# Rules for Expectation: Learning to Generate Rules via Social Environment Modeling

Jiyao Pu, Haoran Duan, *Graduate Student Member, IEEE*, Junzhe Zhao, and Yang Long, *Senior Member, IEEE*

***Abstract*— The evolution of natural life is guided by a perpetually adaptive set of rules, encompassing natural laws, human policies, and game mechanics. Automated game design, through the creation of simulated environments populated by AI agents, embodies these rules, aligning with the objectives of artificial life research that seeks to replicate the dynamics of biological life through computational models. This paper presents a comprehensive framework, the Rule Generation Networks (RGN), devised for automated rule design, evaluation, and evolution in line with controllable expectations. We refine and formalize three cardinal elements - rules, strategies, and evaluation - to elucidate the intricate relationships inherent in rule generation tasks. The RGN integrates generative neural networks for rule design and a suite of reinforcement learning models for rule evaluation. To exemplify rule evolution and adaptation across varying environments, we introduce a controllability metric to gauge game dynamics and evolve the rule designer accordingly. Furthermore, we develop two game environments, Maze Run and Trust Evolution, modelling human exploration and societal trade dynamics, to gamify and evaluate the generated rules.**

***Index Terms*— Rule generation, procedural content generation, artificial life, generative networks, reinforcement learning, automated game design.**

## I. Introduction

IN DIVERSE contexts, 'rules' are delineated as explicit or implicit directives that regulate conduct or outline a procedural blueprint within a specific activity domain. This scope can range from the operational guidelines governing games [1], to the functional principles directing machinery operation [2], and extend to societal laws that influence our collective behaviour [3]. Rules are typically manifested through alterations in values or the instantiation and annihilation of objects [4]. Carefully constructed rules can nurture equitable environments, fostering cooperation and trust, whereas inefficient ones can undermine societal productivity [5]. The task of rule generation is ubiquitous, featuring prominently in areas such as game development [6], rulemaking processes [7], and legislative procedures [8]. Within the scope of game creation, rule design is considered one of the six core elements [9]. Previous research related to rules in the field of machine learning has primarily focused on rule-based learning [10]. Contrary to formulating new rules, the majority of AI-related research prioritizes training models to address specific problems within the constraints of established rules, with applications such as game map generation [6] and elucidation of elementary reactions in chemical kinetics [11]. In well-structured games that accurately reflect real-world scenarios, the potential of automated rule design extends to realms beyond mere gameplay, notably in the fields of medical and electrical engineering. Examples include the automated design of personalized cancer treatment protocols [12] and efficient cooling power modules [13]. Automated rule design can enhance the adaptability of products, increase the efficiency of design processes, and optimize design parameters, materials, and configurations.

Artificial life constitutes the study that explores systems analogous to natural life and evolutionary processes, employing simulations via computer models, robotics, and biochemistry [14], [15], [16]. This discipline has investigated living systems through a synthetic approach, essentially constructing life to gain a deeper understanding of it [17]. Examples of such endeavours include cellular automata [18], machine aquariums [19], and neural MMOs [20]. These projects strive to emulate the evolution of life, aquarium systems, and societal resource changes, thereby augmenting our comprehension of the inherent rules or patterns governing our world.

The metaverse has gained significant interest in industry and academia as a research and technological exploration of the immersive future of the internet. Especially in the gaming domain, it allows users to create avatars and engage in various activities within online virtual worlds, including social interactions, customized environments, and virtual economies [21], [22]. The metaverse encompasses four significant domains: content creation, access and social connectedness, identity and representation, and assessment, validation, and user research [23]. The sustenance of metaverse user experiences depends on two aspects: content experience and content creation. Gamification bridges the gap between content experience and creation, facilitating the construction

Manuscript received 31 May 2023; revised 6 October 2023; accepted 13 November 2023. Date of publication 20 November 2023; date of current version 12 August 2024. This work was supported by the Royal Society International Exchanges Scheme-Towards Collaborative Cloud-Edge Deep Learning Deployment under Grant IEC\NSFC\223523. This article was recommended by Associate Editor S. Li. *(Corresponding authors: Yang Long; Haoran Duan.)*

Jiyao Pu, Haoran Duan, and Yang Long are with the Department of Computer Science, Durham University, DH1 3LE Durham, U.K. (e-mail: jiyao.pu@durham.ac.uk; haoranduan28@gmail.com; yang.long@ieee.org).

Junzhe Zhao is with the School of Computing, Newcastle University, NE1 7RU Newcastle upon Tyne, U.K. (e-mail: zhaojunzhe_bit@163.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2023.3334526.

Digital Object Identifier 10.1109/TCSVT.2023.3334526

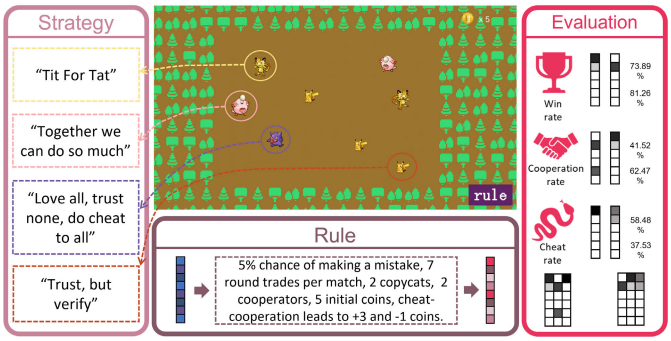



Fig. 1. Given a conceptual description. Linguistic form rules are translated into a set of rule vectors and implemented in a digital game environment. Players develop strategies based on the current rules and their experience, and the rules are evaluated within the game environment.

of virtual worlds by developers and granting player access [24].

Procedural content generation (PCG) represents an algorithmic method for creating elements automatically, leveraging a confluence of human-generated assets, computer-mediated randomness, and computational processing power [25]. It has found extensive application in game design, such as *No Man's Sky* [26], *Minecraft* [27], and *RimWorld* [28]. Within the realm of the gaming industry, a multitude of applications of procedural content generation (PCG) can be categorized as "constructive" techniques, sequentially employing grammar or noise-based algorithms to generate content devoid of evaluation. It also enhances the diversity of game experiences and mitigates the repetitive workload typically associated with design tasks [29]. However, the efficacy of PCG is significantly contingent upon the design and execution of sophisticated algorithms, which often necessitate substantial effort to design and evaluate. While it is common to examine existing game content for inspiration, machine learning methods have far less commonly been used to extract data from existing game content in order to create more content.

Game generation lies at the intersection of a multitude of creative domains, from art and music to rule systems and architecture [4]. Distinct from conventional AI, artificial general intelligence (AGI) is premised on the idea that machines could potentially mimic human cognitive processes in the future [30]. The establishment of General Video Game Artificial Intelligence (GVGAI) was motivated by a desire to steer AI researchers away from an over-reliance on specific tasks or algorithms in game engineering [29]. The use of generative models to construct not only game entities such as maps, characters, audio, and level systems, but also game rules, demonstrates significant potential within the context of GAGAI. Generative models play a vital role in unsupervised learning, offering an efficient means to analyze and comprehend unlabeled data [31]. These models, through learning from the data, develop an understanding of the internal probabilistic distribution necessary for content generation [32]. Significant improvements have been made in generative models such as Variational Autoencoders (VAE) [33], [34], Generative Adversarial Networks (GAN) [35], [36], and flow-based models [37], [38]. Despite these strides, a preponderance of research on generative tasks remains concentrated within the domains of computer vision and natural language processing.

Reinforcement learning (RL) is a mathematical framework for experience-driven autonomous learning [39]. They are designed to learn decision-making and have been employed to address challenges posed by Atari games [40]. Additionally, Multi-Agent Reinforcement Learning (MARL) represents a subfield of reinforcement learning. Multi-agent reinforcement learning concentrates on studying the behaviours of multiple learning agents co-existing in a shared environment [41]. Deep learning facilitated the scalability of reinforcement learning (RL) to address decision-making challenges that were previously deemed intractable, specifically in settings characterized by high-dimensional state and action spaces. Moreover, popular algorithms within deep RL, such as the deep Q-network (DQN) and trust region policy optimization (TRPO), have garnered extensive utilization in the realm of game design.

This project aims to establish a framework for rule generation, evaluation, and evolution. To achieve this, two digital environments, Maze Run and Trust Evolution, are developed. We outline a series of rules that could be translated within these environments. Both environments serve as games for AI, non-player characters (NPCs), and humans. We also distinguished the rule generation task in three aspects: 1. There is no pre-existing dataset for model training; all data are generated and collected by the environment during training. 2. The generated rule vector must be translated into rules that the environment can comprehend. 3. Rule design requirements might be impractical and can affect the model's performance. In response to these challenges, we proposed our rule generation framework and summarized our contributions as follows:

- Three core elements, including rule, strategies, and evaluation, are refined and symbolized to clarify relationships inherent in the automated rule generation task. These elements serve as a structured framework for organizing the rule generation process, and they facilitate a deeper understanding of the different components within the RGN framework. Furthermore, this analytical approach has helped us identify three significant challenges associated with rule generation: no dataset, rule translation, and unreasonable requirements, and enlightens us to introduce the controllability for the system evaluation.
- A rule generation framework is proposed based on generative models, digital environments, and reinforcement learning models. This framework integrates neural networks with automated game design and introduces controllability for both rule designers and game environment evaluation. This framework is initialized with default rules and accepts expected results as input. It evaluates the generated rules based on agents' strategies and refines the rule design process by comparing the expectations with the evaluation outcomes.
- Two digital environments, Maze Run and Trust Evolution, are established using Python and Unity as platforms for the demonstration of automated game design. Translators are employed to connect the generated rules to the games. These environments showcase the multi-platform

adaptability of the proposed framework, provide opportunities for human participation in rule design, and effectively incorporate both cooperative and competitive social modes, which are vital for game rule evolution.

The remainder of the paper is structured as follows. Section II reviews existing methods for generative models, reinforcement learning, automated game design, and procedural content generation. In Section III, we present details for the proposed methodology, including environment creation, RGN framework, rule implementation and translation. After that, we demonstrate details about our digital environment and the experimental results in Section IV. Finally, we place important conclusions and discuss possible future works in Section V.

## II. Related Work

### A. Related Tasks and Applications

The concept of Artificial Life (ALife) was first introduced by Langton in 1989, described as *life made by man rather than by nature* [14]. It is occasionally considered synonymous with open-ended skill learning [42]. Present ALife research spans across 14 themes, including computational biology, artificial societies, and adaptation ecology [17], [43]. Supramaniam et al. have primarily focused on the microfluidics method, honing in on the molecular and cellular biology domain [44]. Environmental task-driven approaches have also flourished in ALife, often achieved through multi-agent reinforcement learning (RL) [45]. A notable contribution in social modelling is the Neural MMO, a large-scale game environment designed for RL [20]. This research attempts to use AI to simulate social patterns based on the environment or specific tasks, rather than exploring the impact of rule changes. This paper introduces the concept of artificial life into the game environment, offering the potential to apply automated game design methodologies to real-world scenarios.

Metaverse, a digital twin to the real world, embodies a symbiotic relationship with the gaming industry, modelling technologies, and social computing. The modelling of the Metaverse is supported by the utilization of game engines such as Cry Engine, Unity Engine, and Unreal Engine [46], [47], [48]. These engines simplify the development process by reducing the requirement from code, thus cultivating a milieu that closely mirrors the real world [49]. In addition, the inherent interactive modalities and scene rendering technologies in games furnish immersive and engaging user experiences, emphasizing the significance of gaming in the formulation of user engagements within the Metaverse. Furthermore, the reliance of the Metaverse on modelling technologies becomes evident in the digital transposition of physical realities and the generation of digital identities via digital twins, identity modelling, and identity addressing [50]. The two environments presented in this paper serve as representations of the real world, while the evaluator forecasts outcomes as digital twins.

Automated game design can be categorized into two focus areas: generation of game stages, levels, and structures, and generation of game rules, mechanics, and dynamics [51]. Notwithstanding, other facets like visuals, audio, and narrative also play a key role in game design. Procedural content generation (PCG) has been pivotal in creating game structure, for instance, generating levels or puzzles for existing games [52]. A shift towards procedural content generation via machine learning (PCGML) has been observed recently, leveraging existing game content to train models that produce new game content, thus eliminating the need for expert design knowledge [53]. On the other hand, the generation of game rules has seen applications of grammars, optimization, and constraints to create new rule sets for existing level designs [54], [55], [9], [56], [57], [58]. Competitions like the general video game rule generation track have spurred advancements in this area, demonstrating the effectiveness of both constructive and genetic algorithm approaches [59]. Our automated rule design study begins with identifying key elements of general rule design, and establishes a machine learning-based framework, demonstrating the potential for complex rule creation.

Procedural content generation via machine learning (PCGML) has garnered significant attention for its versatility in autonomous generation, co-creative design, data compression, and more, offering innovative solutions in game design [25]. PCGML minimizes the need for human input during generation by leveraging representative content for autonomous generation, making it ideal for online content generation, such as in rogue-like games [60]. Moreover, PCGML facilitates co-creation, enabling efficient collaboration between human designers and algorithms in content creation [61], [62]. Notably, PCGML supports content repair by identifying unplayable areas and offering corrective suggestions [63], [64]. In terms of critique and analysis, PCGML outperforms other PCG approaches by providing in-depth analysis and critique of game content [65]. PCGML is also effective in data compression, particularly with autoencoders, allowing for efficient storage of game content [26]. The versatility and efficiency of PCGML in these domains highlight its potential for shaping the future of game design. The proposed framework takes advantage of PCGML to improve the efficiency of designer training and result evaluation.

### B. Related Learning Paradigms

Deep generative models constitute a framework that represents the distribution of generative models using deep neural networks. Recent advancements, such as ChatGPT, BERT, and DALL-E 2 [66], have demonstrated significant potential in recent years. These models can generate text and images based on textual descriptions. Wang et al. demonstrated the performance of generative models in image segmentation [67], [68], [69] and noise removal [70]. Ho et al. have also validated the performance of diffusion models in video generation [71], [72]. Some research teams have explored causality and relationships using generative models [73], [74], [75]. Additionally, generation tasks in the game domain mainly involve game content generation, such as assets [76] or textual elements [77]. The deep generative neural networks are introduced into the designer for rule creation.

Reinforcement learning has evolved based on Markov decision processes, wherein the selection of actions focuses on the current state and potential reward [39]. Recent advancements in RL demonstrate significant progress in multi-agent [41],

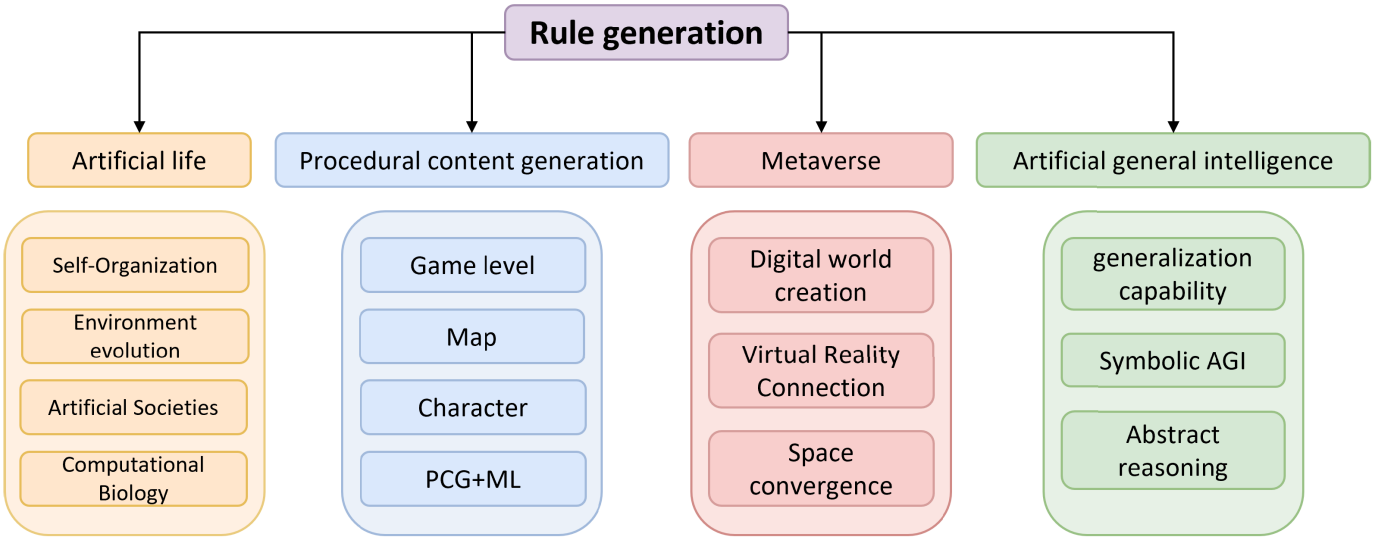



Fig. 2. Overview of the relationship among rule generation tasks, artificial life, procedural content generation, the concept of the metaverse, and artificial general intelligence. Offering a comprehensive understanding of their interconnectivity within the realm of advanced computational studies.

reward-free [78], and generation tasks [79]. The essence of RL models lies in acquiring experience based on rewards obtained from various situations. Well-trained models exhibit reliable performance when participating in games, which can assist in evaluating the game environment. Multiple reinforcement learning agents are utilised in the environments to explore strategies and evaluate created rules in this paper.

The proposed Automated Rule Generation (ARG) will be a new machine learning task in the generative model category. The implementation of the ARG involves deep neural networks as the backbone models. The Deep Generative Model is used for the rule-generation purpose. Reinforcement Learning is not a part of ARG but a learning algorithm paradigm to stimulate AI agent behaviour so that the rule and generated environment can be evaluated and evolve. We do not particularly address the shortcomings of specific models. Instead, the paper aims to propose a new paradigm and machine learning task.

## III. Methodology

The rule generation task aims to comprehend the relationship between input rule parameters and game outcomes, and subsequently train a generative model to formulate rules for specific objectives. This necessitates, at a minimum, a rule designer and an environment capable of implementing rules and recording statistics for rule evaluation. Given that the environment functions as a black box, the establishment of an evaluator, acting as digital twins, can facilitate the designer's optimization. Gamified rules within a digital environment can be engaged by RL models, NPCs, and humans.

As depicted in Fig. 3, our system framework mainly comprises three primary processes: environment development, rule designer training, and rule generation. The environment development involves creating a game with predefined rules represented by a vector. The rule designer training process strives to construct models for rule generation and train them according to the target evaluation metrics. In the rule generation process, the pre-trained generative model designs a set of rules that align with the expected outcomes.

### A. Preliminaries

Extensive-form rule generation tasks involve three critical components: **rule**, **strategy**, **evaluation**. This section presents and explains the definitions and notations of them below.

*1) Rule:* Rules can be regarded as a set of principles in a game, such as players, maps, nodes, functions, natural acts, and decisions [80]. Let $\mathcal{R} = \left[r_{n,d}\right] \in \mathbb{R}^{N_r \times D_r}$ represent a set of rules in a digital environment. Here, $N_r$ denotes the quantity of rules and $D_r$ represents the dimension of each rule. The set includes creation, deletion, and modification of rules. Rule creation increments $N_r$ to $N_r + 1$, adds it to $\mathcal{R}$, and increases the rule quantity. Conversely, deletion involves removing $r_i$ where $i \in \{1, \ldots, N_r\}$ from $\mathcal{R}$, reducing $N_r$. Rule modification updates $r_{i,d} \in \mathcal{R}$ where $i \in \{1, \ldots, N_r\}$ to $r_{i,d}{}'$. The evolved rules can be represented by $\mathcal{R}'$.

*2) Strategies:* Game strategies are a series of complete algorithms selected by players according to the situation, in compliance with rules, and determining the result [81], such as the strategy of the prisoner's dilemma. We assume $\mathcal{S} = \left[s_{n,d}\right] \in \mathbb{R}^{N_s \times D_s}$ to be strategies developed by players. Here, $N_s$ is the strategy number and $D_s$ is the dimension of each strategy. As each game may have more than one strategy and each strategy may have different stages, $D_s$ may vary. If the game is a complete information game, all strategies can be enumerated and $N_s$ can be a specific number, whereas $N_s$ can be infinite in an incomplete information game. Both humans and AI can play and develop strategies based on the reward.

*3) Evaluation:* Game evaluation is associated with high-level heuristics, including spontaneity, interruptability, and continuity [82]. These heuristics are determined by a series of specific game parameters. Let the evaluation result be denoted by $E = \left[e_{n,d}\right] \in \mathbb{R}^{N_e \times D_e}$, where $N_e$ and $D_e$ represent the evaluation metric quantity and dimension, respectively. All rules can be assessed using a set of evaluation criteria $f$ to obtain the result $E$. Each $e \in E$ represents an assessment perspective of the evaluation. Furthermore, the nature of the

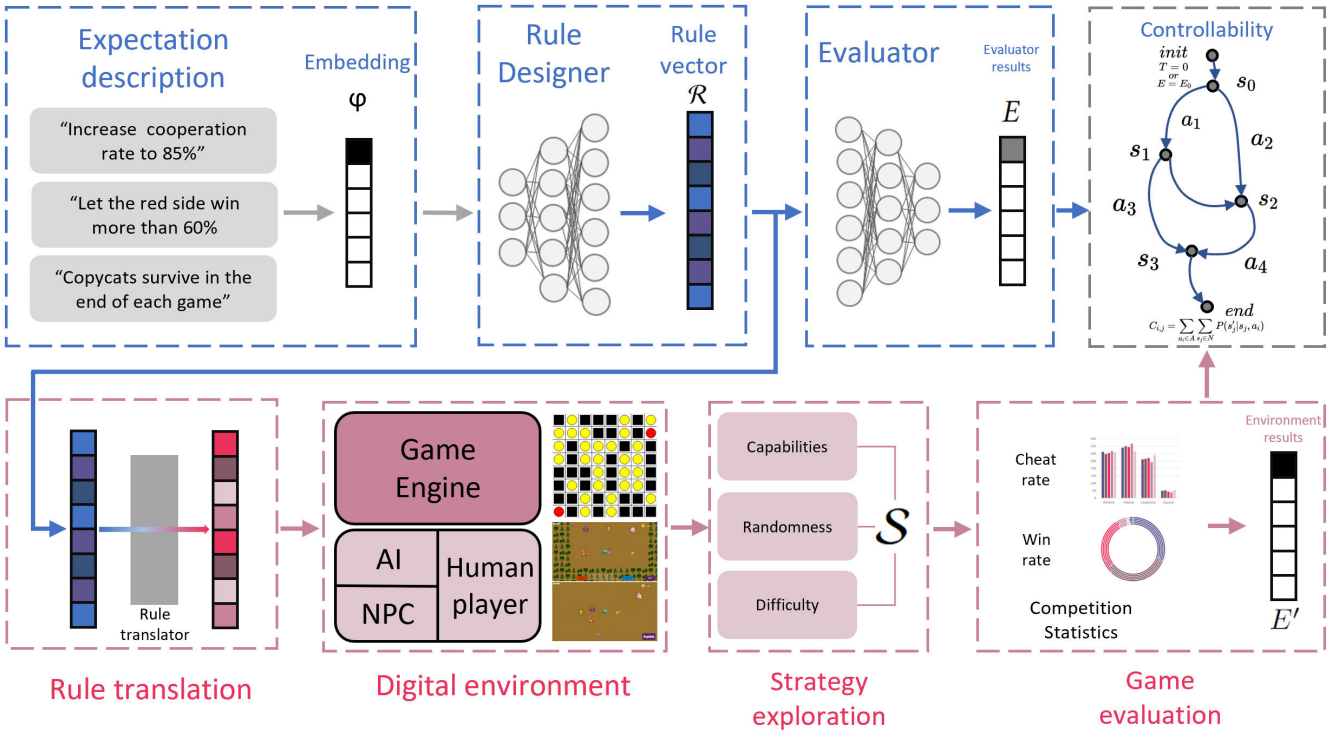



Fig. 3. An illustration of rule generation framework. The rule designer learns to create rules according to the embedded expectation and evaluates the results. The generated rule vector is then sent directly to the evaluator and translated into an executable parameter for the environment. The evaluator learns to validate the rules individually. The RL model learns strategies by exploring the environment. The controllability is tested on both the environment and the rule designer.

evaluation is determined by the game rules, while the actual results are demonstrated and recorded by agents' strategies during gameplay in the digital environment. This process can be represented as $E = f(\mathcal{R}, \mathcal{S})$, where $f$ denotes the evaluation metrics.

*4) Automated Rule Design:* Automated rule design begins with an initial rule set, $\mathcal{R}$, foundational to the game environment. Players or agents then formulate strategies, denoted as $\mathcal{S}$, which are essentially algorithms or behaviours tailored to optimize outcomes within the confines of $\mathcal{R}$. Subsequent evaluations expressed as $E = f(\mathcal{R}, \mathcal{S})$, represent gameplay metrics such as efficacy and fairness. Drawing from $\mathcal{E}$, the system discerns areas for rule modification in $\mathcal{R}$ to enhance gameplay or meet specific objectives. This cycle of strategy formulation, evaluation, and rule refinement iteratively progresses until the system meets predetermined performance or balance benchmarks.

## B. Frameworks

Figure 3 demonstrates that the system consists of three primary components: rule designer, evaluator, and digital environment. Additionally, the digital environment incorporates a set of reinforcement learning agents, non-player characters, and human players. The expectation is a set of linguistic descriptions embedded as a vector $\varphi$. The rule designer generates a set of rule vectors $\mathcal{R}$ based on embedding $\varphi$. Generated rules are then translated into accessible vectors for game platform implementation. Subsequently, the game is made available to Q-learning agents, NPCs, and human players for strategy exploration. The game, featuring experienced players, simulates the evolution of society under the generated rules. All statistics recorded during gameplay are gathered for rule evaluation. The evaluator serves as a digital twin of the environment, simulating output statistics and sharing the same raw rule vector, created by the rule designer, as input. The evaluation results are utilized to compute controllability, which is then employed to upgrade the rule designer.

The objective of generative tasks is to train a generative model, such as a variational autoencoder (VAE), generative adversarial network (GAN), or diffusion model, to create content according to specific requirements. These models are represented by a function denoted as $g : Z \to X$, which is designed to map random noise vectors $Z$ to high-dimensional output $X$. Prior to training, it's crucial to curate a labelled dataset, where the labels serve as the model's ground truth. Unlike a traditional generative model, this framework doesn't require a pre-prepared dataset. The rule generation process aims to train a designer $d : E \to \mathcal{R}$ to create rules $\mathcal{R}$. As a data-free training task, the designer takes the expected result $E$ as input, and the output rules are $\mathcal{R} = d(E)$. The created rules $\mathcal{R}$ will be implemented in the environment and the output can be represented as $E' = Env(d(E), S)$, where $Env$ is the evaluation criteria. The training of $d$ is formulated as follows:

$$\min_{d} V(d, Env) = \log(1 - Env(d(E), S)). \tag{1}$$

Although $Env(d(E), S)$ represents the ground truth, an evaluator, functioning as a digital twin, aids the designer during the backward process. The objective of the evaluator is

to emulate the environment and predict evaluation outcomes. It accepts $\mathcal{R}$ as input, analogous to the environment, and learns to score rules as $E'' = p(d(E))$, which is denoted by $p : \mathcal{R} \rightarrow E''$. Throughout training, the evaluator is refined by minimizing the discrepancy between its own output $E''$ and the ground truth $E'$.

$$\min L(E', E'') = \min L(p(d(E)), Env(d(e))). \quad (2)$$

Reinforcement learning is based on access to the *Markov Decision Process* that can be defined as the tuple $\{S, \mathcal{A}, \mathcal{T}, R, p(s_0), \gamma\}$. These elements represent states, actions, transition probabilities, rewards, initial state probabilities, and discount factors, respectively. This paper utilizes Q-learning models as RL agents. They share the same action list, reward map, and perceptual field as other players. The actions of well-trained agents can be considered objective, as they maximize the reward. The goal of strategy exploration is to train players $P : State \rightarrow Action$ according to the reward.

### C. Environment and Task

This paper presents two digital game environments for rule demonstration, referred to as *Maze Run* (MR) and *Trust Evolution* (TE), purposed to demonstrate the practicability of rule generation with judicious utilization of computational resources. These environments aim to demonstrate the feasibility of rule generation while utilizing minimal computing resources. TE serves as a fusion of artificial life and rule-generation tasks, as it simulates cooperative behaviour among individuals in a society.

*1) Maze Run:* The maze run provides a 2D map with variable height and width. It is full of reward points and traps that can be modified according to different evaluation metrics. The goal of the game is to survive as long as possible. All agents, including human players and Q-learning agents, try to find a strategy that gets more food and avoids traps. As the MR environment aims to match the rule generation task with minimising parameters, here we fix the number of agents as 2, both height and width are 6 grid, and the initial locations are the left bottom and right top corner separately.

Let $M$ be the set of 2D maps, $H$ be the set of heights, $W$ be the set of widths, $G$ be the set of grid cells, and $A$ be the set of agents. Let $h : M \rightarrow H$ be the height function that maps each map to a height. Let $w : M \rightarrow W$ be the width function that maps each map to a width. Let $l : A \rightarrow G$ be the location function that maps each agent to a grid cell. Let $f : M \times G \rightarrow$ reward point, trap be the contents function that maps each map and grid cell to the contents of the cell. Let $t : M \times A \rightarrow R$ be the time function that maps each map and agent to the time they survived. The following constraints hold: (1) For all $m \in M$, $h(m) = 6$ and $w(m) = 6$. (2) For all $m \in M$ and $a \in A$, $l(a)$ is either the left bottom or right top corner of the map $m$. (3) For all $m \in M$, $g \in G$, $f(m, g)$ is either a reward point or a trap. (4) All agents, including human players and Q-learning agents, try to find a strategy that maximizes their time function $t(m, a)$ by collecting reward points and avoiding traps. The goal of the game is to survive as long as possible, so the objective is to maximize the time function $t(m, a)$ for all agents $a \in A$.

*2) Trust Evolution:* In the Trust Evolution environment, the rules can be characterized by parameters such as payoff, population size, the round number, reproduction rate, and mistake probability. The mistake probability represents the likelihood of a player choosing the opposite action, while the round number indicates the number of trades each agent conducts. Throughout each game, players attempt to acquire more coins by engaging in trade with others, choosing between two possible actions: *cheat* or *cooperate*. The payoff maps agents' actions to trade outcomes. Six types of NPCs represent various personalities: *random*, *cheater*, *cooperator*, *copycat*, *grudger*, and *detective*. The first three types consistently choose random, cheat, and cooperation, respectively. Copycats initiate cooperation and subsequently mimic others' last actions, while grudgers always cooperate until betrayed. Detectives start with a sequence of cooperation, cheat, cooperation, and cooperation actions; if others never reciprocate cheating, they continue cheating, otherwise, they adopt the copycat strategy. At the conclusion of each match, a selection process eliminates low-performing players and reproduces top performers.

Let $R$ be the set of rounds, $N$ be the set of players, and $A$ be the set of actions (cheating or cooperation). Let $p$ be the mistake rate such that $0 \leq p \leq 1$. Let $f : N \times R \rightarrow A$ be the action function that maps each player and round to an action. Let $g : A \times A \rightarrow R$ be the payoff function that maps each pair of actions to a reward. Let $T : N \rightarrow$ random, cheater, cooperator, copycat, grudger, detective be the type function that maps each player to their personality. Let $k$ be the reproduction number, such that $k$ is a positive integer. The following constraints hold: For all $n \in N$ and $r \in R$: (1) If $T(n) =$ random, then $f(n, r)$ is chosen randomly. (2) If $T(n) =$ cheater, then $f(n, r) =$ cheating. (3) If $T(n) =$ cooperator, then $f(n, r) =$ cooperation. (4) If $T(n) =$ copycat, then $f(n, r) = f(m, r - 1)$ for some player $m \in N$ such that $f(m, r - 1)$ is the action of $m$ in the previous round. (5) If $T(n) =$ grudger, then: if there exists a player $m \in N$ and a round $s < r$ such that $f(m, s) =$ cheating, then $f(n, r) =$ cheating; otherwise, $f(n, r) =$ cooperation. (6) If $T(n) =$ detective, then: if $r = 1$, then $f(n, r) =$ cooperation; if $r = 2$, then $f(n, r) =$ cheating; if $r = 3$, then $f(n, r) =$ cooperation; if $r = 4$, then $f(n, r) =$ cooperation; if there exists a player $m \in N$ and a round $s < r$ such that $f(m, s) =$ cheating, then $f(n, r) = f(m, r - 1)$; otherwise, $f(n, r) =$ cheating. At the end of each game, the $k$ players who won the highest reward are selected for reproduction and the same quantity of lowest players are eliminated.

*3) Rules Translation:* Demonstrating game rules in a virtual environment is a cross-platform task. The rules are structured as a set of parameters which can not be understood or accessed by neural networks directly. One potential solution is extensible markup language (XML) which encodes arbitrary information into human-readable and machine-readable formats [83]. It has been introduced into some black-box optimization tasks as it can be easily translated to inputs [84], [85]. However, the description of the rule includes not only

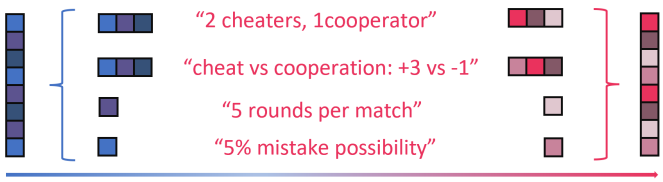



Fig. 4. An illustration of the rule translation progress. The blue vector is generated by the rule designer and subsequently divided into smaller vectors, each representing a distinct rule. These smaller vectors are then encoded as executable parameters for the environment.

objects but also relationships. To convert the generated rule vector into accessible game mechanisms within the environment, we employ rule maps. Owing to the relative simplicity of the Maze Run (MR) environment, the rule vector generated by the rule designer merely contains the reward information for each grid cell in the map. This vector takes the form of a tensor with dimensions corresponding to the map size.

The TE environment is relatively complex as the generated rule vector contains multiple rules. TE's rule is a 15-dimensional vector. The first 6 dimensions of the vector represent the population of each personality, and the total number of people is fixed. The following 7th to 12th dimensions map to the trade payoff: cheat-cheat, cheat-cooperate, cooperate-cooperate. The 13th dimension is related to the round number, and the last two dimensions represent the reproduction number and mistake possibility, respectively.

*4) Ethical Analysis:* In the realm of automated game design, the generation of rules through artificial intelligence introduces the possibility of inadvertently embedding biases into the gaming experience. This paper presents an ethical examination of our Rule Generation Network. Throughout the development of the RGN structure, we incorporated two game environments: MR and TE. It is important to highlight that players' roles within these games are devoid of any attributes related to sex, age, race, or similar socio-cultural factors. In MR, traps and rewards are solely tied to reinforcement learning rewards, eliminating potential subjective biases. Similarly, in TE, the six distinct personalities are purely representative of various NPC behavioral patterns, further emphasizing our commitment to unbiased game design.

*5) Unreasonable Expectation:* In the process of testing the RGN framework, which leverages external expectations to formulate rules, in multiple expectations for future work. We discovered challenges associated with certain expectations that proved to be unreasonable. An illustrative example is the contradiction of expecting both a cheater and a cooperator to win in the same trust evolution game. The contradiction of such unreasonable expectations not only diminishes training efficiency but also adversely impacts the evaluation score of the rule designer within the RGN. Consequently, we highlight the identification of these unreasonable expectations as a hurdle in rule generation. Addressing this challenge may require the application of logical reasoning and structured methodologies.

### D. Controllability

To ensure the rules generated rules of the designer during training are consistent with the input expectation, here we introduced controllability from information theory for designer evolution. The controllability matrix is a fundamental concept in control theory, serving as an essential tool for evaluating the controllability of linear and non-linear systems. Controllability represents the ability to guide a system from any initial state to any target state within a finite period using available control inputs. By mathematically assessing this property, the controllability matrix enables researchers and engineers to analyze the efficacy of control inputs in directing a system's state and informs the design of control strategies for various applications in engineering, robotics, and other fields.

*Proposition:* The linearized system, exemplified by the player's movement, can be expressed as:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{3}$$

$$y(t) = Cx(t), \tag{4}$$

where $x(t) \in \mathbb{R}^n$ denotes the state vector, $u(t) \in \mathbb{R}^m$ signifies the control input vector, $A \in \mathbb{R}^{n\times n}$ is the system matrix, $B \in \mathbb{R}^{n\times m}$ represents the input matrix, and $C \in \mathbb{R}^{p\times n}$ is the output matrix. The controllability matrix is defined as:

$$\mathcal{C} = [B, AB, A^2B, \ldots, A^{n-1}B], \tag{5}$$

where $\mathcal{C} \in \mathbb{R}^{n\times mn}$, and $A^kB$ corresponds to the $k$-th power of matrix $A$ multiplied by matrix $B$. The non-linear system is characterized by the following state-space equation:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \tag{6}$$

where $x(t) \in \mathbb{R}^{15}$ is the 15D rule vector, $u(t) \in \mathbb{R}$ represents the 1D win rate input, and $f(x)$ and $g(x)$ are non-linear vector functions. The Lie derivatives of vector fields $f(x)$ and $g(x)$ can be computed as:

$$L_f, L_g = \frac{\partial g(x)}{\partial x} f(x) - \frac{\partial f(x)}{\partial x} g(x). \tag{7}$$

Construct the Lie algebra $\mathcal{L}$ generated by the Lie derivatives of vector fields $f(x)$ and $g(x)$:

$$\mathcal{L} = f, g, [f, g], [f, [f, g]], [g, [f, g]], \ldots. \tag{8}$$

At a specific point $x_0$, create the distribution matrix $D(x_0)$ using the vectors in the Lie algebra:

$$D(x_0) = \left[ f(x_0)\ g(x_0)\ [f, g] \cdots \right]. \tag{9}$$

If the rank of the distribution matrix $D(x_0)$ is equal to the dimension of the state-space ($\text{rank}(D(x_0)) = 15$), then the system is locally controllable at the point $x_0$.

*Lemma:* Let $N$ denote a set of players, and $A$ represent actions (cheating or cooperation). To investigate the controllability of the system, we must first define the state-space and action-space. The controllability matrix is a matrix that associates the players' actions with alterations in the system's state. Let $C$ be the controllability matrix, with $C_{i,j}$ signifying the impact of player $i$'s action on player $j$'s state. The entries in the controllability matrix can be computed as follows:

$$C_{i,j} = \sum_{a_i \in A} \sum_{s_j \in N} P(s_j'|s_j, a_i), \tag{10}$$

where $s_j'$ is the state of player $j$ after player $i$ takes action $a_i$, and $P(s_j'|s_j, a_i)$ is the transition probability from state $s_j$

**Algorithm 1** RGN Training Algorithm

**Input**:
Linguistic description for rule expectation $E$;
Number of training epochs $Ep$.
Game environment $Env$.
**Output**:
Well-trained rule designer model: $D : E \rightarrow R$.
Well-trained Q-learning model: $Q : S \rightarrow A$.

```
Initialization: Embed the rule expectation as φ.
Add noise z to the expectation E, resulting in φ = φ + z.
for t = 1 : Ep do
    Rules generation: designer create rules R = D(E).
    Train reinforcement learning model: translate R into
    R' = T(R), learning strategies S = Env(R');
    Train evaluator: use E' = Env(R') to train the evalua-
    tor;
    Train designer: use the evaluator's result E' and input
    E to upgrade designer;
    if E'−E == 0 then
        break;
    end if
end for
Save the well-trained designer model D for rule genera-
tion.
Save the well-trained Q-learning model Q for action
selection.
```

to state $s'_j$ given action $a_i$. By calculating the controllability matrix, one can determine the degree to which the players can control the outcome of the game and whether the game is balanced or not. This can help to improve the design of the rule generator and ensure that the generated games are academically sound and enjoyable for players.

As for the designer networks, it provides a mapping from the 1D input to the 15D output space, represented as:

$$x(u) = W_3\sigma(W_2\sigma(W_1u + b_1) + b_2) + b_3. \tag{11}$$

## IV. Experiment

### A. Settings

The experiments focus on three primary objectives: (1) implementing the RGN model across multiple environments; (2) Demonstrating and gamifying the generated rules; (3) generating rules according to specific evaluation metrics. It requires the RGN model to be capable of generating rules that can be instantiated within diverse environments, allowing AI agents, NPCs, and humans to play. Training the rule designer based on various evaluation metrics, with the expectation that a well-trained designer will produce rules that align with the desired outcomes. It also aims to showcase the generated rules within an environment that provides accessible operations and visual representations of the associated statistical data. This comprehensive experimental approach seeks to validate the effectiveness of the proposed RGN model in generating engaging and meaningful game experiences.

*1) Evaluation Criteria:* The evaluation framework for the RGN model is multifaceted, encompassing a rule designer, an evaluator, Q-learning models, and two distinct virtual environments. Consequently, a diverse array of evaluation methodologies and criteria are implemented. For the rule designer and evaluator, qualitative evaluation is undertaken by integrating the generated rules into the two virtual environments (MR and TE). Quantitatively, Mean Squared Error (MSE) loss and cross-entropy loss functioned as training criteria for the rule designer and evaluator, respectively. Additionally, controllability is employed to gauge the performance of the rule designer and game platform. The distribution of potential game outcomes, ascertained by the random sampling of rules, served as a dataset during RGN training.

In establishing a baseline for the RGN, we separate it into three distinct segments, each according to its specific role and utility in the rule design continuum: the designer, evaluator, and tester. For the RGN designer, the baseline is drawn from both human and random designers. The evaluator's baseline is anchored to actual game results, ensuring an empirical point of reference. Meanwhile, the RGN tester's baseline encompasses a comparative assessment of performances across Q-learning, deep Q-learning (DQN), double DQN, categorical DQN, human participants, and NPCs. This stratified baseline approach provides a comprehensive and multifaceted reference for evaluating the efficacy and robustness of the RGN system. Moreover, the assessment of automated game design systems presents a persistent challenge for scholars in the discipline. The comparisons between individual systems remain infrequent, often limited to qualitative expositions in sections dedicated to related literature. This is primarily due to the distinct nature of each automated game design system, characterized by their unique game engines, technological infrastructures, and design philosophies [86].

*2) Implementation Details:* The MR and TE environments are developed using PyPlot and the Unity engine, respectively. In the MR environment, the generated rule vectors encompassed the reward map, whereas in TE, the rule comprised various game parameters such as population distribution, payoff structure, round count, reproduction quantity, and mistake probability. Essential metrics in the TE environment, including the Cooperation Rate (CR), Average Coin Increment (ACI), and Cooperator Proportion (CP), represented players' actions, resource fluctuations, and game trends. To maintain consistency and reliability in the TE environment, we drew upon 'The Evolution of Trust' by Nicky Case, excluding Q-learning and human players for demonstration purposes. The rule designer and evaluator are implemented as models with three fully connected layers. Q-learning models functioned as our RL agents, while a fixed response algorithm, providing specific responses to distinct inputs, is utilized for NPCs.

### B. Qualitative Evaluation

The training of the RGN is conducted across multiple game environments, including MR and TE. Due to the single rule accessibility in the MR environment, we will focus on the variations in rule design within the TE environment.

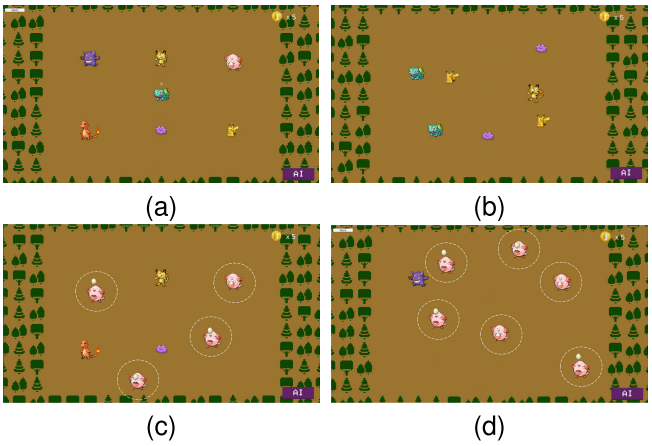



Fig. 5. Illustration of the different rule designs. The entities encompass six distinct character roles, with different appearances representing unique personalities. The rules are generated by humans, random generation, and RGN. (a) Human design. (b) Random design. (c) RGN design (payoff). (d) RGN design (population).

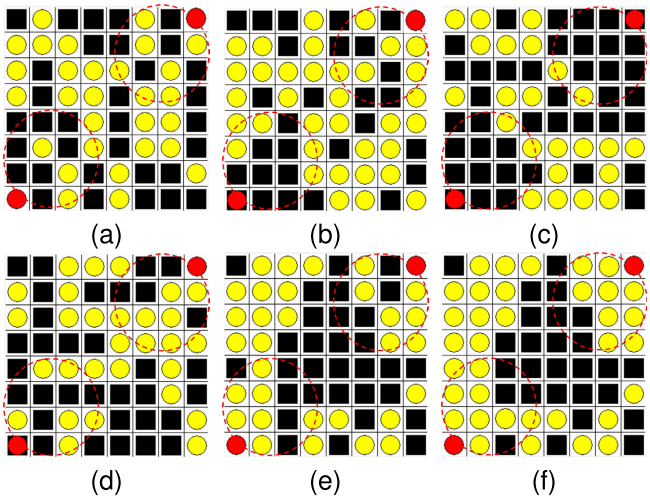



Fig. 6. Illustration of the created rules from the MR during training. The red circles denote the Q-learning agent, the yellow circles indicate reward points, and the black squares represent traps. The red dotted line highlights the primary changes of rule during training. (a) Expected win rate = 0%: epoch 0. (b) Expected win rate = 0%: epoch 10. (c) Expected win rate = 0%: epoch 100. (d) Expected win rate = 100%: epoch 0. (e) Expected win rate = 100%: epoch 10. (f) Expected win rate = 100%: epoch 100.

The implementation of the designed rules is depicted in Fig. 5. The primary design objective sought to encourage a higher survival rate for cooperators. Human-designed rules are easily distinguishable due to their orderly arrangement of roles, as the human designer strategically adjusted the payoff structure to achieve this goal. In contrast, the untrained model generated rules at random. Regarding the rules designed by the well-trained RGN model, a certain level of disorder is apparent in the placement of roles. Both rules designed by the RGN tend to prioritize the initial population over payoff or other parameters. Thus, the proportion of cooperators shows a significant increase, which means the rules are inclined to augment the initial quantity of target winners as a means to ensure they fulfil the expectation.

Fig. 6 illustrates the performance evolution of the rule designer for the MR task during the training process. The RGN

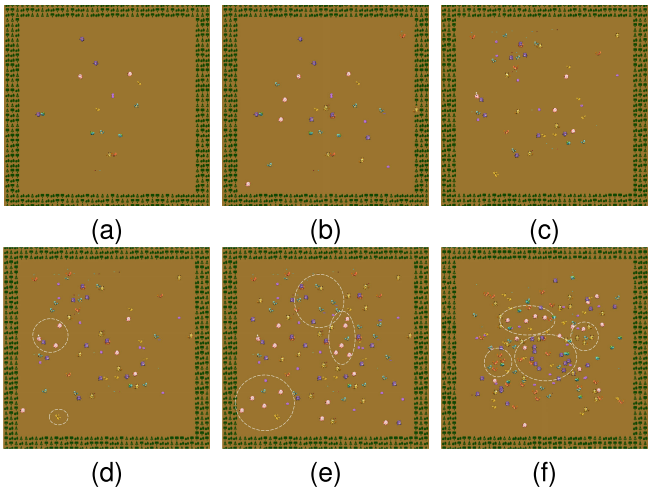



Fig. 7. Illustration of six game scenarios with varying player numbers, specifically containing 15, 30, 45, 60, 100, and 150 Q-learning agents as players, respectively. Apart from the player numbers, all other rules remain consistent across the games. (a) 15 agents. (b) 30 agents. (c) 45 agents. (d) 60 agents. (e) 100 agents. (f) 150 agents.

rule designer's aim is to manipulate the win rate by creating additional traps or rewards in the map, while the Q-learning players attempt to maximize the survival rate and reward coin number. The first row presents the training progression for creating MR game rules where both players have a 0% win rate, whereas the second row focuses on the creation of a game with a 100% win rate. As depicted in Fig. 6a, the untrained designer in the MR environment generated rules randomly. However, during training, it started generating more traps around agents' spawn points and increased trap damage as shown in Fig. 6b to Fig. 6c. In contrast, as illustrated in Fig. 6d, Fig. 6e, and Fig. 6f, the upper right and lower left corners players began to host more reward points, replacing the traps. In summary, the rule designer evolved the rules during training to align with the expected win rate.

In the rule evaluation phase, players, including humans, NPCs, and Q-learning models, endeavoured to devise strategies to maximize their game rewards. Fig. 7 presents the environment's capability to handle multi-agent tasks at varying player count levels. As each agent is required to identify their target trading role and execute trading actions, the complexity increases at a rate of $O(n^2)$. Current experiments in TE have supported up to 150 agents learning rules and developing strategies within the game. Furthermore, as the number of agents increased from Fig. 7d to Fig. 7f, we observed a pattern emerging among agents. As agents tend to trade with those who benefit them the most, some roles exhibit a preference for trading with agents who share the same personality.

Strategies developed by Q-learning players in a structured Maze Run game can be visualized as a map, provided that the learning process is driven by rewards. Figure 8 delineates three distinct games, each aiming for distinct win rates of 0%, 50%, and 100%, respectively. The first row of illustrations portrays the reward map, while the second row demonstrates the movement dynamics of two Q-learning agents. The rewards are governed by transitions in states, implying that different cells within the map may not necessarily exhibit continuity or

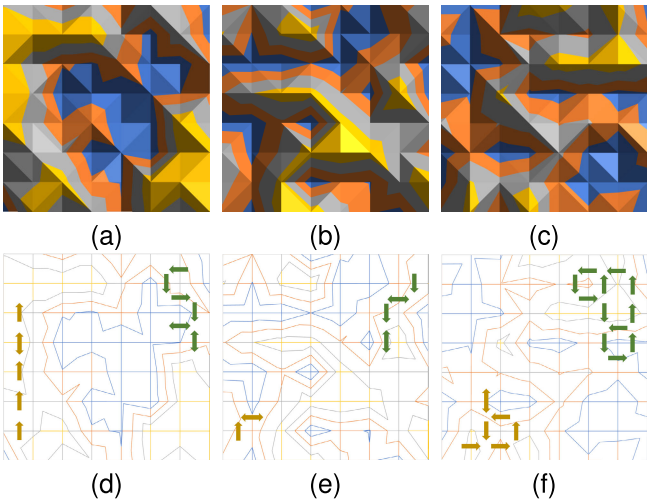



Fig. 8. Visualization of the rules and agents' strategies in the maze run game. The blocks coloured in shades of orange and yellow denote regions associated with high rewards, whereas areas signified by grey and blue correspond to relatively lower rewards. The arrows coloured in yellow indicate the potential movements of the agent that start at the bottom left, while the green arrows exemplify the prospective manoeuvres of the agent positioned at the upper right. (a) Reward map: win rate = 0%. (b) Reward map: win rate = 50%. (c) Reward map: win rate = 100%. (d) Agents' strategy: win rate = 0%. (e) Agents' strategy: win rate = 50%. (f) Agents' strategy: win rate = 100%.

linearity, despite analogous color representations. As can be observed from Figures 8e, 8f, and 8d, the players investigate regions adjacent to their initiation points, and intermittently retrace their trajectories to maximize rewards. Interestingly, during the training phase, the players also venture into areas associated with lower rewards but display rationality in formulating their ultimate strategies, as depicted in Figure 8.

Table I presents a record of players' strategies encompassed within a series of game rules in the trust evolution. These rules have been generated by an adept designer utilizing the RGN designer with the objective of achieving a 100% cooperation rate. Each trade involves two players, each of whom has the option to choose at least one other player with whom to trade, with the outcome contingent on the payoff outlined in the rules. As the goal is complete cooperation, the designer has amplified the rewards for cooperation while penalizing deceit, as demonstrated in the table. Consequently, the majority of players incline towards cooperation, with the exception of those within the 'random' category. Additionally, it has been observed that those who consistently cooperate are widely favoured among players of various personality types.

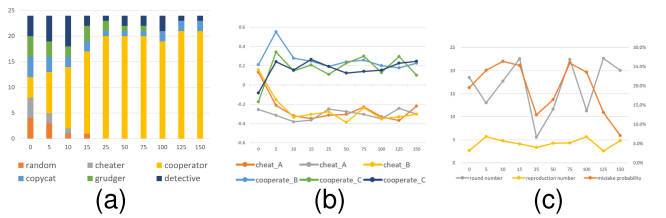



Fig. 9. Illustration of rules evolution in the TE during training. The population figure employs six colours to represent six distinct personalities. The symbols A, B, and C in the payoff figure correspond to the 'cheat-cheat,' 'cheat-cooperate,' and 'cooperate-cooperate' scenarios, respectively. The last figure reveals alterations in the round number, reproduction number, and mistake possibility throughout the training process. (a) Population. (b) Payoff. (c) Other parameters.

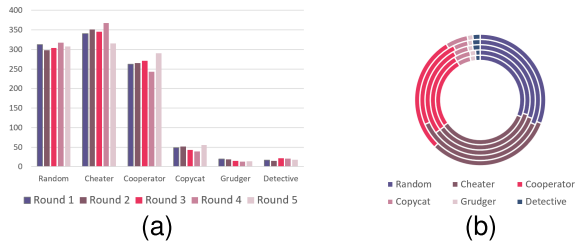



Fig. 10. The quantity and proportion of each personality of role win in five random samplings. In the histogram, columns of the same colour correspond to the same round of sampling, while in the pie chart, circles of the same colour represent the same round of sampling. (a) Size. (b) Ratio.

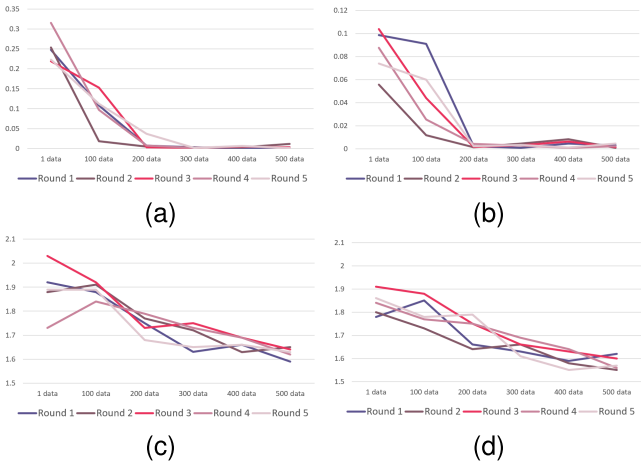



Fig. 11. Comparison of designers' and evaluators' loss change in multiple environments. Each line chart records five individual results of the same experiments. The first row presents the cross-entropy loss change in the trust evolution environment, while the second row displays the MSE loss change in the maze run environment. The first column represents the designer, followed by the evaluator. (a) Designer cross-entropy loss. (b) Evaluator cross-entropy loss. (c) Designer MSE loss. (d) Evaluator MSE loss.

### C. Quantitative Evaluation

Figure 9 represents the evolution of rules during designer training, the objective of which is to formulate rules that align with the anticipated increase in the cooperation rate to 100%. As depicted in Figure 9a, the designer tends to expand the population of cooperators while reducing the number of cheaters and random players. The population of detectives also decreases, caused by their consistent inclination to scrutinize others. Figure 9b portrays modifications in tradeoffs where, regardless of the scenarios being cheat-cheat, cheat-cooperate, or cooperate-cooperate, the designer primarily intensifies the rewards associated with cooperative behaviour. Concerning other parameters in figure 9c such as the number of reproductions, round numbers, and the probability of mistakes, the oscillation visible in their respective curves suggests a relatively minor correlation with the design expectation.

We also demonstrate the distribution of gaming outcomes, which serve as real-time data for training the designer and evaluator. The Monte Carlo method is employed to separately sample 1000 data points 5 times as depicted in Fig. 10. Rules within the TE environment are encoded in a 15-dimensional

TABLE I

STRATEGIES' RECORDS OF TRUST EVOLUTION GAME GENERATED BY WELL-TRAINED RGN NETWORKS BASED ON 100% COOPERATION RATE EXPECTATION. THE POPULATION FOR SIX PERSONALITIES ARE FIXED TO 4, AND ONE OF THE ROLES CAN CHOOSE A TRADE TARGET EACH TIME

| Trade id | 00 | | 01 | | 02 | | 03 | | 04 | | 05 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Player id | A1 | F3 | B1 | C2 | C1 | C3 | D1 | C2 | E1 | E4 | F1 | C3 |
| Operation | cheat | cooperate | cheat | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate |
| Result | win | lose | win | lose | win-win | win-win | win-win | win-win | win-win | win-win | win-win | win-win |
| Coin | +1.32 | -0.34 | +1.32 | -0.34 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 |
| Trade id | 06 | | 07 | | 08 | | 09 | | 10 | | 11 | |
| Player id | A2 | A5 | B2 | F2 | C2 | C1 | D2 | D1 | E2 | C2 | F2 | C4 |
| Operation | cheat | cheat | cheat | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate |
| Result | lose-lose | lose-lose | win | lose | win-win | win-win | win-win | win-win | win-win | win-win | win-win | win-win |
| Coin | -0.21 | -0.21 | +1.32 | -0.34 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 |
| Trade id | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | |
| Player id | A3 | F1 | B3 | A1 | C3 | C2 | D3 | D4 | E3 | E1 | F3 | C2 |
| Operation | cheat | cooperate | cheat | cheat | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate | cooperate |
| Result | win | lose | lose-lose | lose-lose | win-win | win-win | win-win | win-win | win-win | win-win | win-win | win-win |
| Coin | +1.32 | -0.34 | -0.21 | -0.21 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 | +0.87 |

TABLE II

PERFORMANCE COMPARISON OF FOUR RULE DESIGN METHODS FOR VARIOUS DESIGN REQUIREMENTS. FOUR DISTINCT RULE EXPECTATIONS, INCLUDING CR=100%, ACI=30, CP=100%, AND CP=80%, ARE FULFILLED BY FOUR TYPES OF DESIGNERS: HUMAN, RANDOM, UNTRAINED RGN, AND WELL-TRAINED RGN. THREE METRICS ARE COMPARED IN THE TABLE. THE BEST-PERFORMING DESIGN IS BOLDED

| Expectation | CR = 100% | | | ACI = 30 | | | CP = 100% | | | CP = 80% | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | **CR** | ACI | CP | CR | **ACI** | CP | CR | ACI | **CP** | CR | ACI | **CP** |
| Human designed rules | **100%** | 17 | 100% | 78% | 67 | 64% | 60% | 41 | **100%** | 100% | 23 | 52% |
| Random rules | 55% | 5 | 32% | 23% | -5 | 44% | 25% | 4 | 0% | 42% | 11 | 0% |
| Untrained RGN | 63% | -15 | 0% | 54% | 6 | 58% | 0% | -5 | 0% | 68% | 8 | 0% |
| Well-trained RGN | **100%** | 17 | 100% | 100% | **24** | 82% | 100% | 25 | **100%** | 76% | 15 | **64%** |

vector, capable of decoding into rules encompassing approximately $3.56265 \times 10^{15}$ unique cases. As Fig. 10a illustrates, there is a preponderance of random game results favouring cheaters, cooperators, and randoms as winners. In comparison, grudgers only possess a 4.764% chance of victory against cheaters. Surprisingly, copycats did not perform better, despite their potential highlighted in Case's model. The pie chart in Fig. 10b supports this finding, indicating that it is easier to design rules favouring the victory of cheaters, cooperators, and randoms, which aligns with experimental observations.

The rule designer is flexible in creating rules of varying dimensions, such as the one-dimensional win rate in the MR and the 16-dimensional parameter in the TE. Both MSE loss and cross-entropy loss are applicable to the rule designer and evaluator. Fig. 11a and Fig. 11b illustrate the progression of cross-entropy loss during the training phase in the TE environment, while Fig. 11c and Fig. 11d concentrate on the MSE loss in the MR. The decline in all curves attests to the improved performance of both evaluators and designers throughout the training process. Typically, the decrease in evaluators' losses precedes that of designers' losses due to the former's involvement in the backward pass of the designer. In essence, designers' performance is contingent on that of the evaluators. The MSE loss in the MR environment undergoes a more substantial change compared to the cross-entropy loss in the TE environment during training, primarily because the designer evolves the rules based solely on the win rate. Consequently, the rule evolution process can be considered a black-box optimization task. Increasing rule complexity exacerbates the challenge faced by designers in discerning the appropriate rule-generation strategies.

Comparing the performance of human and RGN designers, we established fixed game result expectations in the TE as the design criterion and evaluated their well-designed rules. As evident in Table II, the RGN designer consistently outperforms random generation and untrained models. However, human designers occasionally produce comparable results, and can even surpass the RGN designer, particularly when they iteratively test their rules. Increasing the cooperation rate logically entails a decrease in the presence of cheaters, randoms, and grudgers. Designers can also manipulate the payoff value to incentivize cooperation over cheating by offering more coins as rewards. A subtle alteration in the design requirement from 100% to 80% cooperator proportion at the game's conclusion distinguishes the last two sections in the table. This led to the RGN designer emerging victorious. We conclude that the AI designer excels when design requirements are not extreme, more abstract, or when the causality between input rules and output game results is obfuscated.

The performance of RL agents is evaluated in a fixed game rule, which was generated by the well-trained RGN designer. Fig. 12a illustrates the efficiency of these agents during the training phase. Notably, the categorical DQN emerges as the most efficient, followed by the double DQN, deep Q-learning, and Q-learning. However, this performance gap diminishes after 10 training epochs. After that, the agents exhibit comparable performance within the game. As evident from Fig. 12b, during the game testing phase, the agents' performance converges to a similar value. Because the game environment proposed in this paper involves only a few rules, even Q-learning agents can achieve reliable performance when testing the generated rules.

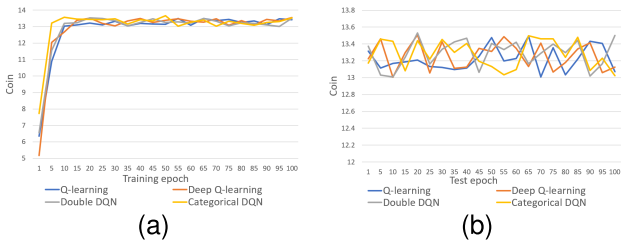



Fig. 12. Comparison of testers' coin changes in a well-designed MR environment. Four reinforcement learning algorithms participate in the comparison: Q-learning, deep Q-learning, double DQN, and categorical DQN. The first figure records the agents' coin numbers during the training, and the second figure records the testing stage. (a) Tester comparison in training. (b) Tester comparison in testing.

## V. Conclusion

This paper proposed an innovative rule generation framework RGN that leverages generative models, reinforcement learning models, and game environments to address the challenges of no dataset, rule translation, and unreasonable expectations in automated rule design, evaluation, and evolution, in accordance with controllable expectations. We initially refined and notated three fundamental elements of the rule generation task and established two digital environments, maze run and trust evolution, for implementation and demonstration. A well-trained rule designer can generate rules aligned with expectations, except in some unreasonable circumstances. By utilizing the environments, AI and NPC, as well as human players, can engage in the generated game rules. Moreover, we observed the rule evolution pattern during training and the competition among Q-learning agents within the game.

For future work, we intend to explore deeper into unreasonable expectations, which were discovered during the research. Unreasonable expectations, such as requiring two different personality groups to win at the same time in TE, can confuse RGN in both the training and evolving stages. It also leads to an unfair evaluation result during the test. We also aspire to investigate the causality between each dimension within the vector and its corresponding rule. Additionally, we are intrigued by the potential effects of sampling data from the environment and pretraining the designer to generate certain unique conditions. Moreover, we intend to scrutinize the relationships among different categories of rules, such as those pertaining to population dynamics and payoff distribution. By introducing some seemingly extraneous rules into the system, we aim to enhance our understanding of the causality at play.

## References

[1] K. S. Tekinbas and E. Zimmerman, *Rules of Play: Game Design Fundamentals*. Cambridge, MA, USA: MIT Press, 2003.

[2] M. A. Nowak, "Five rules for the evolution of cooperation," *Science*, vol. 314, no. 5805, pp. 1560–1563, Dec. 2006.

[3] B. Z. Tamanaha, *On the Rule of Law: History, Politics, Theory*. Cambridge, U.K.: Univ. Press, 2004.

[4] A. Liapis, G. N. Yannakakis, M. J. Nelson, M. Preuss, and R. Bidarra, "Orchestrating game generation," *IEEE Trans. Games*, vol. 11, no. 1, pp. 48–68, Mar. 2019.

[5] R. A. Posner, *Law, Pragmatism, and Democracy*. Cambridge, MA, USA: Harvard Univ. Press, 2009.

[6] Z. Wu, Y. Mao, and Q. Li, "Procedural game map generation using multi-leveled cellular automata by machine learning," in *Proc. 2nd Int. Symp. Artif. Intell. Med. Sci.*, Oct. 2021, pp. 168–172.

[7] S. W. Yackee, "The politics of rulemaking in the United States," *Annu. Rev. Political Sci.*, vol. 22, no. 1, pp. 37–55, May 2019.

[8] J. P. da Costa, C. Mouneyrac, M. Costa, A. C. Duarte, and T. Rocha-Santos, "The role of legislation, regulatory initiatives and guidelines on the control of plastic pollution," *Frontiers Environ. Sci.*, vol. 8, p. 104, Jul. 2020.

[9] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proc. IEEE Symp. Comput. Intell. Games*, Dec. 2008, pp. 111–118.

[10] A. Farizawani, M. Puteh, Y. Marina, and A. Rivaie, "A review of artificial neural network learning rule based on multiple variant of conjugate gradient approaches," *J. Phys., Conf.*, vol. 1529, no. 2, Apr. 2020, Art. no. 022040.

[11] M. Liu et al., "Reaction mechanism generator v3. 0: Advances in automatic mechanism generation," *J. Chem. Inf. Model.*, vol. 61, no. 6, pp. 2686–2696, 2021.

[12] F. Angaroni et al., "An optimal control framework for the automated design of personalized cancer treatments," *Frontiers Bioeng. Biotechnol.*, vol. 8, p. 523, May 2020.

[13] Z. Zeng, K. Ou, L. Wang, and Y. Yu, "Reliability-oriented automated design of double-sided cooling power module: A thermo-mechanical-coordinated and multi-objective-oriented optimization methodology," *IEEE Trans. Device Mater. Rel.*, vol. 20, no. 3, pp. 584–595, Sep. 2020.

[14] C. G. Langton, *Genetic Algorithms and Artificial Life*. Cambridge, MA, USA: MIT Press, 1997, pp. 267–289.

[15] S. G. Ficici and J. B. Pollack, "Challenges in coevolutionary learning: Arms-race dynamics," in *Proc. 6th Int. Conf. Artif. Life*, vol. 6. Cambridge, MA, USA: MIT Press, 1998, p. 238.

[16] H. Liu et al., "AudioLDM: Text-to-audio generation with latent diffusion models," *arXiv:2301.12503*, 2023.

[17] W. Aguilar, G. Santamaria-Bonfil, T. Froese, and C. Gershenson, "The past, present, and future of artificial life," *Frontiers Robot. AI*, vol. 1, p. 8, Oct. 2014.

[18] B. Chopard and M. Droz, "Cellular automata modeling," in *Cellular Automata Modeling of Physical Systems*. Cambridge, U.K.: Cambridge Univ. Press, 1998, pp. 21–65, doi: 10.1017/CBO9780511549755.003.

[19] N. S. Lachenmyer and S. Akasha, "An aquarium of machines: A physically realized artificial life simulation," *Proc. ACM Comput. Graph. Interact. Techn.*, vol. 5, no. 4, pp. 1–11, Sep. 2022.

[20] J. Suarez, Y. Du, P. Isola, and I. Mordatch, "Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents," 2019, *arXiv:1903.00784*.

[21] M. Stylianos, "Metaverse," *Encyclopedia*, vol. 2, no. 1, pp. 486–497, 2022.

[22] Z. Yang, B. Gong, L. Wang, W. Huang, D. Yu, and J. Luo, "A fast and accurate one-stage approach to visual grounding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4682–4692.

[23] P. Mirza-Babaei, R. Robinson, R. Mandryk, J. Pirker, C. Kang, and A. Fletcher, "Games and the metaverse," in *Proc. Extended Abstr. Annu. Symp. Comput.-Human Interact. Play*, 2022, pp. 318–319.

[24] E. Shin and J. H. Kim, "The metaverse and video games: Merging media to improve soft skills training," *J. Internet Comput. Services*, vol. 23, no. 1, pp. 69–76, 2022.

[25] A. Summerville et al., "Procedural content generation via machine learning (PCGML)," *IEEE Trans. Games*, vol. 10, no. 3, pp. 257–270, Sep. 2018.

[26] S. Risi and J. Togelius, "Increasing generality in machine learning through procedural content generation," 2019, *arXiv:1911.13071*.

[27] C. Schifter and M. Cipollone, "Minecraft as a teaching tool: One case study," in *Proc. Soc. Inf. Technol. Teacher Educ. Int. Conf.*, 2013, pp. 2951–2955.

[28] P. Mawhorter, "Efficiency, realism, and representation in generated content: A case study using family tree generation," in *Proc. 12th Int. Conf. Found. Digit. Games*, Aug. 2017, pp. 1–4.

[29] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, "General video game AI: A multitrack framework for evaluating agents, games, and content generation algorithms," *IEEE Trans. Games*, vol. 11, no. 3, pp. 195–214, Sep. 2019.

[30] A. Strong, "Applications of artificial intelligence & associated technologies," *Science*, vol. 5, no. 6, pp. 64–67, 2016.

[31] A. Oussidi and A. Elhassouny, "Deep generative models: Survey," in *Proc. Int. Conf. Intell. Syst. Comput. Vis. (ISCV)*, Apr. 2018, pp. 1–8.

[32] M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti, "A survey of unsupervised generative models for exploratory data analysis and representation learning," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–40, Jun. 2022.

[33] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *Stat*, vol. 1050, pp. 1–14, May 2014.

[34] X. Li, H. Zhang, and R. Zhang, "Adaptive graph auto-encoder for general data clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9725–9732, Dec. 2022.

[35] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[36] F. Zhou, S. Yang, H. Fujita, D. Chen, and C. Wen, "Deep learning fault diagnosis method based on global optimization GAN for unbalanced data," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104837.

[37] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 3617–3621.

[38] X. Sheng, L. Li, D. Liu, Z. Xiong, Z. Li, and F. Wu, "Deep-PCAC: An end-to-end deep lossy compression framework for point cloud attributes," *IEEE Trans. Multimedia*, vol. 24, pp. 2617–2632, 2022.

[39] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. G. I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[41] K. Zhang, Z. Yang, and T. Basar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*. New York, NY, USA: Springer, 2021, pp. 321–384.

[42] L. Yaeger, "Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or poly world: Life in a new context," MDPI, Basel, Switzerland, Tech. Rep. 5, 1994.

[43] B. Zhao, M. Ye, L. Stankovic, and V. Stankovic, "Non-intrusive load disaggregation solutions for very low-rate smart meter data," *Appl. Energy*, vol. 268, Jun. 2020, Art. no. 114949.

[44] P. Supramaniam, O. Ces, and A. Salehi-Reyhani, "Microfluidics for artificial life: Techniques for bottom-up synthetic biology," *Micromachines*, vol. 10, no. 5, p. 299, Apr. 2019.

[45] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[46] H. Ning et al., "A survey on the metaverse: The state-of-the-art, technologies, applications, and challenges," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14671–14688, Aug. 2023.

[47] S. Zhang, H. Peng, J. Fu, and J. Luo, "Learning 2D temporal adjacent networks for moment localization with natural language," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 12870–12877.

[48] S. Schwarz et al., "Emerging MPEG standards for point cloud compression," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[49] G. E. Raptis, C. Fidas, and N. Avouris, "Effects of mixed-reality on players' behaviour and immersion in a cultural tourism game: A cognitive processing perspective," *Int. J. Hum.-Comput. Stud.*, vol. 114, pp. 69–79, Jun. 2018.

[50] H. Ning, Z. Zhen, F. Shi, and M. Daneshmand, "A survey of identity modeling and identity addressing in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4697–4710, Jun. 2020.

[51] M. Guzdial and M. Riedl, "Automated game design via conceptual expansion," in *Proc. AAAI Conf. Artif. Intell. Interact. Digital Entertainment*, 2018, vol. 14, no. 1, pp. 31–37.

[52] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1, pp. 1–22, 2013.

[53] A. Summerville, J. Osborn, and M. Mateas, "CHARDA: Causal hybrid automata recovery via dynamic analysis," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 2800–2806.

[54] B. Pell, "Metagame in symmetric chess-like games," *Comput. Intell.*, vol. 12, no. 1, pp. 177–198, 1996, doi: 10.1111/j.1467-8640.1996.tb00258.x.

[55] V. Hom and J. Marks, "Automatic design of balanced board games," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2007, vol. 3, no. 1, pp. 25–30.

[56] C. Browne and F. Maire, "Evolutionary game design," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 1, pp. 1–16, Mar. 2010.

[57] M. Cook, S. Colton, A. Raad, and J. Gow, "Mechanic miner: Reflection-driven game mechanic discovery and level design," in *Applications of Evolutionary Computation*. Berlin, Germany: Springer, 2013, pp. 284–293.

[58] L. Chiariglione et al., "Towards a standard for human interaction with connected autonomous vehicles," in *Proc. 5th Int. Conf. Connected Auto. Driving (MetroCAD)*, Apr. 2022, pp. 63–71.

[59] A. Khalifa, M. C. Green, D. Perez-Liebana, and J. Togelius, "General video game rule generation," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2017, pp. 170–177.

[60] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation," in *Applications of Evolutionary Computation: EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC*. Istanbul, Turkey: Springer, 2010, pp. 141–150.

[61] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, pp. 201–215, Sep. 2011.

[62] N. Shaker, M. Shaker, and J. Togelius, "Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2013, vol. 9, no. 1, pp. 215–216.

[63] A. J. Summerville and M. Mateas, "Super Mario as a string: Platformer level generation via LSTMs," 2016, *arXiv:1603.00930*.

[64] R. Jain, A. Isaksen, C. Holmgård, and J. Togelius, "Autoencoders for level generation, repair, and recognition," in *Proc. ICCC Workshop Comput. Creativity Games*, vol. 9, 2016, pp. 1–9.

[65] A. Summerville, J. R. H. Mariño, S. Snodgrass, S. Ontañón, and L. H. S. Lelis, "Understanding mario: An evaluation of design metrics for platformers," in *Proc. 12th Int. Conf. Found. Digit. Games*, Aug. 2017, pp. 1–10.

[66] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," 2022, *arXiv:2204.06125*.

[67] C. Wang, W. Pedrycz, J. Yang, M. Zhou, and Z. Li, "Wavelet frame-based fuzzy C-means clustering for segmenting images on graphs," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3938–3949, Sep. 2020.

[68] C. Wang, W. Pedrycz, Z. Li, and M. Zhou, "Residual-driven fuzzy C-means clustering for image segmentation," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 4, pp. 876–889, Apr. 2021.

[69] C. Wang, W. Pedrycz, M. Zhou, and Z. Li, "Sparse regularization-based fuzzy C-means clustering incorporating morphological grayscale reconstruction and wavelet frames," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 7, pp. 1826–1840, Jul. 2021.

[70] C. Wang, Z. Yan, W. Pedrycz, M. Zhou, and Z. Li, "A weighted fidelity and regularization-based method for mixed or unknown noise removal from images on graphs," *IEEE Trans. Image Process.*, vol. 29, pp. 5229–5243, 2020.

[71] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," 2022, *arXiv:2204.03458*.

[72] Z. Chen et al., "InferGrad: Improving diffusion models for vocoder by considering inference in training," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 8432–8436.

[73] X. Zhang et al., "Learning causal representation for training cross-domain pose estimator via generative interventions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 11250–11260.

[74] W. Lin, H. Lan, H. Wang, and B. Li, "OrphicX: A causality-inspired latent variable model for interpreting graph neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 13719–13728.

[75] S. Li, X. Zhao, L. Stankovic, and D. Mandic, "Demystifying CNNs for images by matched filters," 2022, *arXiv:2210.08521*.

[76] R. Karp and Z. Swiderska-Chadaj, "Automatic generation of graphical game assets using GAN," in *Proc. 7th Int. Conf. Comput. Technol. Appl.*, Jul. 2021, pp. 7–12.

[77] X. Sheng, L. Xu, Y. Xu, D. Jiang, and B. Ren, "Semantic-preserving abstractive text summarization with Siamese generative adversarial net," in *Proc. Findings Assoc. Comput. Linguistics, NAACL*, 2022, pp. 2121–2132.

[78] W. Zhang, D. Zhou, and Q. Gu, "Reward-free model-based reinforcement learning with linear function approximation," in *Adv. Neural Inf. Process. Syst.*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran Associates, 2021, pp. 1582–1593. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/0cb929eae7a499e50248a3a78f7acfc7-Paper.pdf

[79] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, "Automatic data augmentation for generalization in reinforcement learning," in *Adv. Neural Inf. Process. Syst.*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran Associates, 2021, pp. 5402–5415. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/2b38c2df6a49b97f706ec9148ce48d86-Paper.pdf

[80] B. H. Zhang and T. Sandholm, "Finding and certifying (near-) optimal strategies in black-box extensive-form games," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 6, pp. 5779–5788.

[81] P. D. Straffin, *Game Theory and Strategy*, vol. 36. Providence, RI, USA: American Mathematical Society, 2023.

[82] J. Paavilainen, "Critical review on video game evaluation heuristics: Social games perspective," in *Proc. Int. Academic Conf. Future Game Design Technol.*, May 2010, pp. 56–65.

[83] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan, "Extensible markup language (XML)," *World Wide Web J.*, vol. 2, no. 4, pp. 27–66, 1997.

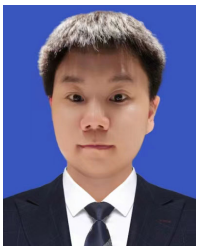
[84] J. Wu, J. B. Tenenbaum, and P. Kohli, "Neural scene de-rendering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7035–7043.

[85] X. Lu, J. Gonzalez, Z. Dai, and N. D. Lawrence, "Structured variationally auto-encoded optimization," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3267–3275.

[86] M. Cook, "Puck: A slow and personal automated game designer," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2022, vol. 18, no. 1, pp. 232–239.

**Jiyao Pu** received the bachelor's degree from the University of Electronic Science and Technology of China in 2016 and the M.Sc. degree from the School of Computing Science, The University of Newcastle, U.K., in 2020. He is currently pursuing the Ph.D. degree with the Hybrid Intelligence Laboratory, Department of Computer Science, Durham University. His research interests include computer vision, reinforcement learning, automated game design, and procedural content generation.

**Haoran Duan** (Graduate Student Member, IEEE) received the M.S. degree (Hons.) in data science from Newcastle University, U.K., in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Durham University. After that, he was a Research Student with the Open Laboratory, Newcastle University, where he is also a Research Associate with the Networked and Ubiquitous Systems Engineering Group, School of Computing, working on deep learning applications. His current research interests include the applications/theories of deep learning. He is an active reviewer of CVPR, ECCV, AAAI, BMVC, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON MULTIMEDIA, and Ubicomp.



**Junzhe Zhao** received the bachelor's degree from the Beijing Institute of Technology and the master's degree from Newcastle University, U.K. His previous research involved VR and panoramic video. His current research interests include large language model pre-training, SFT, and agent framework construction.

**Yang Long** (Senior Member, IEEE) is currently an Assistant Professor with the Department of Computer Science, Durham University. He is also an MRC Innovation Fellow aiming to design scalable AI solutions for large-scale healthcare applications. His research background is in the highly interdisciplinary field of computer vision and machine learning. While he is passionate about unveiling the black-box of AI brain and transferring the knowledge to seek scalable, intractable, interpretable, and sustainable solutions for other disciplinary researches, e.g., physical activity, mental health, design, education, security, and geoengineering. He has authored/coauthored more than 30 top-tier papers in refereed journals/conferences, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON IMAGE PROCESSING, CVPR, AAAI, and ACM MM, and holds a patent and a Chinese National Grant.