforts aim to establish more adaptive parameter adjustment schemes, there remains a need for a governance model that is both more efficient and resilient to significant market manipulations. In this paper, we introduce "Auto.gov", a learning-based governance framework that employs a deep Q-network (DQN) reinforcement learning (RL) strategy to perform semi-automated, data-driven parameter adjustments. We create a DeFi environment with an encoded action-state space akin to the Aave lending protocol for simulation and testing purposes, where Auto.gov has demonstrated the capability to retain funds that would have otherwise been lost to price oracle attacks. In tests with real-world data, Auto.gov outperforms the benchmark approaches by at least 14% and the static baseline model by tenfold, in terms of the preset performance metric—protocol profitability. Overall, the comprehensive evaluations confirm that Auto.gov is more efficient and effective than traditional governance methods, thereby enhancing the security, profitability, and ultimately, the sustainability of DeFi protocols.**

***Index Terms*—Governance, Decentralized Finance (DeFi), Reinforcement Learning (RL), Artificial Intelligence (AI).**

I. Introduction

GOVERNANCE is a crucial aspect of blockchain-based systems, designed to ensure their stability, integrity, and security [1]. Despite blockchains' ability to store transaction histories without relying on trusted third parties, certain decisions in blockchain systems necessitate an orchestration or *governance* processes. This requirement applies to both chains such as Bitcoin and Ethereum (sometimes called layer-1 chains) as well as higher-level developments, such as decentralized finance (DeFi) protocols [2]. DeFi protocols [3] (e.g. Aave, Uniswap, and Curve) facilitate borrowing, lending, and exchanging crypto-assets without a financial intermediary [4], but rather via *parameterized* algorithms. For instance, an Aave-like DeFi lending protocol—the focal protocol of this study—operates using various parameters, including interest rate model parameters as well as risk parameters such as *collateral factor* and *liquidation threshold*. These asset-specific risk parameters dictate the extent of overcollateralization and liquidation incentivization of each eligible cryptocurrency, which are vital in protecting the protocol from insolvency due to loan defaults or adverse market movements. As such, those parameters require ongoing monitoring and adjustments to ensure their suitability in the fast-changing crypto market.

Commonly, the adjustment of protocol parameters is determined through a governance process [5]–[7]. A proposal on such an adjustment is first posted on the protocol's forum, where the community forming the protocol's decentralized autonomous organization (DAO) voice their opinion. The decision of whether or not to adopt a particular proposal is generally subject to a vote by holders of the governance tokens of the corresponding protocol [8]. Voting can be done directly in the protocol's own governance forum, which typically employs single-choice voting, and/or on open platforms such as Snapshot (snapshot.org) and Tally (tally.xyz) where more sophisticated voting mechanisms such as weighted voting or quadratic voting can be supported [9].

a) Motivation: In recent years, DeFi governance forums have witnessed increased activity, particularly on risk-related matters, reflecting the community's growing recognition of the importance of DeFi governance (see Appendix A). Unfortunately, the protocol's core team often holds a significant portion of governance tokens [10] and assumes responsibility for post-voting execution, defeating the purpose of decentralized voting. In addition, the current governance process—comprising observing/ideation, proposing, discussion, voting, and eventual abortion/implementation—is labor-intensive and time-consuming. As an example, Figure 1 exhibits a governance proposal on the Aave (the protocol) governance forum suggesting to increase the *liquidation threshold* for the AAVE (the underlying token) market. The person justified their proposal by quoting the price performance of the AAVE token compared to a reference token LINK. The discussion of this thread lasted almost three weeks—from 3rd to 21st of January 2021, eventually resulting in the rejection of the proposal. This lack of agility inherent to the lengthy governance procedure is especially problematic in the rapidly evolving and volatile crypto market. As demonstrated in Appendix D, Aave risk parameters often remain unchanged even amid significant fluctuations in key market indicators such as price volatility, trading volume, and utilization ratio (borrow over supply).

The rigidity of the current governance model also leaves a protocol vulnerable to malicious exploitation such as price oracle attacks [12], where an attacker temporarily inflates the

Received December 2024; revised 12 February 2025; accepted 10 March 2025. (Corresponding author: Yebo Feng.)

Jiahua Xu is with Centre for Blockchain Technologies, University College London, UK, and Exponential Science. Email: jiahua.xu@ucl.ac.uk.

Yebo Feng is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. Email: yebo.feng@ntu.edu.sg.

Daniel Perez is with the Computer Science Department, Imperial College London, UK. Email: daniel.perez@imperial.ac.uk.

Benjamin Livshits is with Imperial College London, UK. Email: b.livshits@imperial.ac.uk.

Digital Object Identifier 10.1109/TSC.2025.3553700

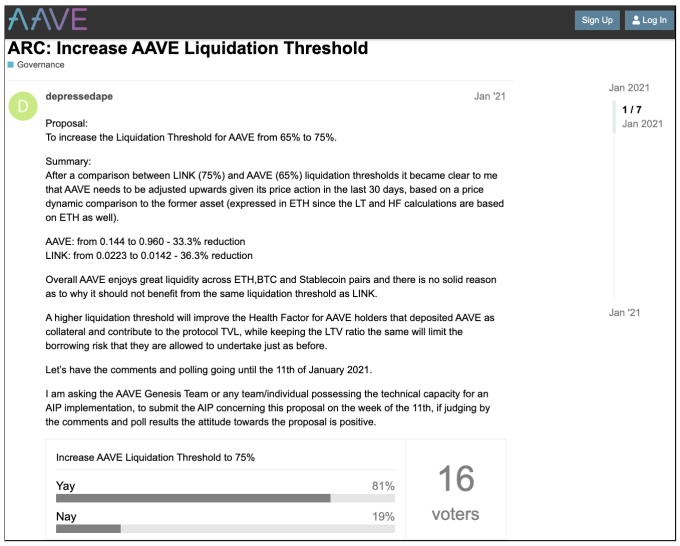


Figure 1: A proposal in Aave governance forum to increase the *liquidation threshold* of the AAVE token [11].

price of a collateralizing asset in order to borrow more than they should, with no intention of repayment. Lending protocols such as bZx Protocol and Cream Finance suffered losses of $8 million in 2020 and $130 million in 2021, respectively, due to price oracle exploits using "flash loans" [13].

Overall, the current de facto centralization of governance processes and their inability to promptly respond to dynamic market changes jeopardize the integrity of DeFi protocols and compromise their security. Despite recent efforts, such as [14], to establish a more adaptive DeFi parameter adjustment scheme, there exists no framework for a governance model that is both more efficient and effective in general, and resilient against major market manipulations, such as price oracle attacks, in particular.

The aforementioned challenges underscore the urgent need for more agile, automated, attack-resilient, and generalizable governance mechanisms. In this paper, we address this need by introducing Auto.gov, a parameter adjustment framework based on deep Q-network (DQN) reinforcement learning (RL). Specifically, using the lending protocol mechanism for the environment setup, we apply DQN RL to train a governance agent to determine the optimal adjustment policy for lending pool assets' respective *collateral factors*, under scenarios both with and without price oracle attacks. Established as a proof-of-concept using an abstract DeFi environment, our approach lays the groundwork for more concrete implementations.

Equipped with RL, Auto.gov is well-suited for this type of automation for several reasons. Firstly, the DeFi environment can be modeled as a Markov Decision Process (MDP), where future states are determined solely by current states and actions, aligning well with RL principles. Secondly, unlike many other machine learning models that rely on static datasets and require retraining to adapt, RL excels at rapidly responding to the complex and unpredictable dynamics inherent in DeFi systems, providing a governance approach that is robust, resilient, and capable of evolving alongside shifting market conditions. Thirdly, Auto.gov takes into account the long-term consequences of decisions, a critical aspect for DeFi governance where decisions can significantly impact protocol stability, user trust, and risk tolerance. Lastly, the design of Auto.gov reduces the potential for human error and mitigates the risk of "black-swan" events through the prevention of erratic system behavior as only gradual adjustments at each step are allowed.

b) Contributions: Our contributions are summarized below[1]:

- To the best of our knowledge, this paper represents the first attempt to integrate DeFi governance with artificial intelligence (AI).
- We formalize and parameterize an Aave-like DeFi environment that models the interaction between the protocol, user, and external market; the stylized environment can be easily modified to apply to different types of DeFi protocols and to accommodate diverse user behaviors.
- We quantitatively demonstrate Auto.gov's superiority in both *efficacy* and *efficiency*. Auto.gov exhibits particular effectiveness in countering oracle attacks: in a simulated training environment, it boosts the DeFi protocol's final profit by over 60% compared to the baseline governance agent; in real-world testing, it outperforms the best benchmark approach by 14% and the static baseline approach tenfold. Besides, Auto.gov adapts swiftly to DeFi environments, making robust, profitable decisions in about 4 hours of training using only a laptop, several orders of magnitude faster than the week-long voting-based governance process.

II. Related work

a) DeFi modeling and management: Our work is inspired by existing literature on modeling the behavior and risks associated with DeFi, as well as effective protocol management.

Kao et al. [15] employed agent-based modeling to conduct a stress test on the market risks faced by participants of the Compound protocol. They recommend that the community utilize their simulation-based assessment to evaluate any new assets proposed for introduction to the protocol; Bartoletti et al. [16] formalized the mechanisms of DeFi lending protocols and describe the over- and under-collateralization risks associated with those protocols; Perez et al. [17] depicted an anecdotal liquidation event on Compound and analyze its root cause. The three aforementioned works concentrate on modeling and analyzing different aspects of DeFi protocols. In contrast, our approach directly formulates governance measures aimed at improving the overall safety and profitability of the protocol.

In 2024, Bastankhah et al. [14] introduced an adaptive, data-driven protocol for DeFi borrowing and lending, which is closely related to ours. This protocol features a high-frequency controller that dynamically adjusts interest rates to ensure market stability and competitiveness with external markets. However, there are three key differences between their work and ours: (i) Their primary goal is to develop a DeFi protocol focusing on efficiency, whereas our focus is

[1]The code is open-sourced at https://github.com/xujiahuayz/auto-gov.

on revenue maximization; (ii) They design an entirely new DeFi protocol, while our work involves a DeFi governance agent that can be integrated into existing DeFi protocols for better governance; (iii) In terms of methodology, we employ AI, whereas their approach relies on optimization techniques.

b) Machine learning for DeFi: Our work is connected to the literature on machine learning for DeFi. Babu et al. [18] proposed a decentralized interest rate swap platform using machine learning algorithms, like LSTM and SVM, to predict interest rate volatility. Palaiokrassas et al. [19] investigated the application of machine learning (e.g., logistic regression, random forest, XGBoost, CatBoost, LightGBM, and CNN) for credit risk assessment in Multichain DeFi. John et al. [20] proposed a swarm-learning-based credit scoring approach for P2P lending protocols. Jiang et al. [21] applied a deep RL framework to achieve a model-free portfolio optimization with crypto-assets. Hou et al. [22] used deep RL to automate attack analysis on blockchain incentive mechanisms. These works focus on predicting, optimizing, or evaluating specific elements of DeFi with machine learning, offering recommendations to users as a result. However, unlike Auto.gov, none of them are directly engaged in the governance of DeFi.

c) Related work in traditional finance (TradFi): Our research also intersects with the body of literature on algorithms and automation applied in the broader (traditional) finance field. Algorithmic trading in particular has been intensively studied in both finance and computer science literature [23]–[25]. Cao [26] surveyed AI and data science (AIDS) techniques, as well as their challenges and opportunities in financial businesses. Huang et al. [27] devised a two-step machine learning algorithm—the Supervised Adaptive Group LASSO (SAGLasso)—to construct return predictors for TradFi instruments such as government bonds. Spiegeleer et al. [28] demonstrated the usage of machine learning to achieve fast pricing, hedging, and fitting for financial derivatives. While all these studies apply machine learning to various areas of finance, our research is distinctly focused on DeFi governance.

III. Environment and Agent Modeling

For our RL experiments, we establish a simplified DeFi environment that consists of users and an Aave-like lending protocol. The RL agent seeks to optimize the protocol's *net position* by adjusting each lending pool's *collateral factor*. Figure 2 illustrates the complete action-state space of our RL environment, further elaborated in Section III-A and Section III-B.

A. Actions

Actions, denoted as a, refer to decisions made by a related party in response to the current state of the environment.

1) Governance agent actions:

a) Raise / lower / keep collateral factor: When the governance agent raises, lowers, or keeps the existing collateral factor of a specific lending pool, the pool's collateral factor increases, decreases, or remains the same, respectively. In our pursuit to develop an automated governance agent that dynamically adjusts policies using machine learning, the quality of

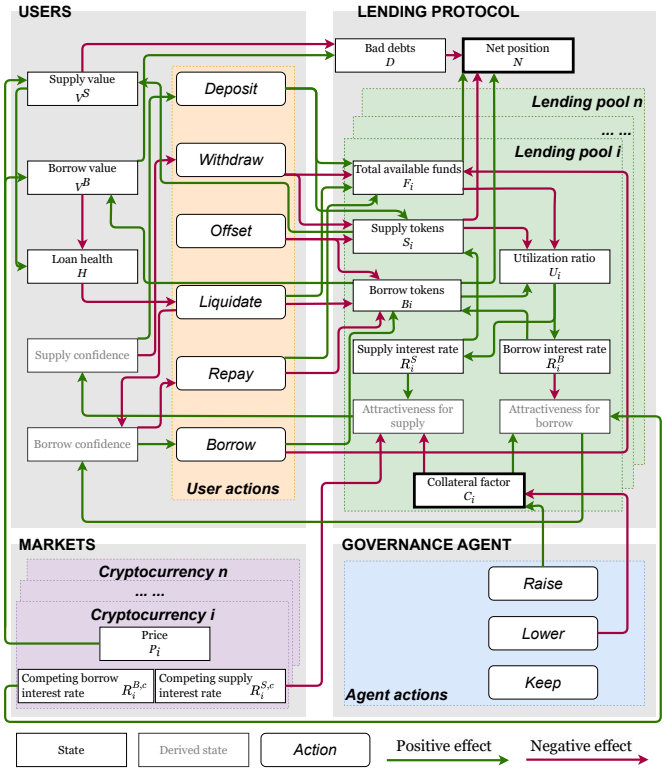


Figure 2: Action-state space encompassing users, lending pools and external markets of individual cryptocurrencies, and governance agent.

the agent's decisions becomes crucial for the proper operation of the lending protocol. However, the inherent lack of transparency and control in machine learning models may introduce additional risks and uncertainties to the system. Therefore, to guarantee the certainty, controllability, and security of the actions made by the governance agent, we have implemented an additional layer of constraints on the agent's actions. First of all, we check the validity of agent actions (e.g., collateral factors must never exceed 1 after adjustment) so that only reasonable actions are applied to the protocol. Besides, our implementation caps the incremental increase or decrease at 2.5%, meaning that even if the governance agent makes poor decisions, any potential harm to the DeFi protocol would only be gradual and minimal, allowing for easy detection and correction by the community.

2) User actions:

a) Deposit: When a user deposits into a specific lending pool (i.e., depositing ETH to the ETH pool, USDC to the USDC pool, and so on), the pool's *total available funds* increase by the deposited quantity. Simultaneously, the same amount of interest-accruing *supply tokens* are minted to their address, tracking the accrual of interest at the *supply interest rate* and reflecting the increase in the user's *supply value*.

b) Withdraw: When a user withdraws from a lending pool, its *total available funds* decrease by the withdrawn amount, and the same quantity of interest-accruing *supply tokens* are burned. The quantity withdrawn cannot exceed the user's existing balance of *supply tokens*, and, after the withdrawal, the user's *loan health H* (see Section III-B2) must

not drop below 1.

c) Borrow: When a user borrows from a specific lending pool, the pool's *total available funds* decrease by the borrowed quantity. Simultaneously, the same quantity of interest-accruing *borrow tokens* are minted, tracking the accrual of interest at the *borrow interest rate* and reflecting the increase in the user's *borrow value*. The maximum borrowable amount is such that after the borrow, the user's *loan health* H must be at least 1.

d) Repay: When a user repays, the pool's *total available funds* increase by the repaid amount, and the same quantity of interest-accruing *borrow tokens* are burned.

e) Liquidate: In our stylized training environment (see Configuration i) in Section V-A), liquidating a borrow position—on an aggregate level—updates the pool states (*total available funds* and interest-accruing *borrow tokens*) just like a repayment, but it also lowers the user's *borrow confidence* (see Section III-B2).

f) Offset: Instead of repaying with the underlying asset, a user can also reduce the borrow position by burning the same quantity of the asset's *supply tokens* and *borrow tokens*. The action is equivalent to *withdrawing* and *repaying* combined.

B. States

The states, collectively denoted as $\mathbb{S}$, represent the current situations or configurations of an environment that an agent interacts with. They consist of lending pool states, user states, and market states.

1) Lending pool states:

a) Total available funds F_i: Total available funds of lending pool i is the amount of its underlying token i remaining in the pool available to be withdrawn or borrowed. From the accounting perspective, total available funds are on the asset side—equivalent to cash—of the pool's balance sheet, and hence positively contribute to the pool's *net position* (Equation 1).

b) Supply tokens S_i: Supply tokens are interest-accruing tokens (such as Aave's aTokens) that keep track of the amount of funds a lending pool owes to the user over time. From the accounting perspective, supply tokens, bearing the nature of payables, are on the liability side of the pool's balance sheet, and reduce the pool's *net position* (Equation 1) [29]. The number of supply tokens also negatively influences the *utilization ratio* of the pool. A user's *supply value* within a pool equals the value of a user's allocated supply tokens within that pool times the current price of the underlying token.

c) Borrow tokens B_i: Borrow tokens are interest-accruing tokens that keep track of the amount of funds the user owes to the lending pool over time. From the accounting perspective, *borrow tokens*, bearing the nature of receivables, are on the asset side of the pool's balance sheet, and hence enhance the pool's *net position* (Equation 1). The number of *borrow tokens* also positively influences the *utilization ratio* of the pool (see Equation 2 below). A user's *borrow value* within a pool equals the value of a user's allocated borrow tokens within that pool times the current price of the underlying token.

d) Bad debts D: While the lending protocol is generally secured by its overcollateralization and liquidation mechanism, under extreme market conditions, e.g. when the price of a collateral asset experiences a sudden drop, the protocol may still suffer from bad debts. Bad debts are the amount of funds that the lending protocol is unable to recover from the user; they occur when a user's loan position is undercollateralized—i.e. when a user's *supply value* does not suffice to cover their *borrow value*, leaving the user with no incentive to repay the loan and other network participants with no incentive to liquidate the position (see also Section V-C2). From the accounting perspective, bad debts are deemed expenses, and hence negatively contribute to the pool's *net position* (Equation 1).

e) Net position N: The net position of the lending protocol is calculated as the difference between the protocol's *total reserve* and *bad debts*, specifically:

$$N = \Sigma_i[P_i \cdot \underbrace{(F_i + B_i - S_i)}_{=W_i}] - D \ , \tag{1}$$

where F_i denotes lending pool i's total available funds (Assets: Cash), B_i the borrow tokens (Assets: Receivables), S_i the supply tokens (Liabilities: Payables), all denominated in token unit, and P_i denotes the price of token i. Reserve W_i reflects lending pool i's net worth under the assumption that all outstanding loans will be paid back; the lending protocol is *insolvent* when $\Sigma_i(P_i \cdot W_i) < D$, a state we term "bankruptcy".

f) Utilization ratio U_i: The utilization ratio of lending pool i is the ratio of the number of *borrow tokens* to the number of *supply tokens*, formally

$$U_i = \tfrac{B_i}{S_i}. \tag{2}$$

A low utilization ratio indicates capital inefficiency, while a high utilization ratio signals a high liquidity risk of the pool. The utilization ratio is therefore used to determine the pool's supply and borrow interest rates through a pre-defined interest rate model, a monotonically increasing function of the utilization ratio. When the utilization ratio is low, both the supply and borrow interest rates are low, in order to discourage supply and encourage borrow to drive the utilization ratio higher; when the utilization ratio becomes too high, both the supply and borrow interest rates increase exponentially, in order to discourage borrow and encourage supply to drive the utilization ratio lower and to mitigate liquidity risk.

g) Collateral factor C_i: The collateral factor $C_i \in [0, 1)$ of a specific lending pool i determines the fraction of a user's *supply value* in pool i counted towards the user's total borrowable value. In other words, the user's total borrowable value is given by $\sum_i (V_i^S \cdot C_i)$. As a numerical example, consider $C_1 = 0.2$, $C_2 = 0.8$, $V_1^S = \$100$, and $V_2^S = \$50$; that is, the collateral factor for lending pool 1 is 0.2 and for pool 2 is 0.8, and the user has a *supply value* of \$100 in pool 1 and \$50 in pool 2. The user can therefore borrow up to \$60 ($= \$100 \times 0.2 + \$50 \times 0.8$) in total across all lending pools.

Conventionally, the collateral factor associated with each lending pool is manually adjusted by the protocol team based on the underlying token's risk profile such as price volatility and level of centralization. The riskier an asset is deemed,

the lower the collateral factor is set to be. A higher collateral factor means a higher borrowing capacity, and hence a higher liquidity risk. Therefore, the collateral factor positively influences a pool's *attractiveness for borrow*, and negatively influences its *attractiveness for supply*.

h) Supply interest rate R_i^S: The supply interest rate of lending pool i is the interest rate that the lending pool pays to the user for supplying funds. A higher supply interest rate means a higher proliferating speed for supply tokens due to interest accrual. The supply interest rate is set such that, at each step, the supply interest to be accrued can be fully covered by the borrow interest to be accrued with some spread $\varsigma \in (0, 1)$, i.e. $R_i^S = R_i^B \cdot U_i \cdot (1 - \varsigma)$.

i) Borrow interest rate R_i^B: The borrow interest rate of the lending pool i is the interest rate that the user pays to the lending pool for borrowing funds. A higher borrow interest rate means a higher increasing speed for borrowing tokens due to interest accrual. The borrow interest rate is set according to a pre-defined interest rate model, which in our environment, is set to be $R_i^B = \frac{1}{b \cdot (1 - U_i)}$, where $b > 0$.

j) Attractiveness for supply: The attractiveness for the supply of a lending pool is a derived state designed to model how a pool's states influence the user's willingness to supply funds to the entire lending protocol. A user's *supply confidence* in the lending protocol is positively affected by the aggregate attractiveness for the supply of all lending pools in the protocol (see Figure 2).

k) Attractiveness for borrow: The attractiveness for borrow of a lending pool is a derived state designed to model how a pool's states influence the user's willingness to borrow funds from the entire lending protocol. A user's *borrow confidence* in the lending protocol is positively affected by the aggregate attractiveness for the borrow of all lending pools in the protocol.

2) User states:

a) Supply value V^S: The supply value of a user is the sum of their supply value across all lending pools: $V^S = \sum_i V_i^S = \sum_i (S_i \cdot P_i)$, where S_i is the quantity of the user's token i supply quantity and P_i the price of token i. All other things equal, a higher supply value decreases the user's loan-to-value ratio $\frac{V^B}{V^S}$, thus increasing the user's *loan health*.

b) Borrow value V^B: The borrow value of a user is the sum of their borrow value across all lending pools: $V^B = \sum_i V_i^B = \sum_i (B_i \cdot P_i)$, where B_i is the quantity of the user's token i borrow quantity. All other things equal, a higher borrow value increases the user's loan-to-value ratio $\frac{V^B}{V^S}$, and hence decreases the user's *loan health*.

c) Loan health H: The loan health of a user is the ratio between their aggregate loanable value, which equals the sum of the collateralizing asset's *supply value* respectively discounted with each asset's *collateral factor* (see numerical example under Section III-B1), and their aggregate *borrow value*: $H = \frac{\sum_i (V_i^S \cdot C_i)}{V^B}$. $H \geq 1$ signifies a healthy loan position, and $H < 1$ triggers liquidation.

d) Supply confidence: The supply confidence of a user is a derived state designed to model their willingness to supply funds to the lending protocol. Numerically speaking in our environment, supply confidence is measured by *supply buffer* $\delta^S \in [0, 1]$, which is the amount of funds that a user withholds as a fraction of their total funds willing to be supplied to this specific lending protocol (as opposed to competing protocols). High supply confidence—or low supply buffer—encourages the supply action, while low supply confidence—or high supply buffer—discourages it (see Section V-C3 and Section V-D3).

e) Borrow confidence: The borrow confidence of a user is a derived state designed to model their willingness to borrow funds from the lending protocol. Similar to supply confidence, borrow confidence is measured by *borrow buffer* $\delta^B \in [0, 1]$. High borrow confidence—or low borrow buffer—encourages the borrow action, while low borrow confidence—or high borrow buffer—has the opposite effect.

3) Market states:

a) Price P_i: The price of an underlying token of a lending pool i positively affects both the supply value and the borrowed value of a user within that pool. Token price movements follow a stochastic process (see Section V-D2 for simulated price time series).

b) Competing supply interest rate $R_i^{S,c}$: The competing supply interest rate of token i is offered by the external market, representing the opportunity cost of supplying funds to the lending protocol. A higher competing supply interest rate reduces the *attractiveness for supply* of the lending pool i, and encourages the user to withdraw funds from the protocol (to supply to a competing protocol) (see Figure 5 under Section V-C).

c) Competing borrow interest rate $R_i^{B,c}$: The competing borrow interest rate of token i is offered by the external market. A higher competing borrow interest rate increases the *attractiveness for borrow* of the lending pool i, while a low one encourages the user to repay their loan (and to instead borrow from a competing protocol).

C. Reward

We approximate the protocol's profitability using the change in the *net position* N after each incremental step: $\Delta N_t = N_t - N_{t-1}$. The governance agent acts in the interests of the lending protocol, whose ultimate objective is profit maximization.[2] To evaluate the agent's performance, we compare ΔN_t in the RL environment against the baseline environment (see Section V-D1). The difference quantifies the extent to which the governance agent outperforms the baseline agent. The resulting performance metric forms the reward for the governance agent, with an additional penalty applied in the case of an invalid action (see Section III-A1). Specifically, the reward is given by:

$$r_t = \Delta N_t^{\text{RL}} - \Delta N_t^{\text{baseline}} - \text{Penalty} \cdot \mathbf{1}_{\text{invalid action}}, \quad (3)$$

[2]While alternative optimization objectives (e.g. the number of users, the average duration of deposits) could be considered and the design can be somewhat arbitrary, the rationale behind our choice of *net position* maximization as the objective is because the protocol ultimately belongs to its governance tokenholders. These individuals are financially incentivized by the price appreciation of governance tokens, whose fundamental value is directly tied to the protocol's profitability [30], [31].

where $\mathbf{1}_{\text{invalid action}}$ is an indicator function that equals 1 if the action is invalid and 0 otherwise. This reward serves as feedback, informing the agent about the immediate consequence of its action in the given state. The discounted cumulative reward, in turn, reflects the agent's long-term optimization objective (see Section IV-A).

D. Transition function

The transition function is a key element representing the environment's response to the actions taken by the governance agent, seamlessly integrated into our modeled DeFi environment. Each action performed by the governance agent prompts a corresponding change in the environment, resulting in the generation of respective state $\mathbb{S}$.

IV. Design of the Governance Agent

In this section, we present the design and training process of the governance agent, which is responsible for making appropriate decisions for DeFi protocols.

A. Reinforcement learning-based agent

We employ reinforcement learning (RL), an artificial intelligence approach for training agents to make optimal decisions in complex and uncertain environments (i.e., DeFi ecosystems) in the presence of attacks. Specifically, we use a deep RL agent [32] to serve as the governance agent, which dynamically selects the best action based on the current state of the DeFi protocols. This approach is well-suited for our purpose, as it enables the agent to learn from past experience and adapt its behavior to changing conditions in real time. By leveraging a RL agent, we can ensure that our protocol is flexible, adaptable, resilient to attacks, and able to optimize its behavior to maximize its performance under various scenarios.

Figure 3 illustrates the architecture of the proposed approach. Our goal is to train a policy π for the governance agent to maximize the discounted, cumulative reward $\mathbb{R}_0 = \sum_{t=0}^{\infty} \gamma^t r_t$, where γ denotes the discount factor. We leverage deep Q-learning [33] to conduct the learning process for the governance agent, where a function $Q : \mathbb{S} \times a \rightarrow r$ can tell the agent what the reward would be. Then, the policy π of the governance agent can be represented as Equation 4, which always selects the action a that can maximize the value of the Q function. In this equation, $\mathbb{S}_t$ denotes the state $\mathbb{S}$ at time t.

$$\pi(\mathbb{S}_t) = \arg\max_{a \in \mathbf{a}} Q(\mathbb{S}_t, a) \tag{4}$$

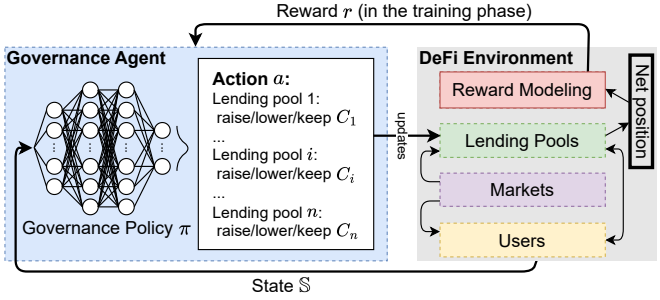


Figure 3: Architecture of Auto.gov. The detailed DeFi environment is illustrated in Figure 2.

Algorithm 1 High-level training procedure for the RL-based governance agent.

Initialize DQN Q
Initialize target network $\hat{Q}$
Initialize experience replay memory D
while not converged **do**
  $\epsilon \leftarrow$ new epsilon with ϵ-decay ▷ Sampling
  Choose an action a from $\mathbb{S}$ using policy ϵ-greedy(Q)
  Agent takes a, observe reward r and the next state $\mathbb{S}'$
  Store transition $(\mathbb{S}, a, r, \mathbb{S}', is\_done)$ in D
  if enough experience in D **then** ▷ Learning
    Sample a *batch* of transitions from D
    for every transition $(\mathbb{S}_i, a_i, r_i, \mathbb{S}'_i, is\_done_i)$ in *batch* **do**
      if is_done_i **then**
        $y_i \leftarrow r_i$
      else
        $y_i \leftarrow r_i + \gamma \max_{a' \in \mathbf{a}} \hat{Q}(\mathbb{S}'_i, a')$
      end if
      $\delta \leftarrow \frac{1}{N} \sum_{i=0}^{N-1} (Q(\mathbb{S}_i, a_i) - y_i)^2$ ▷ Error
      $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$ ▷ Adam optimization
      while Q_t not converged **do**
        $t \leftarrow t + 1$
        $g_t \leftarrow \nabla_Q \delta_{t-1}$
        $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
        $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
        $\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$ ▷ Correcting bias
        $\hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$
        $Q_t \leftarrow \prod_{\mathcal{F}, \sqrt{\hat{v}_t}} (Q_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \mu})$ ▷ Updating
      end while
    end for
    Synchronize $\hat{Q}$ with Q if reaches threshold
  end if
end while

For each training update, we enforce that every Q function for the policy π follows the Bellman equation (Equation 5), where the difference between the two sides of the equality is known as the temporal difference error δ (Equation 6). Here, $\mathbb{S}'$ represents the state that the environment transitions to after the agent takes an action from the current state $\mathbb{S}$.

$$Q^{\pi}(\mathbb{S}, a) = r + \gamma Q^{\pi}(\mathbb{S}', \pi(\mathbb{S}')) \tag{5}$$

$$\delta = Q(\mathbb{S}, a) - (r + \gamma \max_{a'} Q(\mathbb{S}', a')) \tag{6}$$

Additionally, considering the potentially vast state space and the expected growth with the increasing number of lending pools, we employ a DQN [34] to serve as the policy network for the governance agent, enabling it to monitor multiple pools simultaneously. In a DQN, a deep neural network acts as an approximator for the Q-function [35]. The DQN combines the strengths of both deep learning and Q-learning to tackle complex reinforcement learning problems with large state spaces, providing an efficient and robust approach for learning optimal policies.

Furthermore, to expedite the process of error elimination, we utilize Adam optimization [36] during training. Adam is an optimization algorithm that is commonly used to train deep neural networks. It is a variant of stochastic gradient descent (SGD) that adapts the learning rate for each weight in the network based on historical gradient information. Other optimization algorithms such as Gradient Descent and traditional SGD were attempted and Adam outperformed them all.

Algorithm 1 formalizes the training procedures including sampling, Adam optimization, and neural network updating,

where g_t denotes the gradient at step t, m_t is the exponential moving average (EMA) of g_t, v_t is the EMA of g_t^2, β_1 and β_2 are hyperparameters used in the moving averages of g_t^2 and g_t, α denotes the learning rate, and μ is a small number to prevent the denominator from becoming 0. Additionally, ϵ-greedy(Q) is a method to balance exploration and exploitation, where ϵ refers to the probability of choosing to explore as opposed to exploiting. During the training, the input data is the states and rewards that come from the simulated market environment.

B. Target network

To prevent overfitting during the training process and enhance the agent's resilience to potential attacks, we have implemented a target network [32]. The target network is a critical component in the training process of DQNs. It consists of a separate neural network with the same architecture as the primary or online network, but with a distinct set of weights. The primary purpose of the target network is to provide more stable and consistent target values for the Q-learning updates. In our approach, the training procedure involves periodically copying the weights from the online network to the separate target network, as illustrated in Figure 4 and Algorithm 1. The target network aids the governance agent in enhancing stability and convergence, ultimately resulting in more reliable and robust learned policies.

However, simply employing the target network may bring side effects, such as conservative Q-value estimations, temporary performance drops, and slower training. To address the disadvantages associated with the target network, we propose a hybrid approach that employs the primary network and the target network during different phases of the agent training. This new method results in faster training and better agent performance. We provide more details regarding this hybrid approach in Appendix B.

C. Prioritized experience replay

To equip the governance agent with the ability to adapt to diverse DeFi environments, expedite the policy network's training, and swiftly enhance its resistance to potential attacks, we also have incorporated prioritized experience replay into our RL agent as an optional training feature.

Prioritized experience replay [37] enhances the standard experience replay technique, which uniformly stores and samples experiences during training. While experience replay helps break the correlation between consecutive samples and decrease update variance, prioritized experience replay assigns significance to each experience based on the discrepancy between actual and predicted Q-values. This approach increases the likelihood of sampling and learning from unexpected experiences or those with greater potential to boost the agent's performance.

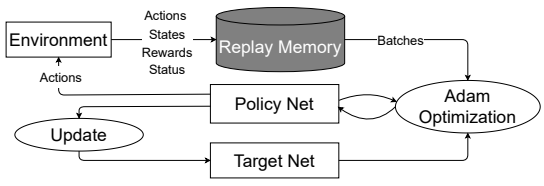


Figure 4: Operations for training the governance agent with the target network.

Once the DQN Q is fully converged and the governance agent is properly trained, the governance agent can be used for setting parameters for the lending pool individually, even without the reward feedback from the DeFi environment.

V. Simulation setup

A. Environment abstraction

For ease of experimentation, we abstract the DeFi lending protocol environment with the following configurations without loss of generality:

i) Following the common practice in macroeconomics [38], we model market user actions on an aggregate level, i.e. we have one representative agent who behaves as the entire user base collectively does.
ii) There is no liquidation incentive, as it is a zero-sum game among users (the liquidator wins and the borrower loses), and on an aggregate level, the inclusion or exclusion of liquidation incentive does not directly affect our optimization goal: maximizing the protocol's *net position* N (see Equation 1).
iii) We equate *liquidation threshold* with *collateral factor*.

B. Lending pool establishment

We experiment with how asset price volatilities might affect the RL agent's collateral factor determination. Two assets are insufficient to model one stable asset and one volatile asset; if Asset A is volatile against B, then B must be volatile against A. Thus, we need at least three assets to demonstrate the volatility effect on the collateral factor in a self-contained modeling environment. To this end, we include in the lending protocol environment three lending pools covering all three types of crypto-assets: ETH—the numeraire or the denominating asset of the protocol, USDC—a USD-pegged stablecoin, and TKN—some arbitrary token.

C. User reaction simulation

Figure 5 illustrates the series of user's reactions, preprogrammed in our model environment, according to the market condition as well as the user's own financial status.

1) Attack scenario: In each episode, the environment randomly generates 3 steps where a price oracle attack will be performed before other ordinary actions from the market user.

While Auto.gov is agnostic about *how* exactly the market manipulation occurs, a typical way to achieve an arbitrary price hike is through a **flash-loan-funded price oracle attack** [39]–[41] that involves:

i) driving the price of a crypto-asset high temporarily by buying a sizable amount of that asset with funds (usually in ETH or stablecoin) borrowed through a flash loan which requires no collateralization;
ii) depositing this asset at the inflated price;
iii) borrowing out funds worth more than the true value of the deposited asset;

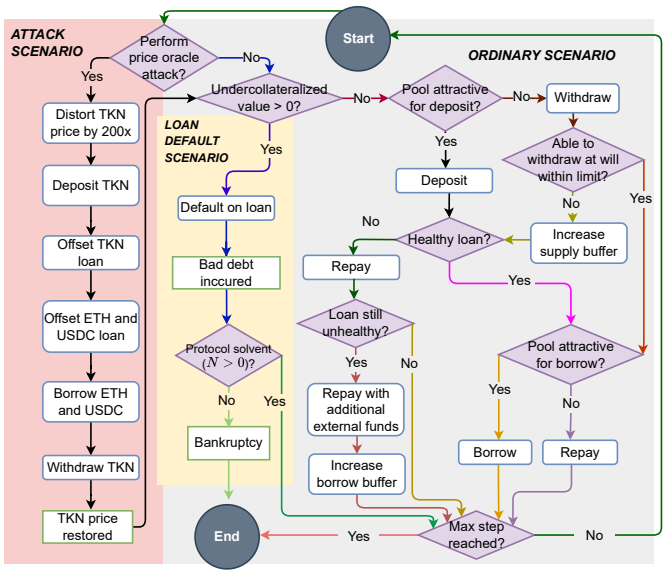


Figure 5: Series of user reactions at each step (from "Start"-node to "Max step reached?"-node) of an episode under attack (Section V-C1), loan default (Section V-C2), and ordinary (Section V-C3) scenarios, looping until episode ends ("End"-node).

iv) using part of the borrowed funds to repay the flashloan and profiting from the excess.

Crypto-assets with a thinner market are generally more vulnerable to price manipulation; more widely-adopted assets such as ETH or USD-pegged stablecoins often get drained from the lending protocol under the attack described above.

At the step where an attack shall occur, the price of TKN first becomes inflated by 200 times. Then the market user deposits all their available TKN to the lending pool to increase their supply token balance in TKN. Next, the user offsets their TKN loan with TKN supply tokens as much as possible, such that all the remaining TKN supply tokens can be used to back other, more valuable loans. The user subsequently offsets their ETH and USDC loans with the corresponding supply tokens as much as possible, such that the remaining ETH and USDC loans will be mainly backed by TKN at an inflated price. The user then seeks to borrow out as much additional ETH and USDC as possible from the lending pools, possibly depleting the pools before the faked borrow quota is used up. Finally, the user withdraws TKN to restore their initial TKN balance as much as possible.

2) Loan default scenario: A user is likely to default on their loan if their undercollateralized value—defined as their total *borrow value* less total *supply value*—is positive, leaving them with no incentive to repay the loan for collateral rescue. As intended by the attacker, loan default is most likely to occur following a price oracle attack, although it may also happen—albeit much less likely—absent malicious behavior when the market simply is too volatile. When a loan defaults, the undercollateralized value will be treated as *bad debt* expenses, reducing the net position of the protocol (see Equation 1). Under major attacks when the attacker manages to borrow out a relatively large amount of ETH and USDC, the bad debt expenses can suffice to cause the protocol to bankrupt, i.e. having $N \leq 0$, a non-positive *net position* (see Section III-B1), which ends the episode prematurely.

3) Ordinary scenario: In an ordinary scenario, the user deposits (withdraws) based on how attractive (unattractive) the lending pool is for depositing, which then depends on how much higher (lower) the lending interest rate and the collateral factor of the underlying asset of the lending pool is compared to the external market level. If the user decides that the lending pool is relatively unattractive, they will seek to withdraw liquidity. If they are not able to withdraw within the limit at will—e.g., when their deposited funds are lent out by the protocol, leaving the pool with insufficient liquidity for large withdrawal—they will withdraw as much as possible but also increase their supply buffer, meaning they will supply less liquidity in the future steps for precaution even when the lending pool is attractive for depositing.

The user subsequently checks their loan health, i.e. whether their loan value has exceeded the aggregate loanable value backed by the collateral. If healthy, the user decides whether to borrow more or repay depending on how much more or less the lending pool's borrow interest is compared to the market level. If unhealthy, the user tries to repay and restore their loan health as much as they can in the first instance. If still unhealthy, additional external funds will be injected into the user's account to repay the loan. This is to mimic the effect of liquidation, where the liquidator repays the loan on behalf of the borrower and seizes the latter's collateral. Note that as discussed in Section V-A, we only model one market user to represent the aggregate user behavior's effect on the protocol. In that sense, a loan turning liquidatable can in fact directly increase the liquidity of the lending pool. On the flip side, however, the user loses some *borrow confidence* and consequently increases their *borrow buffer* every time their collateral becomes liquidatable, meaning they will borrow less in future steps for precaution even when the lending pool is attractive for borrowing. In addition, the newly injected liquidity is still subject to withdrawal in future steps should the lending protocol exhibit unfavorable conditions for depositing.

D. Additional setups for the DeFi lending environment

1) Baseline environment: The baseline environment will be initiated exactly like the main training environment, including the price trajectories of all tokens, as well as the steps at which an attack would occur. The only difference is that there is no governance agent in the baseline environment, and the collateral factors will remain at their initial value of 0.8 for all three tokens throughout each episode.

2) Asset price simulation: The initial prices—all denominated in ETH—of TKN, USDC, and ETH are all normalized to 1. Following the common practice in econometrics and financial stochastics [42], we model the token price time series P_t as a geometric Brownian motion with μ_t and σ_t that can be time-variant but are piece-wise constant within every step. The price update at each step can be formally expressed as:

$$P_t = e^{(\mu_t - \frac{\sigma_t^2}{2})t + \sigma_t w_t} , \tag{7}$$

Table I: Key hyperparameters of the RL agent and its DQN.

Total number of layers	5	Input neurons	10×pool_num
DQN hidden layers	3	Number of training episodes	7,301
Neurons in hidden layer 1	256	Target net switch on point	0.3
Neurons in hidden layer 2	256	Learning rate (α)	0.001
Neurons in hidden layer 3	256	Gamma (γ)	0.5
Epsilon (ϵ) start	1	Batch size (with attack)	32
Epsilon (ϵ) end	2e-3	Batch size (without attack)	256
Epsilon (ϵ) decay	5e-7		

where $w_t \sim \mathbb{N}(0,1)$.

As a numeraire, ETH has by definition $\mu_t \equiv 0, \sigma \equiv 0$, i.e. $P_t \equiv 1$. For USDC, we set $\mu_t \equiv 10^{-4}, \sigma_t \equiv 0.05$. For contrast, we set TKN to be most volatile, with $\mu_t \equiv 10^{-5}$ and $\sigma_t = 0.05 + \frac{(t-200)^2}{5\times10^5}$. Clearly, we attempt to synthesize TKN price such that it becomes gradually less volatile before day 200, and then increasingly more volatile. Note that the simulated dummy price time series are only used for model training purposes; we use real-life price data for the final test (see Section VI-D).

3) Other DeFi environment parameters: We set the initial collateral factor for all three assets to be 0.8. We assume the external competing supply and borrow interest rates to be 0.05 and 0.15 per annum, respectively, for all three tokens. We additionally set the external competing collateral factors for ETH, USDC, and TKN to be 0.7, 0.65, and 0, respectively.

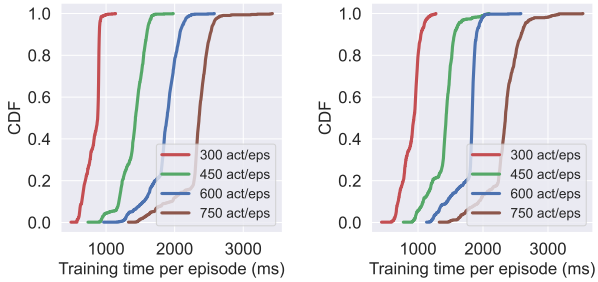
The market user in the model has a starting balance of 20,000 units of ETH, USDC, and TKN each, and initiates the three lending pools by depositing 15,000 units in each at time 0. We further assume that the market user has an initial safety supply margin and borrow margin both equal to 0.5, meaning they will supply and borrow half as much as they can initially. The supply and borrow buffers increase when the user experiences withdrawal restrictions and liquidations as described in Section V-C. The buffers also increase when the collateral factor is adjusted due to the resultant unstable image of the protocol perceived by the user. The buffer values decrease if the user experiences smooth supplies and borrows without restrictions or liquidations for over 20 steps consecutively.

4) Random events and stochastic processes: For each episode, new price trajectories (see Equation 7) and attack time points (see Section V-C1) are generated and equally applied to both the RL and baseline environments. By incorporating random events and stochastic processes in each episode, this approach prevents the RL agent from overfitting to specific price trajectories or attack timings, thereby enhancing the robustness and generalizability of the trained model.

VI. Results

In this section, we outline the implementation specifics of our approach and present the evaluation results. We record training times to gauge the model's efficiency under various configurations. The effectiveness of the model is directly quantified by the training score, defined as the cumulative rewards (see Equation 3 for reward function) at the end of each episode. Since profit maximization is the predefined objective (see Section III-C), we also evaluate *net position* N to assess the model's efficacy, which closely aligns with the training score.[3] To demonstrate the model's resilience to adversarial conditions, we report training score and *net position* N in scenarios both with and without attacks.

A noteworthy mention is that Section V-D provides details on how we initialize the DeFi environment.



(a) Without target networks. (b) With target networks.

Figure 6: Cumulative distribution functions (CDFs) of training time per episode with different numbers of actions per episode.

A. Hyperparameter tuning

Our approach is implemented using PyTorch for the governance agent and OpenAI Gym for the DeFi environment. The training evaluation processes are performed on a Mac laptop with an M2 Pro CPU and 24 GB of RAM.

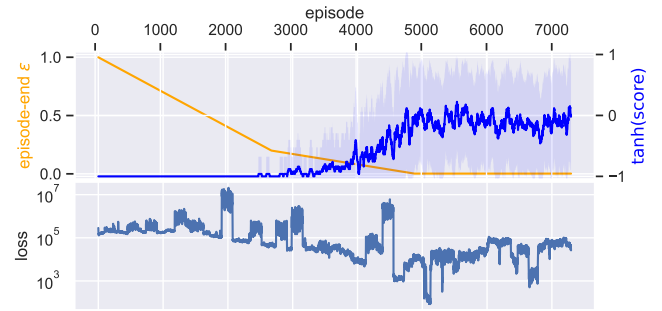
Fine-tuning hyperparameters is an essential part of building the governance agent. Table I provides a detailed list of the key hyperparameters used in the governance agent and its DQN. Due to space constraints, we skip the lengthy process of fine-tuning hyperparameters and place the justifications for key hyperparameter selections in Appendix C.

B. Training of the governance agent

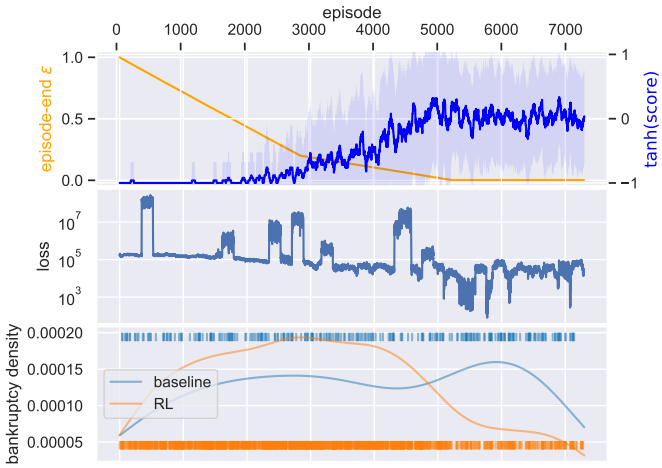
Our initial evaluation focuses on the training of the governance agent, particularly examining the speed and effectiveness of its training process.

First of all, the training speed is crucial due to the need for retraining in changing DeFi environments or protocol updates. Figure 6 depicts the cumulative distribution functions (CDFs) of training time per episode for varying numbers of actions. Training speed is crucial for the practicality of our approach, as the governance agent may need updating or retraining in response to different DeFi environments or protocol changes. The results demonstrate that for all scenarios (i.e., 300, 450, 600, and 750 actions per episode), the training time remains below 3000ms per episode. With 300 actions per episode, the DQN can finish decision-making and training for each episode within just 1000ms. Our experiments indicate that a higher number of actions per training step can slightly enhance training quality but can also considerably prolong the training duration. Based on our findings, we determined that training with 600 actions per episode is adequate for the rapid and

[3]The training score differs from the *net position* N in that the former reflects the RL's excess *net position* relative to the baseline model's *net position* after deducting any applicable penalties.



(a) Without attacks.



(b) With attacks.

Figure 7: Changes in episode-end epsilon ϵ, score, and loss with increasing training episodes. Bankruptcy occurs exclusively (though not always) in the presence of attacks; in the last plot of (b), the occurrence of bankruptcy at each episode is marked by a short vertical line "|", and its density approximated by kernel density estimation (KDE) curves.

efficient training of the governance agent. Consequently, we utilize 600 actions per training episode in all subsequent evaluations, consistently yielding satisfactory results. Moreover, by comparing Figures 6a and 6b, we observe that using target networks does not noticeably increase training time, as their time costs are very similar. The remaining evaluation results are derived from training that incorporates both the primary network and the target network, as detailed in Appendix B. The target network is activated only when ϵ is less than 0.3 (indicated in Table I). Moreover, as outlined in Table I, the agent requires 7,301 episodes for full training, resulting in a total training time of approximately 4 hours.

After landing on the appropriate hyperparameters, we train the governance agent for two scenarios: one without attacks and one with price oracle attacks occurring at random steps within each training episode (see Section V-C). Figure 7 illustrates the changes in epsilon, losses, and scores as the number of training episodes increases in both scenarios. Epsilon (ϵ) in RL represents the probability of the agent selecting a random action to explore; otherwise, it chooses the best-known action to exploit. As shown in Table I and Figure 7, we initialize ϵ at 1 to emphasize exploration during the early training phase. Over the first 5,000 episodes, ϵ decays gradually, shifting the agent's focus towards exploitation as it learns optimal actions. Afterwards, ϵ stabilizes at 5×10^{-7}, prescribing the agent to concentrate on exploitation.

To enhance interpretability and mitigate fluctuations caused by randomness, we employ the $\tanh$ function to normalize scores to the range $(-1, 1)$ [43]. Our experiment results in Figure 7 indicate that the governance agent converges effectively after approximately 4,800 episodes, with $\tanh(\text{score})$ initially increasing before plateauing; correspondingly, while losses briefly increase midway through training, they stabilize and decrease as training progresses. In the absence of adversarial attacks, the lending protocol can remain solvent (Figure 7a). However, when subjected to price oracle attacks, bankruptcy likely occurs (Figure 7b). In the baseline environment, the appearance of bankruptcies is distributed relatively evenly throughout the episodes. In contrast, in the RL environment, bankruptcies arise more frequently during the initial episodes but the frequency decreases substantially after approximately 5,000 episodes, demonstrating the effectiveness of the training process. In addition, we observe that even in the presence of attacks, the governance agent's convergence speed is not slower than in their absence. This is attributed to the target network, which helps prevent overfitting to the environment and delivers a more robust policy.

Additionally, when combined with the delay information in Figure 6, the agent can rapidly adapt to a specific DeFi environment and make robust, profitable decisions within several hours, using only the computing resources of a personal laptop. Overall, our findings suggest that our approach can effectively train a governance agent capable of making profitable and robust decisions in various DeFi environments in a timely and efficient manner.

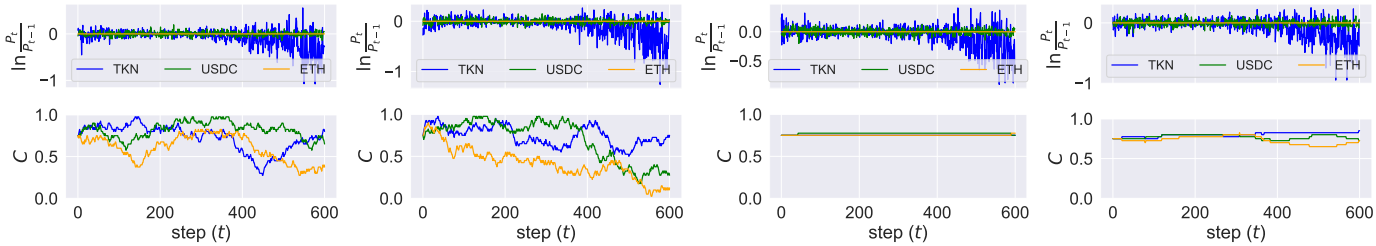
C. Governance results

In this subsection, we analyze the governance decisions implemented by the agent to determine the effectiveness of the governance agent in managing the lending protocol.

For each scenario with or without attacks, we select one early episode and one well-trained episode to demonstrate the governance results. The early episode is the first episode whose final score equals the medium score among all the episodes before the target net is switched on. The well-trained episode is the first episode that has the top 25 percentile final score among all the episodes with a positive score after the target net is switched on.

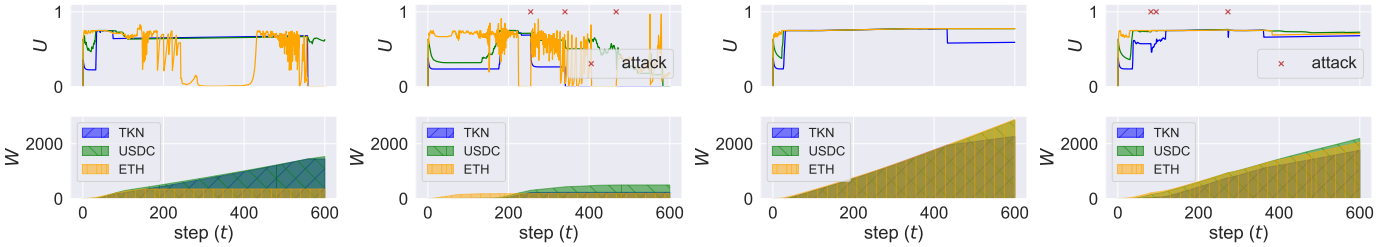
Figure 8 shows for selected episodes the RL governance agent's collateral factor determination for each token vis-à-vis the tokens' price movements[4]. We have noticed that in early episodes either with or without attacks, the agent exhibits a tendency to alter the collateral factor in a random and

[4]Borrowing a common practice from TradFi (see e.g. [44]), we use the logarithmic price return, expressed as $\ln \frac{P_t}{P_{t-1}}$, to represent price changes. Using returns makes price movements across assets comparable, unaffected by absolute price levels; using logged values symmetrically reflects the extent of price increases and decreases (in contrast, unlogged values can produce extreme positive returns that are theoretically infinite, while extreme negative returns are capped at -100%).



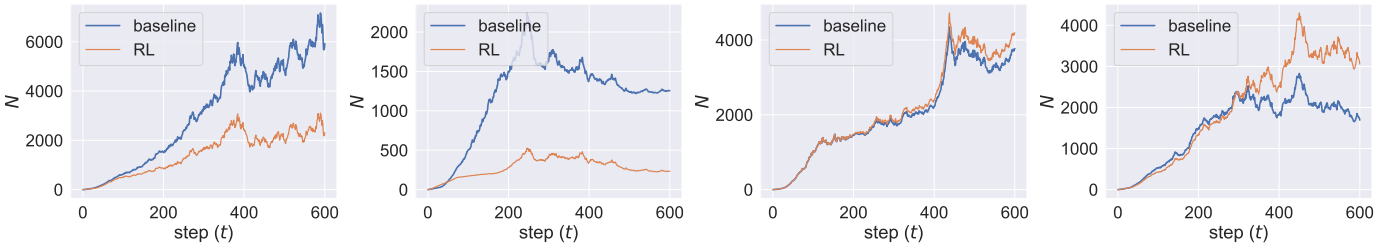
(a) Early episode, no attacks. (b) Early episode, with attacks. (c) Well-trained episode, no attacks. (d) Well-trained episode, with attacks.

Figure 8: ETH-denominated price change (expressed in log return $\ln \frac{P_t}{P_{t-1}}$) and collateral factor (C) adjustment of all tokens in the RL environment throughout selected episodes.



(a) Early episode, no attacks. (b) Early episode, with attacks. (c) Well-trained episode, no attacks. (d) Well-trained episode, with attacks.

Figure 9: Utilization ratio (U) and reserve (W) of the RL environment's lending pools throughout each of the selected episodes. The red crosses ($\times$) in (b) and (d) indicate the steps (t) at which attacks occur.



(a) Early episode, no attacks. (b) Early episode, with attacks. (c) Well-trained episode, no attacks. (d) Well-trained episode, with attacks.

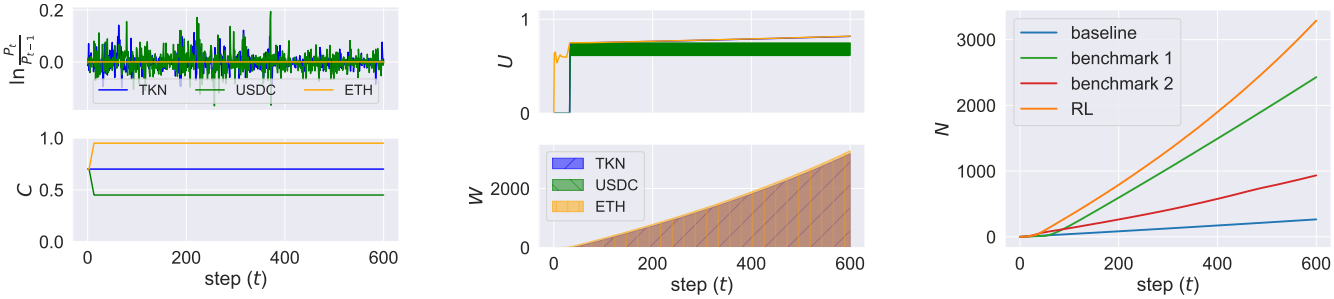
Figure 10: Protocol's net position (N) of the RL environment compared with the baseline environment (see Section V-D1).

frequent manner. Note that during this phase, the agent has not yet undergone training. After the agent has been trained, the change of collateral factor becomes infrequent. This is the correct learning outcome reacting to the design of the market user behavior that a change of collateral factor increases their supply buffer, which further decreases the funds available for borrowing and ultimately reduces the protocol's revenue.

In addition, the agent tends to adopt a more aggressive strategy, i.e. maintaining collateral factors at a consistently high level throughout the episode, in the scenario without attacks than in the scenario with attacks. This behavior is intuitive, as opting for a relatively dynamic collateral factor can enhance the lending protocol's resilience against price attacks, although it may result in reduced profitability. The agent also appears to understand that, even when the possibility of attacks is present, as long as they are infrequent, the collateral factor can still be set at a moderately high level to attract deposits and allow borrows, ultimately leading to higher profits. Being able to balance between keeping the protocol safe and maximizing profits is the key learning outcome of the agent.

Figure 9 illustrates the utilization ratio and reserve quantity of the three token assets in the lending protocol for selected episodes. Under price oracle attacks whose occurrences are marked with "$\times$" in the two subfigures, the utilization ratio of TKN typically experiences a dip as the attacker always first offsets TKN loans (see Section V-C). Under no attacks, especially in well-trained episodes when collateral factors do not undergo frequent and random alteration, the utilization ratios of all three tokens almost always remain at an optimal level of between 0.6 and 0.8. This is likely driven by the correctly encoded market user reactions that tend to be *borrow* when the utilization ratio is low (hence low *borrow interest rate*) and *supply* when the utilization ratio is high (hence high *supply interest rate*) (see Section III-B1), thus keeping the utilization ratio at a steady, equilibrated level. Comparing Figure 9c and 9d with Figure 9a and 8b, it is evident that the governance agent has learned to effectively increase the protocol reserve after being trained in scenarios both with and without attacks.

The positive outcomes of training and learning are partic-



(a) Price change in log return ($\ln \frac{P_t}{P_{t-1}}$) and collateral factor (C) generated by the RL governance agent.

(b) Lending pool states, utilization ratio (U) and reserve (W), over time in the RL environment.

(c) Protocol's net position (N) in the RL, benchmarks 1 & 2, and baseline environments.

Figure 11: Test with trained RL model using real historical price data.

ularly evident in Figure 10, which illustrate the protocol's net position. In early episodes, the undertrained RL agent underperforms the baseline either without (Figure 10a) or with (Figure 10b) the existence of attacks. The performance rank flips in later episodes when the RL agent becomes well-trained. In scenarios without attacks (Figure 10c), the RL agent consistently produces a higher net position (as calculated with Equation 1). In the presence of attacks (Figure 10d), the RL agent slightly lags behind the baseline in terms of net position during the initial steps. However, as the number of steps increases, the agent remarkably outperforms the baseline, showcasing the robust effectiveness of our model.

We further observe that the superiority of RL becomes more pronounced in the presence of attacks: while the RL agent outperforms the baseline agent by approximately 10% absent attacks (Figure 10c), this advantage surges to over 60% when attacks are present (Figure 10d), demonstrating the particular advantage of Auto.gov in the attack-prone DeFi ecosystem. Another point worth noting is that the agent's reward value is approximately equal to the difference between the orange line and the blue line in Figure 10. It is evident that well-trained agents achieve positive cumulative rewards over time, whereas under-trained agents end up with negative cumulative rewards.

D. Tests with real-world data

We further evaluate our well-trained governance agent by testing it against other approaches using real-world data.

Regarding the real-world test data, we fetch from CryptoCompare.com the time series of daily ETH-denominated prices of USDC and LINK, where LINK—the protocol token of Chainlink—plays the role equivalent to TKN's in our training stage. We do not plant any attack event, as we believe that all price manipulation activities—if any—have already been manifested in the historical price data retrieved.

We test the policy generated by the well-trained agent in comparison with not only the static baseline strategy (see Section V-D1); additionally, we compare Auto.gov with two benchmark approaches: a statistical approach and a Q-learning-based approach. Since there are no existing approaches for automated DeFi governance, we have designed and implemented these two benchmark approaches as follows.

a) Benchmark 1: statistical approach: This approach utilizes a dynamic governance strategy derived from the analysis of empirical data. The environment of Benchmark 1 is initiated like Auto.gov and the baseline environments, and the strategy used by Benchmark 1 to adjust the collateral factor in response to market volatility following the two steps below:

i) Every 7 days, the governance agent in the benchmark 1 environment first calculates the theoretical collateral factor:

$$C_{i,t} = \max(C_{i,0} - b \cdot \sigma_{i,t}, 0),$$

where $C_{i,0} = 75\%$, $b = 0.08$ and $\sigma_{i,t}$ denotes the standard deviation of daily log returns from $t-6$ to t. The value of $C_{i,0}$ corresponds to the historical collateral factor value of ETH (see Figure 13a) and the value of coefficient b corresponds to the coefficient of the 7-day volatility in a simple ordinary least squares (OLS) regression with the collateral factor being the dependent variable using historical data (the same data used to generate Table II).

ii) The theoretical collateral factor $C_{i,t}$ is compared with the current collateral factor to see whether an incremental adjustment of 0.25 (see Section III-A1a) is needed: if the theoretical value exceeds or falls below the current one by over 0.25, then a downward or upward adjustment of 0.25 is executed, respectively; otherwise, the current collateral factor remains for another 7-day cycle.

b) Benchmark 2: Q-learning-based approach: We implement an alternative automated DeFi governance approach based on RL. The environment and agent modeling methods are identical to those used in Auto.gov. However, Benchmark 2 relies on the original tabular Q-learning algorithm, which uses a Q-table [33] to store state-action pairs. Since the state is represented by continuous values, which could make the Q-table infinitely large, we discretize the state values to ensure they fit within the Q-table.

Figure 11 illustrates the test result. The collateral factor adjustment policy determined by the trained RL agent (Auto.gov) is depicted in Figure 11a. The adjustment is only concentrated within the first few steps and the collateral factors quickly stabilize for the rest of the testing episode. The infrequent adjustment aligns with what we observe in the well-trained episodes from the training stage (Figure 8c and 8d). Figure 11b

shows that all three lending pools in the RL environment consistently build up reserve over time, and maintain a healthy utilization ratio of between 0.6 and 0.8 quickly after the start of the testing episode. Finally in Figure 11c, we demonstrate our RL model's superior capability in meeting its pre-set objective of profit maximization, i.e. to increase N as much as possible (see Section III-C): the RL strategy outperforms not only the static baseline strategy but also the two benchmark approaches. Although Benchmark 2 is also based on RL, the Q-table struggles to handle large state spaces effectively, leading to significantly worse performance compared to Auto.gov and Benchmark 1. This limitation arises from the Q-table's inability to generalize across continuous state values and its exponential growth with increasing state dimensions. These shortcomings highlight the necessity of using DQN to power Auto.gov, as it can efficiently manage large and continuous state spaces through function approximation.

VII. Discussion

This section discusses the limitations of Auto.gov and proposes avenues for future improvements.

A. Limitations

a) Environment variations: For the ease of result interpretation and due to facility constraints, we used a stylized DeFi environment (see Section V-A) to achieve demonstration purposes. Depending on the actual complexity of the specific DeFi protocol and desired automation level, the training time may increase, enhanced computational resources may be required, and the agent efficacy may vary.

b) Operator risks: Our approach may be subject to risks stemming from the training operator. Currently, we are agnostic about the training operator in our framework. Depending on the exact design of the operator (e.g., DAO and elected party), they may be able to employ biased training strategies to instruct the governance agent in a manner that could generate profits for the operators themselves.

c) Adversarial machine learning attacks: Our approach might be exposed to threats stemming from adversarial machine learning tactics [45]. Firstly, adversaries might deduce the agent model by probing or observation [46], subsequently forecasting the upcoming collateral factor for conducting malicious activities. Conversely, by identifying weaknesses of the model, adversaries may destabilize the agent with certain input modifications by altering the environment. Addressing this concern entails more rigorous training, regular validation, real-time monitoring, and timely model updates [47].

B. Adjustments and expansions

For improvement and to address the limitations listed above, our framework may be further adjusted and expanded as follows:

a) More training dimensions: Instead of only training the governance agent to find the optimal collateral factor adjustment policy, we can also allow it to adjust other parameters such as other risk parameters and the interest rate model parameters.

b) More training scenarios: We can train the governance agent under various scenarios such as different market conditions (e.g. time-varying competing interest rates) and different user behaviors (e.g. various levels of borrow confidence decrease after a collateral factor drop).

c) More sophisticated machine learning techniques: We can apply more sophisticated machine learning models such as multi-agent reinforcement learning by allowing users to also be reinforcement learning agents with their own objectives (e.g. maximizing their terminal wealth).

d) Model adaptability and transferability: With some tweaks and modifications, our model can be easily extended to other lending protocols such as Compound and dForce. Upon proper amendment, the model can also be applied to other types of DeFi protocols such as automated market makers (AMMs). For example, the governance agent can dynamically adjust AMM pool parameters—such as Curve's amplification coefficient [48]—based on market conditions.

VIII. Conclusion

In this paper, we attempt to build an optimal, resilient DeFi governance solution by first abstracting a simplified yet comprehensive DeFi environment to model the governance agent's reward function, and then applying DQN RL to train the agent to find the optimal policy for protocol parameter adjustment. We use an Aave-like lending protocol, with the likelihood of price oracle attacks as an example, but our environment can be easily adjusted to apply to other protocol categories and account for different types of attacks.

Our results clearly show that the trained agent successfully adapts to various scenarios, including those with and without attacks, demonstrating its capability to understand the consequences of setting different values and the necessity of assigning appropriate values based on the specific situation. The trained Auto.gov agent also demonstrates its superiority by outperforming the static baseline approach and two benchmark approaches (one statistical and one tabular Q-learning-based) in environments with both simulated and real-world data.

Our experiment demonstrates the potential to replace the existing lengthy governance procedure, which is fully manual and entails human bias, with an RL-based approach that emphasizes security, profitability, and operating efficiency. We anticipate that more DeFi protocols will adopt (a variation of) our model for their governance process, leading to a more secure and efficient ecosystem.

Acknowledgments

We thank Alice Ng, Yitian Wang, Su Farn Fong, Kamil Tylinski and Ali Irzam Kathia for reviewing the manuscript.

This research / project is partially supported by the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Cyber Security Agency of Singapore.

This material is based upon work partially supported by Ripple under the University Blockchain Research Initiative (UBRI). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Ripple.

REFERENCES

[1] J. R. Jensen, V. von Wachter, and O. Ross, "How decentralized is the governance of blockchain-based finance: Empirical evidence from four governance token distributions," *arXiv preprint arXiv:2102.10096*, 2021.

[2] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. Knottenbelt, "SoK: Decentralized Finance (DeFi)," in *The 4th ACM Conference on Advances in Financial Technologies*. New York, NY, USA: ACM, 9 2022, pp. 30–46. [Online]. Available: https://dl.acm.org/doi/10.1145/3558535.3559780

[3] L. Gudgeon, S. Werner, D. Perez, and W. J. Knottenbelt, "DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency," in *The 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 92–112. [Online]. Available: https://doi.org/10.1145/3419614.3423254

[4] J. Xu and Y. Feng, "Reap the Harvest on Blockchain: A Survey of Yield Farming Protocols," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 858–869, 3 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9953979/

[5] H. Hamid Ekal and S. N. Abdul-wahab, "DeFi Governance and Decision-Making on Blockchain," *Mesopotamian Journal of Computer Science*, vol. 2022, pp. 9–16, 8 2022. [Online]. Available: https://mesopotamian.press/journals/index.php/cs/article/view/64

[6] V. Mohan, P. Khezr, and C. Berg, "Voting with time commitment for decentralized governance: Bond voting as a Sybil-resistant mechanism," *Available at SSRN*, 2022.

[7] A. Kiayias and P. Lazos, "SoK: Blockchain Governance," in *ACM Conference on Advances in Financial Technologies*. New York, NY, USA: ACM, 9 2022, pp. 61–73. [Online]. Available: https://dl.acm.org/doi/10.1145/3558535.3559794

[8] J. Xu and N. Vadgama, "From Banks to DeFi: the Evolution of the Lending Market," in *Enabling the Internet of Value*. Springer, Cham, 2022, pp. 53–66. [Online]. Available: https://link.springer.com/10.1007/978-3-030-78184-2_6

[9] snapshot, "Voting types," 2023. [Online]. Available: https://docs.snapshot.org/user-guides/proposals/voting-types

[10] S. Cousaert, N. Vadgama, and J. Xu, "Token-Based Insurance Solutions on Blockchain," in *Blockchains and the Token Economy*. Palgrave Macmillan, Cham, 2022, pp. 237–260. [Online]. Available: https://link.springer.com/10.1007/978-3-030-95108-5_9

[11] depressedape, "ARC: Increase AAVE Liquidation Threshold," 2021. [Online]. Available: https://governance.aave.com/t/arc-increase-aave-liquidation-threshold/1909

[12] T. Mackinga, T. Nadahalli, and R. Wattenhofer, "TWAP Oracle Attacks: Easier Done than Said?" in *International Conference on Blockchain and Cryptocurrency*. IEEE, 5 2022, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9805499/

[13] P. Qian, R. Cao, Z. Liu, W. Li, M. Li, L. Zhang, Y. Xu, J. Chen, and Q. He, "Empirical Review of Smart Contract and DeFi Security: Vulnerability Detection and Automated Repair," 9 2023. [Online]. Available: http://arxiv.org/abs/2309.02391

[14] M. Bastankhah, V. Nadkarni, C. Jin, S. Kulkarni, P. Viswanath, X. Wang, C. Jin, S. Kulkarni, and P. Viswanath, "Thinking Fast and Slow: Data-Driven Adaptive DeFi Borrow-Lending Protocol," in *6th Conference on Advances in Financial Technologies*, vol. 316. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 7 2024.

[15] H.-T. Kao, T. Chitra, R. Chiang, and J. Morrow, "An analysis of the market risk to participants in the compound protocol," in *Third International Symposium on Foundations and Applications of Blockchains*, 2020. [Online]. Available: https://scfab.github.io/2020/FAB2020_p5.pdf

[16] M. Bartoletti, J. H.-y. Chiang, and A. L. Lafuente, "SoK: Lending Pools in Decentralized Finance," in *Workshop Proceedings of Financial Cryptography and Data Security*. Springer, Berlin, Heidelberg, 3 2021, pp. 553–578. [Online]. Available: https://link.springer.com/10.1007/978-3-662-63958-0_40

[17] D. Perez, S. M. Werner, J. Xu, and B. Livshits, "Liquidations: DeFi on a Knife-Edge," in *International Conference on Financial Cryptography and Data Security (FC)*, vol. 12675 LNCS, 2021, pp. 457–476. [Online]. Available: https://link.springer.com/10.1007/978-3-662-64331-0_24

[18] C. V. Babu, M. Sudhir, L. George Kishore, and V. Sanjay Kumar, "Application of support vector machine algorithm in automated lending protocols for decentralized finance platforms," *Decentralized Finance and Tokenization in FinTech*, pp. 1–20, 6 2024.

[19] G. Palaiokrassas, S. Scherrers, E. Makri, and L. Tassiulas, "Machine Learning in DeFi: Credit Risk Assessment and Liquidation Prediction," in *International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 5 2024, pp. 650–654. [Online]. Available: https://ieeexplore.ieee.org/document/10634435/

[20] A. P. John, J. Devaraj, L. Gandhimaruthian, and J. A. Liakath, "Swarm learning based credit scoring for P2P lending in block chain," *Peer-to-Peer Networking and Applications*, vol. 16, no. 5, pp. 2113–2130, 9 2023. [Online]. Available: https://link.springer.com/article/10.1007/s12083-023-01526-5

[21] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017.

[22] C. Hou, M. Zhou, Y. Ji, P. Daian, F. Tramèr, G. Fanti, and A. Juels, "SquirRL: Automating attack discovery on blockchain incentive mechanisms with deep reinforcement learning," 2019.

[23] A. P. Chaboud, B. Chiquoine, E. Hjalmarsson, and C. Vega, "Rise of the machines: Algorithmic trading in the foreign exchange market," *Journal of Finance*, vol. 69, no. 5, pp. 2045–2084, 10 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/jofi.12186

[24] B. M. Weller, "Does Algorithmic Trading Reduce Information Acquisition?" *The Review of Financial Studies*, vol. 31, no. 6, pp. 2184–2226, 6 2018. [Online]. Available: https://dx.doi.org/10.1093/rfs/hhx137https://academic.oup.com/rfs/article/31/6/2184/4708266

[25] P. Treleaven, M. Galas, and V. Lalchand, "Algorithmic trading review," *Communications of the ACM*, vol. 56, no. 11, pp. 76–85, 11 2013. [Online]. Available: https://dl.acm.org/doi/10.1145/2500117

[26] L. Cao, "AI in Finance: Challenges, Techniques, and Opportunities," *ACM Computing Surveys*, vol. 55, no. 3, pp. 1–38, 3 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3502289

[27] J.-Z. Huang and Z. Shi, "Machine-Learning-Based Return Predictors and the Spanning Controversy in Macro-Finance," *Management Science*, vol. 69, no. 3, pp. 1780–1804, 3 2023. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/mnsc.2022.4386

[28] J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, "Machine learning for quantitative finance: fast derivative pricing, hedging and fitting," *Quantitative Finance*, vol. 18, no. 10, pp. 1635–1643, 10 2018. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/14697688.2018.1495335

[29] Y. Luo, Y. Feng, J. Xu, and P. Tasca, "Piercing the Veil of TVL: DeFi Reappraised," in *International Conference on Financial Cryptography and Data Security (FC)*, 4 2025. [Online]. Available: https://fc25.ifca.ai/preproceedings/94.pdf

[30] T. A. Xu and J. Xu, "A Short Survey on Business Models of Decentralized Finance (DeFi) Protocols," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2023, pp. 197–206. [Online]. Available: https://link.springer.com/10.1007/978-3-031-32415-4_13

[31] T. A. Xu, J. Xu, and K. Lommers, "DeFi versus TradFi: Valuation Using Multiples and Discounted Cash Flows," in *Digital Assets: Pricing, Allocation and Regulation*, R. Aggarwal and P. Tasca, Eds. Cambridge University Press, 2025, ch. 3, pp. 44–68. [Online]. Available: https://doi.org/10.1017/9781009362290.004

[32] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[33] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[34] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 486–489.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *NIPS Deep Learning Workshop*, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[36] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 12 2015. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[37] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[38] J. Geweke, "Macroeconometric modeling and the theory of the representative agent," *American Economic Review*, vol. 75, no. 2, pp. 206–210, 6 1985.

[39] J. Xu, K. Paruch, S. Cousaert, and Y. Feng, "SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–50, 11 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3570639

[40] K. Qin, L. Zhou, B. Livshits, and A. Gervais, "Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit," in *Financial Cryptography and Data Security (FC)*, vol. 12674 LNCS, 2021, pp. 3–32. [Online]. Available: https://link.springer.com/10.1007/978-3-662-64322-8_1

[41] S. Arora, Y. Li, Y. Feng, and J. Xu, "SecPLF: Secure Protocols for Loanable Funds against Oracle Manipulation Attacks," in *Asia Conference on Computer and Communications Security*, vol. 1. New York, NY, USA: ACM, 7 2024, pp. 1394–1405. [Online]. Available: https://dl.acm.org/doi/10.1145/3634737.3637681

[42] M. F. M. Osborne, "Periodic Structure in the Brownian Motion of Stock Prices," *Operations Research*, vol. 10, no. 3, pp. 345–379, 6 1962. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/opre.10.3.345

[43] T. Phan-Minh, F. Howington, T. S. Chu, M. S. Tomov, R. E. Beaudoin, S. U. Lee, N. Li, C. Dicle, S. Findler, F. Suarez-Ruiz, B. Yang, S. Omari, and E. M. Wolff, "DriveIRL: Drive in Real Life with Inverse Reinforcement Learning," in *International Conference on Robotics and Automation*, vol. 2023-May. IEEE, 5 2023, pp. 1544–1550. [Online]. Available: https://ieeexplore.ieee.org/document/10160449/

[44] T. Bollerslev, "A Conditionally Heteroskedastic Time Series Model for Speculative Prices and Rates of Return," *The Review of Economics and Statistics*, vol. 69, no. 3, p. 542, 8 1987. [Online]. Available: https://about.jstor.org/termshttps://www.jstor.org/stable/1925546?origin=crossref

[45] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. New York, NY, USA: ACM, 10 2011, pp. 43–58. [Online]. Available: https://dl.acm.org/doi/10.1145/2046684.2046692

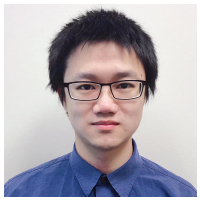
[46] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *ACM Symposium on Information, Computer and Communications Security*, vol. 2006. New York, NY, USA: ACM, 3 2006, pp. 16–25. [Online]. Available: https://dl.acm.org/doi/10.1145/1128817.1128824

[47] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine Learning in Adversarial Settings," *IEEE Security & Privacy*, vol. 14, no. 3, pp. 68–72, 5 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7478523/

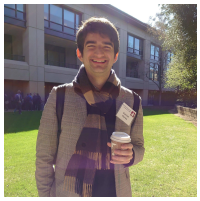
[48] nagaking, "Evaluating Proposed Parameter Changes for FRAXBP (Vote #267)," 2022. [Online]. Available: https://gov.curve.fi/t/evaluating-proposed-parameter-changes-for-fraxbp-vote-267/4317

[49] C. Spearman, "The Proof and Measurement of Association between Two Things," *The American Journal of Psychology*, vol. 15, no. 1, p. 72, 1 1904. [Online]. Available: https://www.jstor.org/stable/1412159?origin=crossref

Jiahua Xu is an Associate Professor in Financial Computing and Programme Director of the MSc Emerging Digital Technologies at UCL. Her research focuses on blockchain economics and decentralized finance, with publications in Usenix Security, ACM IMC, ACM ASIACCS, FC, IEEE ICDCS and IEEE COMST.



Yebo Feng is a research fellow at Nanyang Technological University (NTU). He received his Ph.D. degree in Computer Science from the University of Oregon (UO). His research interests include network security, blockchain security, AI security, and anomaly detection.



Daniel Perez is a software engineer and researcher in cybersecurity and decentralized finance. He is the co-founder Gyroscope, an all-weather stablecoin protocol. He received his PhD at Imperial College London sponsored by the Ethereum Foundation. He has published in Usenix Security, ACM IMC, NDSS, FC and MSR.

Benjamin Livshits is a Reader at Imperial College London and an affiliate professor at the University of Washington in Seattle, USA. Previously, he was a research scientist at Microsoft Research in Seattle for about ten years. He received a bachelor's degree in Computer Science and Math from Cornell University in 1999, and his M.S. and Ph.D. in Computer Science from Stanford University in 2002 and 2006.

APPENDIX

A. Trend of discussion in DeFi governance forum

Figure 12 illustrates the occurrences of forum posts over time across various categories, totaling 1,110 posts from July 2020 until April 2023. The discussion on smart-contract-level adjustment—e.g., the value of risk parameters, acceptance or deprecation of an asset as collateral—is typically under "Governance" and "Risk", two of the most active main categories. Evidently from Figure 12, the governance forum, especially the above-mentioned two categories, has been increasingly used during the past year, indicating the growing significance in DeFi governance perceived by the community.

B. Training with both the primary network and the target network

In this appendix, we introduce our hybrid approach to leveraging both the primary network and the target network in agent training.

In RL, applying the target network is an effective approach to preventing overfitting during training and enhancing the agent's resilience to potential attacks. However, simply employing the target network may bring the following side effects:

- **Stability vs. Responsiveness**: The target network increases stability but reduces responsiveness, causing temporary mismatches between the online network's estimates and target values.
- **Delayed learning**: The infrequent updating of the target network may lead to slower training and temporary performance drops as the online network readjusts.
- **Overestimation bias**: Using a separate target network can help mitigate overestimation bias but may result in conservative Q-value estimations, causing temporary performance drops.

To address the disadvantages associated with the target network, we propose a novel approach for its implementation. Initially, we train the agent using only the primary network, resulting in accelerated early learning. After a certain amount of training, we introduce the target network and allow ϵ to decrease at a more gradual pace. This slower reduction in ϵ means that the agent explores the environment more slowly. This strategy can aid the agent in discovering more optimal solutions. By incorporating the target network later in the training process, we aim to balance the trade-offs among stability, responsiveness, and learning speed. Algorithm 2 illustrates this training procedure, where we employ different ϵ decay functions for the primary and target networks, with the latter being slower. The evaluation results presented in Section VI of the main body are derived from the training using Algorithm 2.

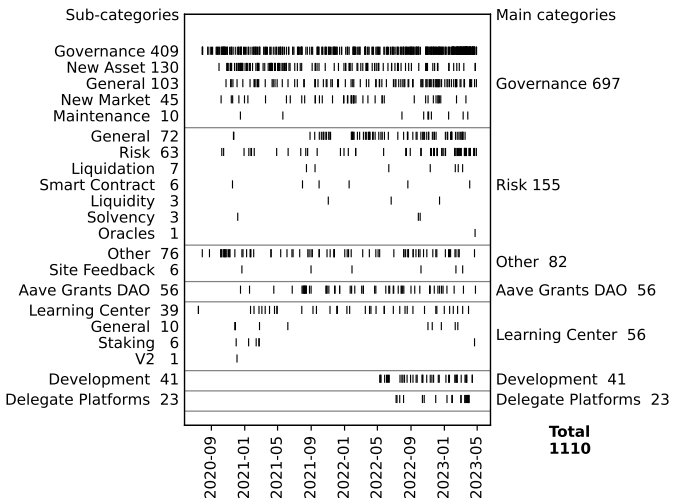


Figure 12: Statistics of Aave governance forum posts.

Algorithm 2 Training with both primary networks and target networks.

Input $total\_episode\_num$
Input $decay\_func1()$, $decay\_func2()$ ▷ Input two different decay functions
Input $target\_switch\_on\_point$
$Ctr \leftarrow 0$ ▷ Initialize the counter
$\epsilon \leftarrow 1$
while $Ctr \leq total\_episode\_num$ **do**
  $Ctr \leftarrow Ctr + 1$
  if $\epsilon < target\_switch\_on\_point$ **then**
    Switch on the target network
    $\epsilon \leftarrow decay\_func2(\epsilon)$
    $train()$ ▷ Train with target net
  else
    Switch off the target network
    $\epsilon \leftarrow decay\_func1(\epsilon)$
    $train()$ ▷ Train without target net
  end if
end while

C. RL key hyperparameters

Our research included extensive experiments to determine the optimal hyperparameters for our methodology. In this appendix, we present the rationale behind key hyperparameter selections.

For the structure of a neural network, particularly the number of hidden layers and neurons per layer, it is generally observed that higher numbers can enhance the network's ability to learn and interpret complex patterns. However, this increase in complexity is not without its drawbacks. Key challenges include the potential for overfitting, where the model excessively learns from the training data to the detriment of its performance on new data; heightened computational demands during both the training and inference phases; and the phenomenon of diminishing returns, where additional complexity fails to significantly improve model performance. Our experiments involved testing various combinations of hyperparameters, such as networks with 1, 2, 3, 4, or 5 hidden layers and varying numbers of neurons per layer (i.e., 64, 128, 256, 512). Through this rigorous testing process, we have identified an optimal configuration: a neural

network architecture comprising 3 hidden layers, each with 256 neurons. This structure strikes a balanced compromise, delivering robust performance while maintaining reasonable computational efficiency.

Another key hyperparameter that has a substantial impact on neural network training is the batch size, which is the number of training examples used in each iteration, and influences several aspects of learning dynamics, including learning speed, generalization, and memory constraints. Through extensive testing of various batch sizes (16, 32, 64, 128, 256, 512, and 1024), we noted distinct optimal sizes for different scenarios in DeFi environments. For scenarios involving attacks, a batch size of 32 proved the most effective, striking a balance between rapid convergence and model generalization. In contrast, for scenarios without attacks, a larger batch size of 256 was optimal, utilizing computational resources more efficiently while still maintaining effective memory management. These chosen batch sizes complement our learning rate well, fostering stable and efficient training dynamics. They represent a tailored approach, recognizing the unique demands of different operational environments in DeFi systems.

Last but not least, a hyperparameter worth mentioning in the context of reinforcement learning is gamma (γ), the discount factor. This parameter is crucial for balancing the importance of immediate versus future rewards. Lower gamma values, such as 0.1, lead the model to prioritize immediate rewards, while higher values like 0.9 shift the focus toward long-term benefits. In our model, we set $\gamma = 0.5$, indicating that we value immediate gains and long-term outcomes equally. Such a balanced choice is particularly strategic in environments where it is essential to weigh short-term actions against their potential long-term impact. Selecting a mid-range gamma value thus allows our model to be responsive to current situations while still considering the repercussions of its actions in the future.

D. Crypto-asset markets on Aave

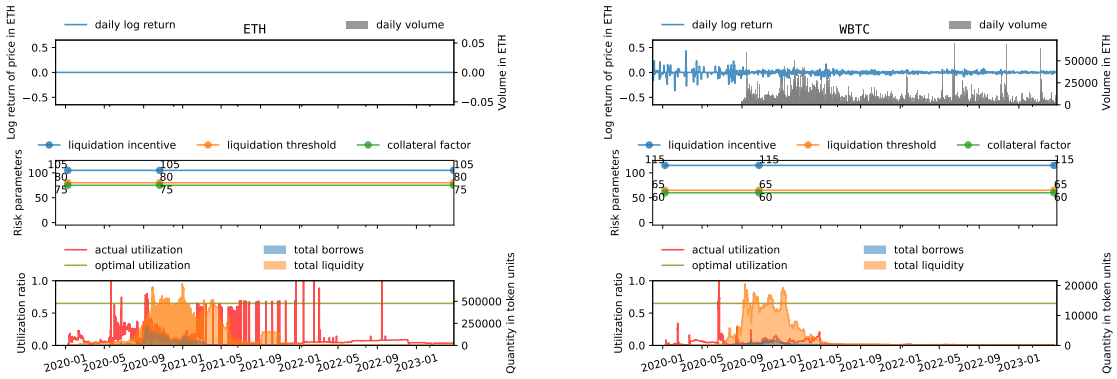
Figure 13 presents the market condition and lending pool state history of tokens in the Aave protocol from January 2020 to April 2023. For each token, we plot the time series of the daily log return of its ETH-denominated price, as well as the trading volume against ETH. We additionally delineate the historical values of each lending pool's three risk parameters: liquidity incentive, liquidation threshold, and collateral factor. Finally, we show the history of each pool's total borrows and total liquidity—both in the unit of the underlying token, as well as its actual and optimal utilization ratios.

In general, assets with higher price volatility and lower liquidity are more likely to cause protocol insolvency, all other factors being equal. Consequently, risk parameters are set on an asset-by-asset basis and may be adjusted, albeit with low historical frequency, in the event of a significant change in the asset's risk profile. The collateral factor and liquidation threshold should theoretically be set lower for assets with higher price volatility, while the liquidation incentive should be set higher for assets with lower market liquidity. Empirically, we observe a similar pattern. Table II displays the correlation matrix between various risk parameters and asset risk metrics of assets listed on Aave, calculated based on asset-day-level observations from January 2021 to April 2023. As expected, the collateral factor and liquidation threshold are negatively correlated with the asset's volatility—measured by the 7-day standard deviation of daily logarithmic returns, and the liquidation incentive is negatively correlated with the asset's market liquidity—measured by the 7-day average daily trading volume.

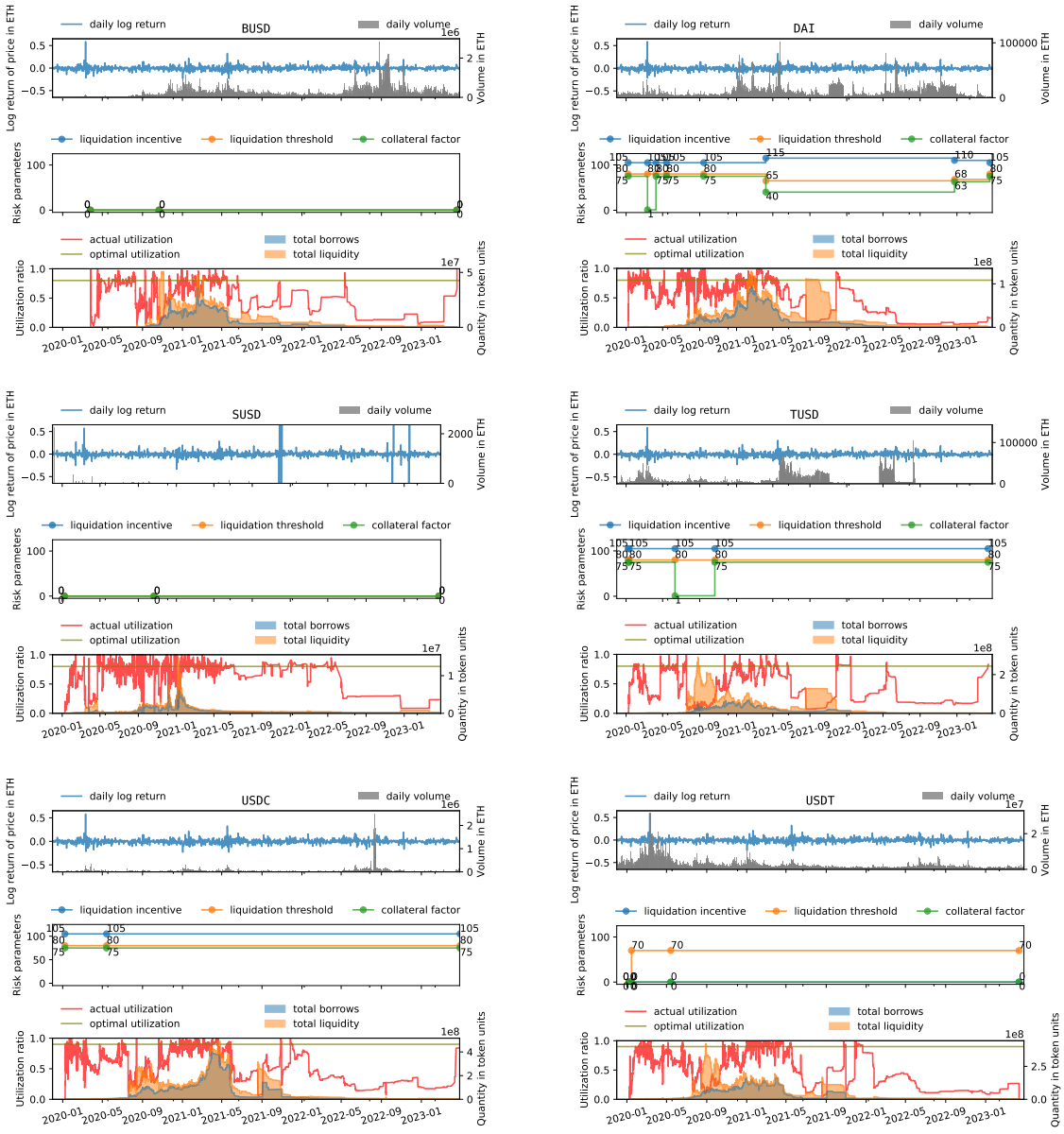
Table II: Spearman's Rank Correlation Coefficient [49] between Risk Parameters and Asset Risk Metrics of a Assets of Aave, Based on Asset-day-observations from January 2021 to April 2023.

	liquidation incentive	liquidation threshold	collateral factor	7-day volatility	7-day average volume
liquidation incentive	1.000^{***}	-0.036	0.258^{*}	0.140	-0.231^{*}
liquidation threshold	-0.036	1.000^{***}	0.670^{***}	-0.114	0.269^{**}
collateral factor	0.258^{*}	0.670^{***}	1.000^{***}	-0.320^{**}	-0.021
7-day volatility	0.140	-0.114	-0.320^{**}	1.000^{***}	0.019
7-day average volume	-0.231^{*}	0.269^{**}	-0.021	0.019	1.000^{***}

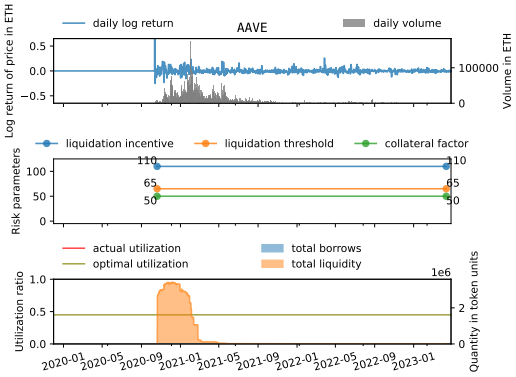
*, **, and *** denote the 1%, 5%, and 10% significance levels, respectively.

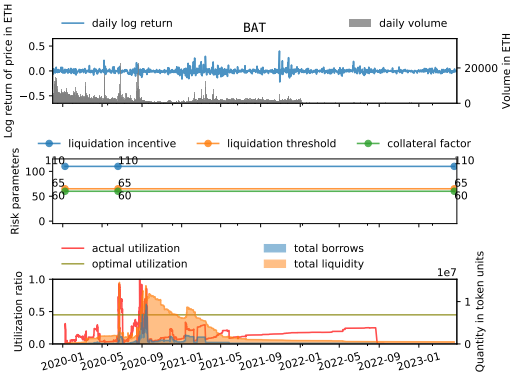


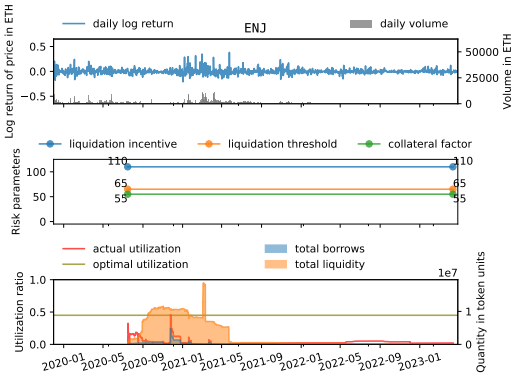
(a) Distributed ledger consensus-layer tokens

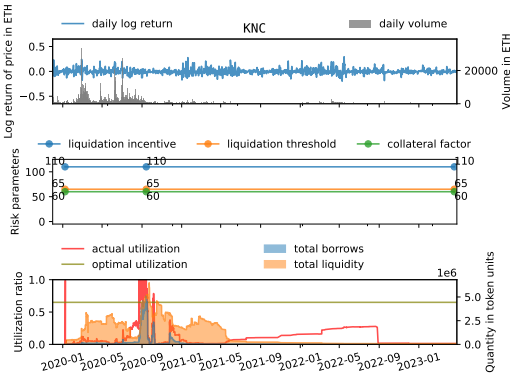


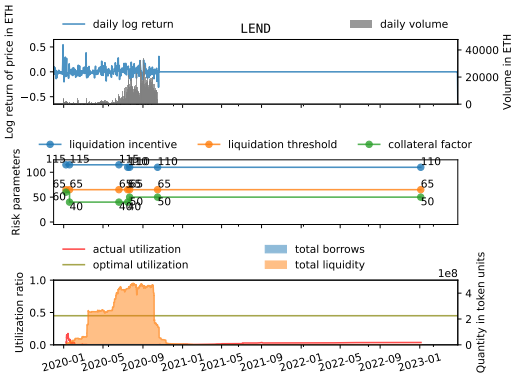
(b) USD-pegged stablecoins

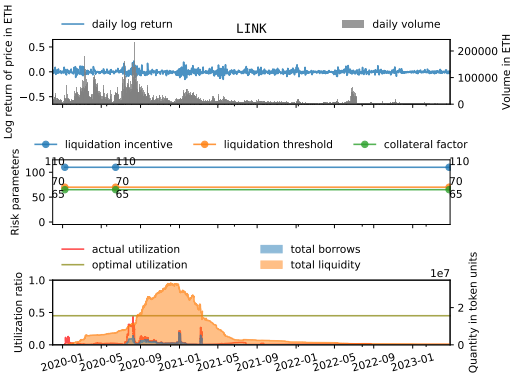

AAVE
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units
1e6

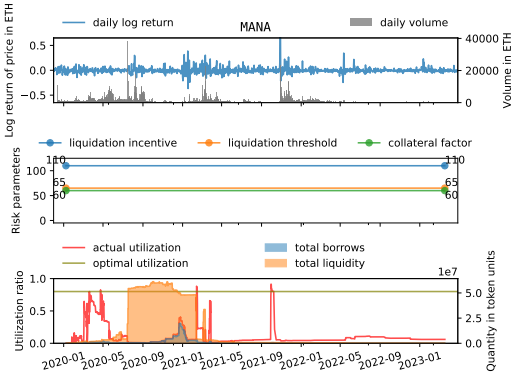

BAT
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units
1e7

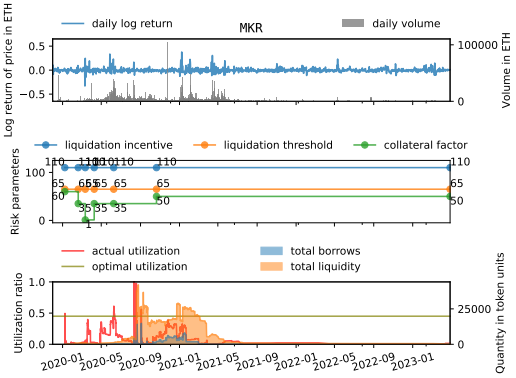

ENJ
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units
1e7

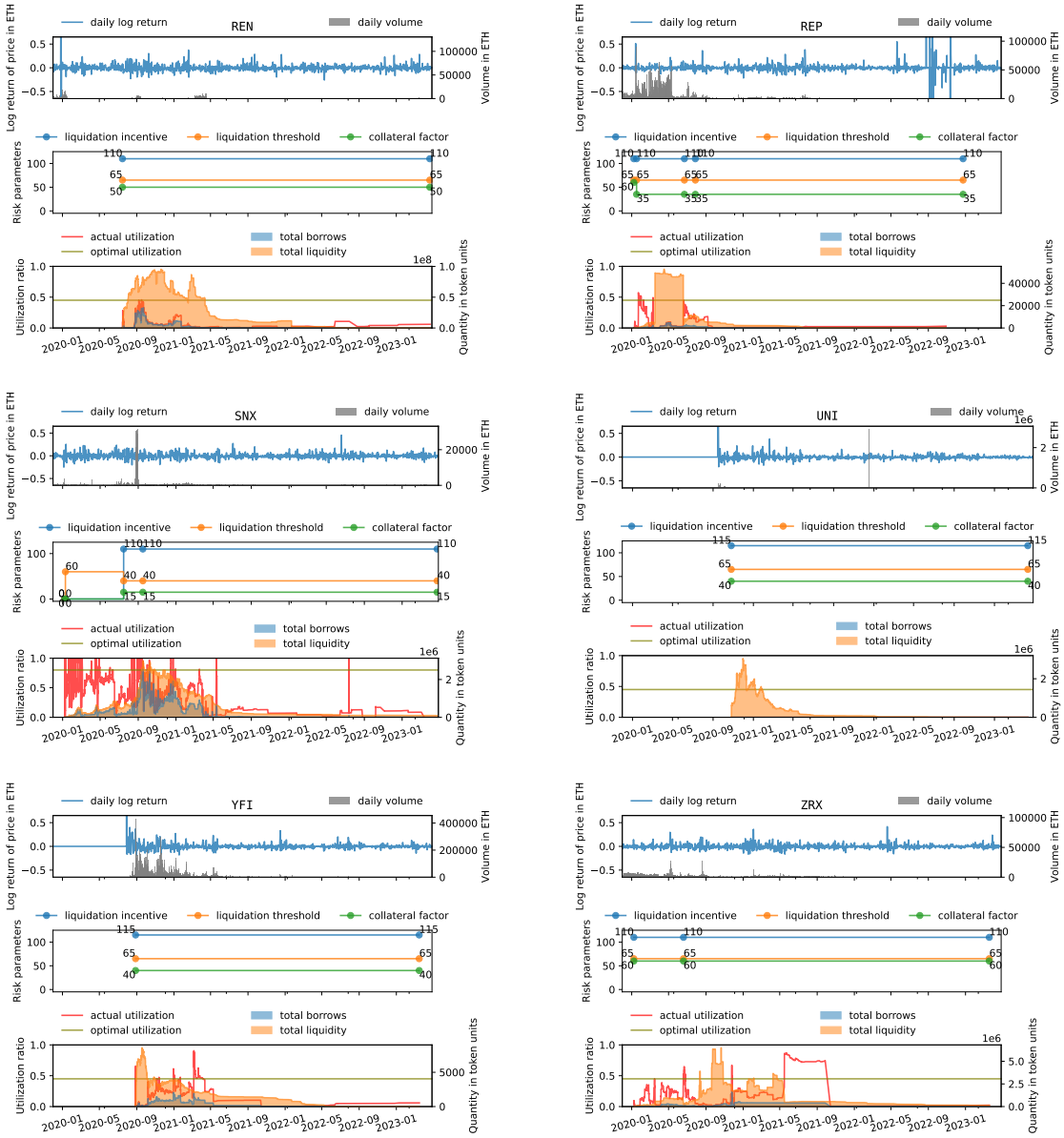

KNC
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units
1e6


LEND
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units
1e8


LINK
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units
1e7


MANA
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units
1e7


MKR
daily log return
daily volume
Log return of price in ETH
Volume in ETH
liquidation incentive
liquidation threshold
collateral factor
Risk parameters
actual utilization
optimal utilization
total borrows
total liquidity
Utilization ratio
Quantity in token units



(c) Application-layer protocol governance tokens

Figure 13: Time series of various crypto-assets' market conditions (return, volume) juxtaposed with their corresponding risk parameter (liquidation incentive, liquidation threshold, collateral factor) values, as well as other state variable values (utilization ratio, total borrow, total liquidity) of lending pools on Aave.