



# Deep learning applications in games: a survey from a data perspective

Zhipeng Hu<sup>1</sup> · Yu Ding<sup>2</sup> · Runze Wu<sup>2</sup> · Lincheng Li<sup>2</sup> · Rongsheng Zhang<sup>2</sup> · Yujing Hu<sup>2</sup> · Feng Qiu<sup>2</sup> ·  
Zhimeng Zhang<sup>2</sup> · Kai Wang<sup>2</sup> · Shiwei Zhao<sup>2</sup> · Yongqiang Zhang<sup>2</sup> · Ji Jiang<sup>2</sup> · Yadong Xi<sup>2</sup> · Jiashu Pu<sup>2</sup> ·  
Wei Zhang<sup>2</sup> · Suzhen Wang<sup>2</sup> · Ke Chen<sup>2</sup> · Tianze Zhou<sup>2</sup> · Jiarui Chen<sup>2</sup> · Yan Song<sup>2</sup> · Tangjie Lv<sup>2</sup> · Changjie Fan<sup>2</sup>

Accepted: 7 October 2023 / Published online: 4 December 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

## Abstract

This paper presents a comprehensive review of deep learning applications in the video game industry, focusing on how these techniques can be utilized in game development, experience, and operation. As relying on computation techniques, the game world can be viewed as an integration of various complex data. This examines the use of deep learning in processing various types of game data. The paper classifies the game data into asset data, interaction data, and player data, according to their utilization in game development, experience, and operation, respectively. Specifically, this paper discusses deep learning applications in generating asset data such as object images, 3D scenes, avatar models, and facial animations; enhancing interaction data through improved text-based conversations and decision-making behaviors; and analyzing player data for cheat detection and match-making purposes. Although this review may not cover all existing applications of deep learning, it aims to provide a thorough presentation of the current state of deep learning in the gaming industry and its potential to revolutionize game production by reducing costs and improving the overall player experience.

**Keywords** Deep learning · Interaction · Virtual character · Facial expression · Animation production · Intelligent NPCs · Data asset · Game industry · Image processing · Graphics · Game AI · Decision making · Natural language processing · User profile

## 1 Introduction

Video games can be seen as virtual worlds generated by computational devices (e.g. computers, mobile phones), providing players with perceptual stimuli and allowing them to manipulate elements of the virtual world and engage in gameplay. All of these stimuli and actions can be considered as various types of data, including visual objects, avatar appearance and animation, dialogue text, game plot, acting policy, user behaviors, and more.

Data plays a significant role in creating a game world. According to the different stages of game development, game experience, and game operation, this paper proposes to group data into asset data, interaction data, and player data.

- Asset data is responsible for constructing the overall appearance of the game world, which is produced during game development.
- Interaction data reflects continuous feedback from the game in response to player input, such as text-based conversation and decision-making behaviors (e.g. selecting specific attacking skills in multiplayer online battle arena (MOBA) game), which are built during game design and performed during game experience.
- Player data records each action of game players while exploring the game world, and can be used to estimate player profiles to provide effective and intelligent game operation for game fairness and balance.

Specifically, the asset data can involve all games, as all video-based games rely on art design to support visual performance. Interaction data is often used in RPG (Role Playing Game) and AVG (Adventure Game) games, since these games often use Non-player characters (NPCs) which are equipped with the capability of conversation. The behavioral aspect of interaction data is adopted in almost all games,

✉ Yu Ding  
dingyu01@corp.netease.com

Tangjie Lv  
hzltangjie@corp.netease.com

Extended author information available on the last page of the article

including board games, MOBA games, sports games (SG), first-person shooting (FPS) games, real-time strategy (RTS) games, battle royale (BR) games, massively multi-player online role-playing games (MMORPGs) and so on, which rely on game bots. Player data is involved almost all games due to their benefits in game operations. The above three types of data will be presented further below.

First, the asset data include the game scenarios, a great diversity of objects, virtual characters, and even animations shown to players, which are often created by graphic designers or artists. They are classified into art design assets, scene and avatar model assets, and animation assets. Specifically, the art design assets refer to the 2D appearance of objects including game character images, player clothing, and accessories, game scenes, game icons, etc. They need to be carefully designed by art designers, often involving a lot of labor. Similarly, the model and animation assets refer to the performance of 3D graphics information, which is often manually created by game artists to enhance the players' game experience, immersing them in a realistic game world.

Then, the interaction data refer to that players constantly interact with the game, e.g. text-based conversations and decision-making behaviors, as they explore the virtual world and advance the plot. For instance, certain non-player characters (NPCs) in MMORPGs are programmed to interact with players to advance the storyline, where the dialogue scripts are typically designed manually. Moreover, several types of games such as car-racing and football games involve the cooperation and competition between human players and game bots. The bots are expected to have interacting abilities and play as reliable teammates, loyal guardians, competitive opponents, or wise mentors to provide players with fruitful experiences. To satisfy these requirements, game bots are often implemented by manually incorporating a lot of human knowledge using programs like the finite-state machine and the behavior tree.

Besides, the player data refer to their operation and performance during the game playing. These data reflect the players' preferences, characteristics, and motivations. According to the player data, game producers are capable of providing personalized service and experience. For example, MMORPGs, of unique socio-economic characteristics, should meet the demands of different game players with various pursuits.

It is however not easy to make every player happy due to the players' diversity. To address this issue, player data, such as tabular profiles, behavior activities, and all kinds of relations, can help game producers set up various personalized services by inferring the preferences, characteristics, and motivations of each player. Knowing player preferences can help game producers design and recommend different types of gameplays for different players, such as competitive player-versus-player (PvP) battles and visual

and exploratory player-versus-environment (PvE) requests or puzzles. Figuring out player characteristics can facilitate making good matches where each player encounters reliable teammates, compatible co-players, and evenly-matched opponents. Understanding player motivations is very necessary for identifying and punishing dishonest players that gain illegal advantages via outside-game cheating tools. Usually, the various personalized services above heavily depend on hand-crafted feature engineering and<sup>1</sup> labor-intensive manual analysis.

As mentioned above, a colorful game world is composed of diverse kinds of game data. It is well known that the game design, development, and operation require a large amount of labor to create or process the above complex game data. Therefore, it is still a challenge to reduce game costs to create or benefit the game data while making a high-quality game experience.

Recently, the deep learning, as a branch of machine learning based on artificial neural networks, has been a prevalent method in the fields of computer vision, natural language processing, reinforcement learning, and so on, as it is capable of automatically learning complex patterns from a large amount of data. It can be used to perform various tasks, including image recognition, object detection, data analysis, image generation, 3D modeling, policy optimization, etc. Particularly, generative artificial intelligence (AI) techniques for artificial intelligence generated content (AIGC), like ChatGPT<sup>2</sup>, DALL·E 2 [108] and Codex<sup>3</sup>, have experienced explosive growth recently, which benefit creating high-quality digital content including images, music, natural language, and even 3D models.

As deep learning makes the process of content production more efficient and accessible, it has been introduced in game production to save a significant amount of manpower and resources. For example, in the game character creation stage, deep learning can be used to automatically create a specific character that looks similar to the input facial image of a player. Deep learning is capable of enhancing the vivacity and interactivity of NPCs in games by equipping them with conversational abilities to engage with players, which enhances the experience of the player. And by combining with decision-making techniques such as reinforcement learning [129] and Monte-Carlo tree search [14], deep learning can also help game developers improve the acting policies of NPCs, or rather game bots. The behaviors of the game bots can be more intelligent or more human-like by these techniques, and thus can bring players unparalleled interactive experiences. In addition, the powerful ability of the bots to execute actions and explore the game environment

<sup>1</sup> <https://www.midjourney.com/app/>

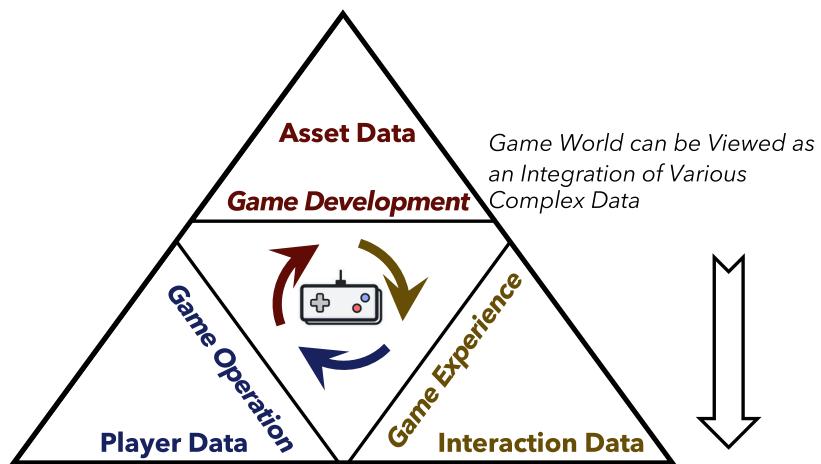
<sup>2</sup> <https://openai.com/blog/chatgpt>

<sup>3</sup> <https://openai.com/blog/openai-codex>

makes it possible to conduct automated game testing without human testers. This ensures a better game quality and provides players with a smoother gaming experience. Moreover, multi-source data, such as user profile, social network, and behavior sequence, can be effectively modeled by deep learning methods for better understanding and predicting future player activities like retention and payment trends.

This survey paper attempts to review existing deep learning applications in game development, game experience, and game operation. It may not include all the existing applications in the game industry, as other existing deep learning applications may not be published. The aim of this paper is to provide a comprehensive introduction for relevant researchers and promote the development of this field. In the

**Fig. 1** Categories of Game Data. This survey proposes to categorize game data into asset data, interaction data, and player data, which play crucial roles in game development, game experience, and game operation, respectively. Interestingly, the above three icons in this figure were automatically generated by Midjourney, according to the text input of game development, game experience, and game operation. The automatic generation of icons for games will be described in Section 3.1



<b>Deep Learning Applications in Games</b>	
	<b>From a Data Perspective</b>
	<b>1 Asset Data</b> <ul style="list-style-type: none"> <li>Art Design Assets</li> <li>Model Assets</li> <li>Animation Assets</li> </ul> <ul style="list-style-type: none"> <li>Text-To-Image Generation</li> <li>Scene Model Creation</li> <li>Avatar Model Creation</li> <li>Face Animation</li> <li>Body Animation</li> </ul>
	<b>2 Interaction Data</b> <ul style="list-style-type: none"> <li>Conversations</li> <li>Behaviors</li> </ul> <ul style="list-style-type: none"> <li>Open-Domain Conversation</li> <li>LLM-Powered Conversation</li> <li>Human-Like Agents</li> <li>Behavioral Diversity</li> <li>Game Testing</li> </ul>
	<b>3 Player Data</b> <ul style="list-style-type: none"> <li>Cheat Detection</li> <li>Match Making</li> </ul> <ul style="list-style-type: none"> <li>Behavior-Based Detection</li> <li>Trajectory-Based Detection</li> <li>Relationship-Based Detection</li> <li>Offline Model Training</li> <li>Online Planning Workflow</li> </ul>

following of this paper, Section 2 briefly reviews existing relevant surveys. Then, from the three aspects of data, this paper introduces deep learning applications in asset data (Section 3), interaction data (Section 4), and player data (Section 5). The asset data involve object image production, 3D scene and avatar models, and facial animation generation; the interaction data review conversation and decision-making behavior data; and the player data present cheat detection and matchmaking. Figure 1 illustrates the categories of game data and their related applications described in this survey.

## 2 Statement of other surveys

The above-mentioned deep learning technique can be viewed as an extension of artificial intelligence application within the gaming industry (Game AI). In fact, artificial intelligence has become a vast and diverse field in the processing of game data, encompassing a wide range of theoretical and practical techniques. It has facilitated the automation of various tasks involved in game development, game experience, and game operation.

A variety of survey articles about deep learning on games can be found [29, 34, 39, 54, 60, 87, 93, 110, 159, 163], which explore the application of artificial intelligence in games. Hooshyar et al. [54] systematically reviews the relevant articles on player modeling methods based on deep learning, which involve players' behavior modeling and goal recognition. In the book *Artificial Intelligence and Games*, Yannakakis et al. [159] provide a more comprehensive introduction and detail the important uses of AI in games, including procedural content generation (PCG), player modeling and game playing. This book not only provides a comprehensive overview of the entire field, but also points out the path of future development for game AI. It claims that more and more research will focus on the extensibility of AI roles within games and the generality across tasks, games and users. As the PCG has been popular in digital games for decades, Barbara et al. [29] give an overview of PCG for puzzles, which can improve gameplay by offering players interesting problems to solve. With the breakthrough of game AI like AlphaGo, OpenAI Five and AlphaStar, Yin et al. [163] survey AI applied in four typical types of human-computer games, i.e. board games, card games, first-person shooting (FPS) games and real-time strategy (RTS) games. Meanwhile, Justesen et al. [60] review recent deep learning techniques applied to video games and highlight the challenges. In addition, as game AI continues to rapidly evolve, Risi et al. [110] put their focus on exploring its applications beyond games, such as in robotics or the synthesis of chemicals.

However, these AI technologies or games are just a limited part of game AI as defined. Different from previous works,

this work refers to all contents and data in games as the game assets, including textures, scenes, character models, objects, scripts and even the profiles of players and the acting strategy of non-player characters. And the purpose of incorporating AI technologies in games is to streamline the process of generating game assets and improve the productivity, making it faster, more efficient and producing contents with higher quality.

Furthermore, due to the rapid development of deep learning technology, many new technologies are emerging constantly. As new models and techniques emerge, there have been new forms of AI applications in games. One notable example is the recently popular ChatGPT, which has exhibited great utility in creating intelligent NPCs. With its advanced natural language processing capabilities, ChatGPT can facilitate more realistic and engaging interactions between players and NPCs. Another example is a new class of generative models, called diffusion models [49]. It significantly enhances the text to image generation, and helps art designers improve efficiency in the production of game resources such as scene drawing, character drawing, and game icon creation. A diffusion model can generate data similar to ones they are trained on. Therefore, it can be used to produce high-quality images of various game scenes, like cities, forests, and buildings.

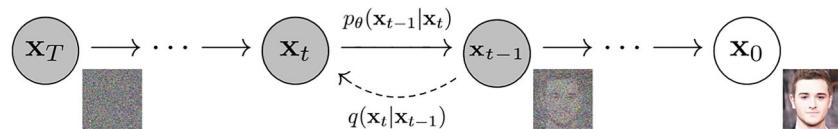
This paper will introduce various applications, the challenges and also the potential of the latest AI technologies in games from the perspective of game assets production.

## 3 Asset data

During the game development, a considerable amount of artistic cost is invested in producing various art assets, including art design, 3D scene and character models, and character animations. High-quality art assets are essential to providing players with visual engagement in the game. Improving the efficiency of art asset production has always been a concern for game industry professionals. In recent years, deep learning has made significant progress in art asset production, including icon generation, avatar face creation, and expression animation synthesis. Additionally, there are also some other applications that have not yet been widely used but have attracted a lot of research attention. The following will introduce the progress of deep learning in art asset production, including art asset production, 3D model production, and 3D animation synthesis.

### 3.1 Art design assets

During the process of game resource production, a range of artistic needs inevitably arise, encompassing concept illustration, scene design, character design, graphical user interface



**Fig. 2** Illustration of diffusion model in [50]. It consists of two Markov chain processes: the forward diffusion process (the input image  $x_0$  to the noise image  $X_T$ ) and the reverse diffusion process (the noise image  $X_T$  to the input image  $x_0$ )

(GUI) design, and so on. Nevertheless, there are two major challenges in game art design. Firstly, this process places high demands on the quantity and quality of artwork. For instance, with respect to icon design, hundreds of icons must be created each month, with an average of one day's labor required for each icon. Moreover, producing high-quality artwork - such as scene design, character design, or game illustration - often requires a significant amount of time (e.g., one week or more). Secondly, there is a substantial communication cost associated with the need for game planners and art designers to accurately convey painting requirements. This process often involves language communication, art draft creation, feedback, and revision, due to inconsistent understanding and expression. Overall, these pain points result in a significant increase in art design cost, accounting for nearly half of the game production cost, based on experience.

To reduce the cost of art design, the process of manual drawing can be first examined. Typically, based on their expertise, humans conceive ideas in their minds first and then translate those ideas into drawings. In essence, this process is to convert text (i.e., the ideas) into images (i.e., paintings).

Thanks to the advancements in pre-training technology, significant progress has been made in text-to-image generation research, such as DALL-E 2 [108] and Stable Diffusion [111]. These models are usually implemented using diffusion model [50], as shown in Fig. 2. This model consists of two Markov chain processes: the forward diffusion process and the reverse diffusion process. The diffusion process is a

$T$ -round Gaussian noise addition process that transforms the input image  $x_0$  into the noise image  $X_T$ , while the reverse diffusion process recovers the noise image  $X_T$  from the original image  $X_0$  by denoising network. Since the dimension of  $X_i$  and  $X_{i+1}$  stay invariant throughout each denoising process, the diffusion model can preserve the semantic structure of the data. Additionally, because the entire process involves random and step-by-step noise addition, the diffusion model has been shown to produce more diverse images and is not prone to mode collapse. However, due to the requirement that all Markov states be in memory for prediction at all times, diffusion models are computationally intensive, with training necessitating a significant amount of memory and resulting in a large carbon footprint.

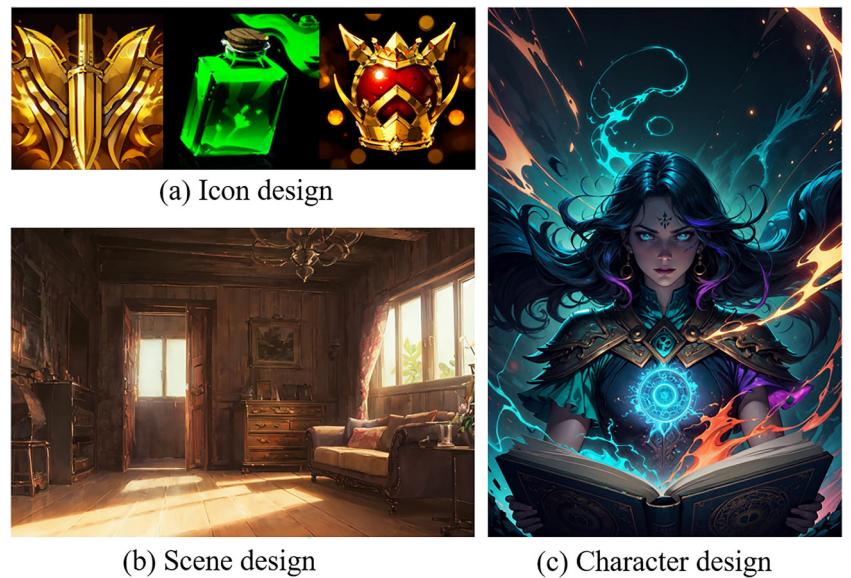
Fortunately, computing power has grown rapidly over the past few years, enabling diffusion models to generate higher quality images with larger model and image sizes, even reaching the level of industrial applications, as demonstrated in Fig. 3 shows<sup>4</sup>. In game art production, text-to-image generation technology can obviously help art designers improve the efficiency of game art pre-concept design or provide them inspiration. However, simple end-to-end text-to-image generation is insufficient for game art production. It lacks the necessary control needed to maintain consistent game style, unchanged game characters, fixed element placement, fixed pose, and other critical design factors. For instance, in character design, if art designers have already created several character drawings and intend to use text-to-image generation technology to generate additional drawings while preserving the character's style, specific models for the character must be learned from the existing drawings. Notably, recent researches have explored several approaches to enhance the controllability of text-to-image generation through two main approaches: (1) fine-tuning the model to maintain style or identity consistency, and (2) incorporating additional information inputs such as line drawings or depth maps to generate compliant images. We will elaborate on these approaches below.



**Fig. 3** Théâtre d'Opéra Spatial, won the 2022 Colorado State Fair's annual fine art competition

<sup>4</sup> Created by Jason M. Allen using the generative Artificial Intelligence platform Midjourney. The painting became a news story when it won the 2022 Colorado State Fair's annual fine art competition on 5 September, becoming one of the first AI generated images to win such a prize.

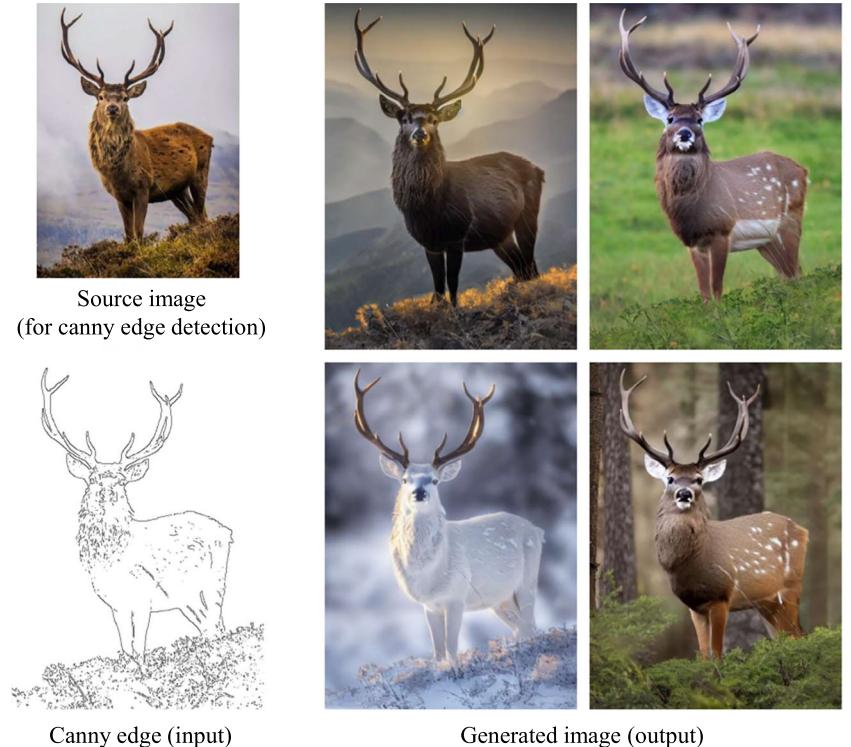
**Fig. 4** The generated examples of different art style (icon design, scene design, character design) models uploaded by designers in the Civitai website



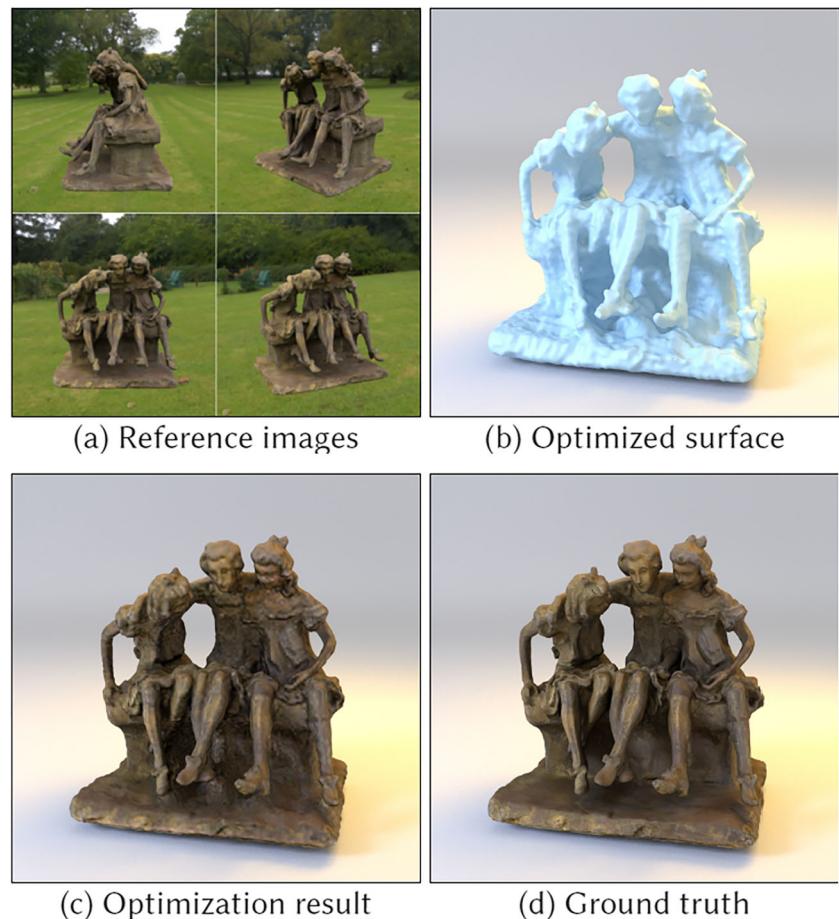
Firstly, to ensure consistent image style or identity during game art production, text-to-image fine-tuning techniques such as Dreambooth [112] and LORA [55] are useful. Specifically, Dreambooth has a stronger fitting ability for larger data volumes because it fine-tunes the entire network. Differently, LORA is suitable for fine-tuning on small datasets because it proposes to freeze original pre-trained model

and inject trainable layers (rank-decomposition matrices) in each transformer block. During fine-tuning, only the parameters of the injected layers are updated, which comprise a small percentage of the entire model and significantly reduce GPU memory requirements. For instance, in art design as illustrated in Fig. 4, we can use a limited number of existing images to fine-tune our own style model (e.g., for icon

**Fig. 5** The example figure in ControlNet [166]. Conditioned with the extracted canny edge (bottom left) from the source image (top left), four images with the deer in the same shape (right) in different scenes can be generated



**Fig. 6** Illustration of 3D reconstruction from calibrated multi-view images [141]. Taking a group of calibrated multi-view images (a) as input, the mesh (b) and texture (c) are reconstructed. The ground truth model is rendered in (d)



design, scene design, or character design) based on the base model, enabling us to generate more images that conform to the style of<sup>5</sup> the training data more quickly.

Secondly, to regulate the generation process, the widely used ControlNet [91, 166] proposes a method to enhance diffusion models by incorporating extra conditional inputs like sketches, edge maps, segmentation maps, pose keypoints, and so on. Specifically, ControlNet duplicates the weights of a large diffusion model into a “trainable copy” and a “locked copy”. The locked copy retains the network’s capacity learned on billions of images, while the trainable copy is trained on specific datasets (i.e., aligned triples  $\{text, image, conditional\ input\}$ ) to learn the conditional control. As depicted in Fig. 5, conditioned on the extracted canny edge of the source image, a deer with the same shape and in the same position can be generated. Overall, these AI techniques are shifting text-to-image generation from academic research to industrial production, and more art designers are learning to utilize AI tools (e.g., webUI<sup>6</sup>) to enhance the efficiency of game resource production.

Despite the progress made in AI painting for games, there are still some challenges. Firstly, original game paintings typically have high image resolution and detail requirements, so improving the quality of generated images remains a long-term exploration. Additionally, integrating AI painting deeply into the art production development process (e.g., developing new features, auxiliary art creation tools) is also a direction for future work.

### 3.2 Model assets

The creation of 3D models, including scene models and avatar models, is a crucial and time-consuming step in game development. With the fast development of deep learning, many model assets can be created automatically nowadays. Some methods take images as input and reconstruct the 3D object into game mesh. Other methods use text as convenient input to synthesize consistent game objects.

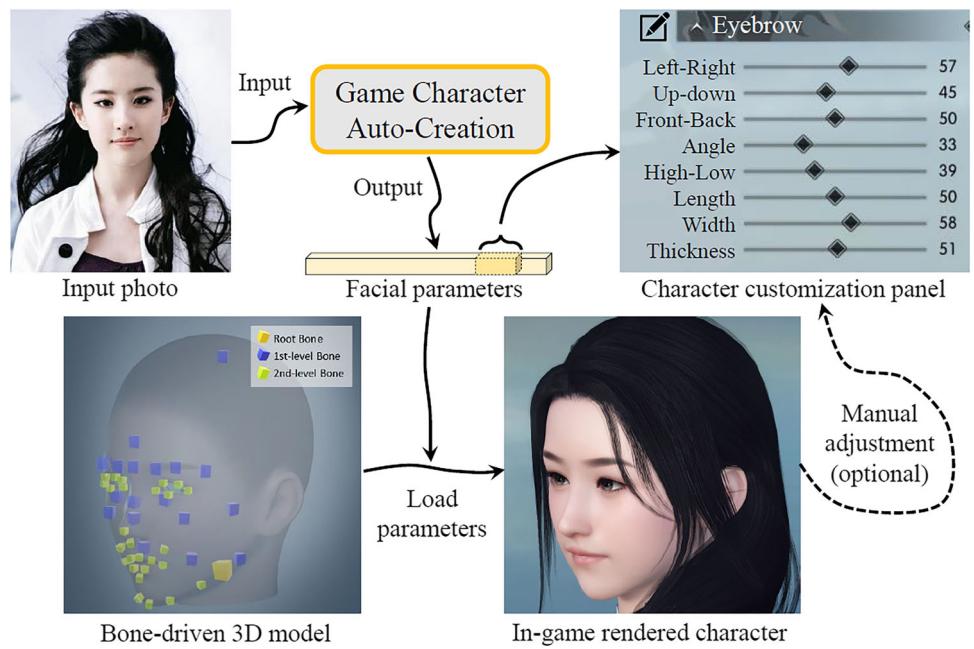
#### 3.2.1 Scene model creation

3D reconstruction is a fundamental task in 3D games, augmented reality (AR) and virtual reality (VR) applications.

<sup>5</sup> <https://civitai.com/>

<sup>6</sup> <https://github.com/AUTOMATIC1111/stable-diffusion-webui>

**Fig. 7** Illustration of the controller-based character auto-creation [118]. A single input face photo is first converted to a large set of physically meaningful facial parameters, then the parameters are applied on the controller-driven 3D model to generate the created game character head



Given calibrated multi-view images of an object, the goal is to reconstruct the mesh and texture of the object, which keeps the geometry of the object and can be rendered similar to the real one, as shown in Fig. 6.

3D reconstruction technique provides an efficient way for modeling artists and consumers to create 3D scenes based on reality or imagination, such as toys, furniture, buildings and rooms. 3D reconstruction was innovatively used in the game *The Vanishing of Ethan Carter* [104] in 2014. Then researchers from Electronic Arts<sup>7</sup> adopted photogrammetry in the game *Star Wars Battlefront* [10] in 2015. Since then, many high-quality 3D games adopted 3D reconstruction technique to create 3D meshes and textures, e.g. *Ride 3* [83]<sup>8</sup>, *Call of Duty: Modern Warfare* [150], *A Plague Tale: Innocence* [84]. Today 3D reconstruction technique from multi-view images has been well known in the game industry and has become a powerful tool for high-quality 3D games [127].

Traditional multi-view stereo (MVS) methods reconstruct 3D point clouds from predicted depth maps. As a typical Patchmatch-based MVS method, COLMAP [116] optimizes depth and normal maps with a graph model. Then a depth fusion step is processed to get a dense point cloud, finally, a meshing algorithm like screened Poisson Surface Reconstruction [63] is used to reconstruct the surface. Benefiting from deep learning, supervised MVS methods have also become popular. These methods show impressive perfor-

mance on multiple benchmarks [58, 65], but they have to be trained on specific datasets.

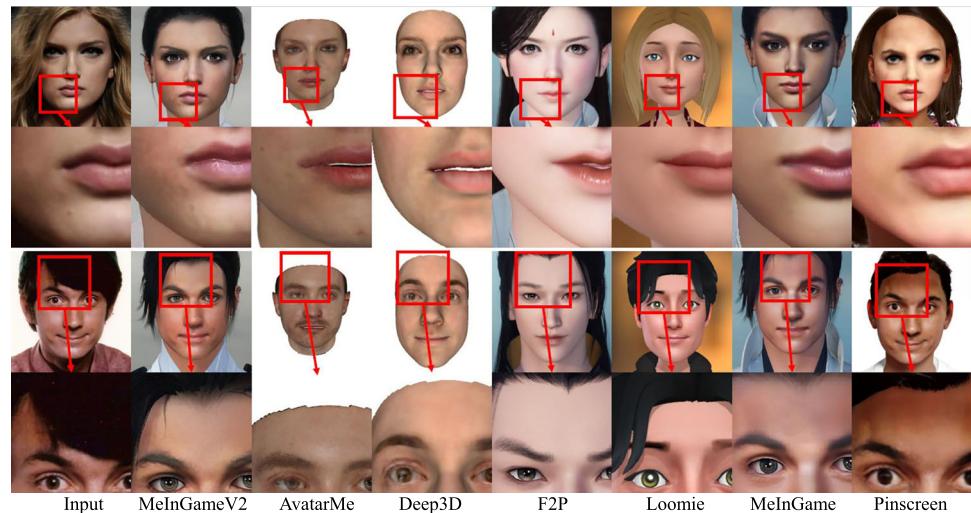
Recently, with the development of differentiable rendering, neural implicit surfaces have been introduced, which represent a surface as SDF [97] or an occupancy field by a neural network [86]. Moreover, these representations are combined with surface rendering or volume rendering. Surface-rendering methods assume that the color of a ray only relies on the color of the first intersection of the ray with the surface, which makes the gradient only back propagated to a local region near the intersection. Thus such methods are hard to handle with severe self-occlusions and sudden depth changes. As a famous neural volume-rendering method, NeRF [89] combines classical volumetric rendering with implicit function to render high-quality 2D images, but it can not export high-quality geometry. Recent works improve the geometric network and build connections between density-based representation and surface-based representation, which can extract more accurate and smooth surfaces.

We will furthermore review several works with more detail. Gu et al. [44] apply the cascade cost volume to the representative MVS-Net [160]. In this work, the single cost volume is decomposed into a cascade formulation of multiple stages, thus depth range and total number of hypothesis planes at each stage can be reduced. This coarse-to-fine structure remarkably decreases time cost and GPU memory consumption, and it also achieves the state-of-the-art performance on multiple benchmarks. Different from data-driven MVS-Nets, Yariv et al. [161] introduce the implicit differentiable renderer (IDR), which is an end-to-end self-supervised neural system. Although IDR can learn 3D surface, appear-

<sup>7</sup> [https://en.wikipedia.org/wiki/Electronic\\_Arts](https://en.wikipedia.org/wiki/Electronic_Arts)

<sup>8</sup> <https://blog.playstation.com/archive/2018/07/09/how-milestone-created-the-breathtaking-bikers-paradise-that-is-strada-della-forra-in-upcoming-ps4-racer-ride-3>

**Fig. 8** Results of the different mesh-based character auto-creation methods, including MeInGameV2 [79], AvatarMe [68], Deep3D [31], F2P [118], Loomie(<https://loomai.com>), MeInGame [78], Pinscreen(<https://pinscreen.com>), reported by [79]



ance, and cameras from posed images and noisy poses, it is heavily dependent on silhouette masks. Yariv et al. [162] further model the volume density as Laplace's cumulative distribution function (CDF) applied to a SDF representation. Benefiting from this simple but effective density representation, a bound on the opacity approximation error can be deduced, leading to more accurate sampling of the volume rendering integral. Moreover, the background is modeled using an additional NeRF network [165] to predict the point density and radiance field outside of the focused object. Thus it can reconstruct surfaces even without silhouette masks.

Although recent methods based on differentiable rendering have made great progress on 3D reconstruction from calibrated multi-view images, there are still several limitations. First, the computation of self-supervised methods is time-consuming. Second, pre-training general neural implicit surfaces across various scenes is still in research. Third, multi-view reconstruction of dynamic scenes is still an open problem. Finally, it is still a challenging task to model objects with strong reflection or refraction.

### 3.2.2 Avatar model creation

In online game applications, the character customization system provides an interactive way for users to specify the appearance of their avatars. The users are allowed to edit the facial identity of their character through some facial parameter controllers, e.g. the size of the eyes, and the shape of the eyebrows. However, manually editing hundreds of facial parameters is usually time-consuming for most users. It may take hours to manipulate the controllers to create an avatar face that looks like a specific person. As a result, for game applications, it is essential to automatically create an avatar face according to user-friendly hints, e.g. image or text.

Given an input facial image or text, the goal of the character auto-creation method is to generate a game face mesh, whose appearance is consistent with an input image or text, and has the same topology with the default game characters. In this way, the game face mesh can be embedded to the player-controlled character in the game engine. Some methods output a group of controller parameters as the structured representation of the 3D face (Fig. 7), while other methods directly modify the vertex positions and the texture map of the 3D face (Fig. 8).

As the first attempt, the character auto-creation method is proposed in the game "Justice Online". The players are allowed to specify the appearance of the character according to an input facial image. Later, many role-playing games (RPGs) integrate the character auto-creation method as part of the character customization system, supporting that the players efficiently create a character from a single input photo, and then control the character in the game with better immersion. Nowadays, the character auto-creation method has become a standard function for high-quality games, e.g. *Naraka Bladepoint*<sup>9</sup>, *Grand Theft Auto Online*<sup>10</sup>, *Loomie*<sup>11</sup>, *Pinscreen*<sup>12</sup>. Recently, the game *Justice Mobile*<sup>13</sup> first proposes to create characters from the input text.

As the general case of character auto-creation, early works perform 3D face reconstruction from 3D morphable face models (3DMM) [7, 8, 15]. Since the reconstructed 3DMM can not be directly applied on game characters, several approaches [118–120] propose to operate the predefined game character controllers. In this way, the created characters

<sup>9</sup> <http://www.narakathegame.com/>

<sup>10</sup> <http://www.rockstargames.com/GTAOnline>

<sup>11</sup> <https://loomai.com>

<sup>12</sup> <https://pinscreen.com>

<sup>13</sup> <https://h.163.com>

can be integrated in the game engine seamlessly. However, the controller-based methods have limited freedom of facial variation, and therefore can not deal with facial details such as wrinkles or freckles very well. Another group of works [68, 77, 78] directly operate the game mesh for more details. After the 3D face reconstruction in game character topology, some works copy the pixel colors of the input image as texture map, while others further remove the shadows and noise of the generated texture map to match the game style.

We will furthermore review several works with more detail. Shi et al. [118] first propose the controller-based character auto-creation method that allows users to upload a single face photo, and automatically generate the corresponding face controller parameters. The best parameters are found through iterative search. To make the parameter searching pipeline end-to-end differentiable, they train an imitator to imitate the rendering behavior of the game engine. As a result, the process is framed as a self-supervised learning paradigm. For better efficiency, their later works [119, 121] further replace the iterative searching strategy with the feed-forward prediction. Lin et al. [78] first manage to integrate the reconstructed 3D face into the game character topology and style. The 3DMM fitting is first carried out to provide the 3D face shape. Then, a shape transfer algorithm is proposed to transform the topology of a 3DMM mesh to game topology. Besides, a facial texture conversion method is designed to generate the game style texture that is similar to the input portrait but avoids the influence of lighting and occlusions. To preserve more geometry details, a later variant [81] proposes to reconstruct the 3D face in the position map representation, since the position map generally has higher spatial resolution. In addition, a new network architecture named affine convolution layer is designed to overcome the spatial misalignment between the natural view of the input image and the canonical view of the position map. Recently, [168] first propose to auto-create game character from the input text. It benefits from the power of large-scale pre-trained multi-modal CLIP [107], which is proposed by OpenAI. In addition to the continuous facial parameters, it also searches the non-differentiable discrete facial parameters in a unified framework. It first pre-trains a translator to translate the

CLIP image embeddings of random game characters to their facial parameters, then finetunes the translator on unseen text embeddings to predict continuous parameters. Besides, discrete parameters are further evolutionally searched. Some results are shown in Fig. 9.

While many identity similarity metrics have been proposed, they are all indirect metrics for the identity distance evaluation between two images. These metrics are not always consistent with human subjective feelings. It is still challenging to define a better identity similarity metric, especially when the game style is not realistic. Besides, how to adapt the expression controllers onto the customized game characters is also a problem to be solved.

### 3.3 Animation assets

In game production, animation assets play essential roles, including facial and body animations. They usually appear in game plot animation and player-character interaction, to promote the development of game storyline and create a plausible virtual character. Deep learning networks have been widely used in facial animation production, in particular speech-driven facial animation generation. The networks have the benefits of learning sophisticated cross-modal mapping from audio signals to facial movements, thus can produce believable facial animations in practical game production. However, although numerous research papers have reported promising progress on body animations, it remains challenging to automatically generate body animations using deep learning methods for game production. This difficulty can be attributed to the lack of sufficient high-quality body data. In this section, we mainly detail the audio-driven facial animation generation, besides, we briefly review some works of body animation generation.

**Face animation** Facial animations contain verbal and nonverbal behaviors [36, 70]. Game characters utilize lip movements, as verbal animations, to convey speech content; they also utilize eyebrow motions, as nonverbal animations, to repeat, emphasize and substitute spoken content. The combination of verbal and nonverbal behaviors in facial animations makes game characters believable and thus improves the quality of game production.

**Fig. 9** Results of text-driven character auto-creation [168]. The 3D game characters are generated from the corresponding input text



While game developers have made great efforts on synthesizing expressive facial animations, the efficiency is improved greatly in recent years, benefiting from the development of deep learning algorithm. Traditional pipelines of facial animation generation are time-consuming, labor-intensive, and expensive [72]. In the early years, animators need to set a facial expression in each keyframe by manually tuning parameters to combine blendshapes (i.e. basic facial expressions). This hand-crafted based method makes it impossible to produce large amounts of animations. Recently, professional systems of facial motion capture are proposed to generate facial animations according to actor (or actress) performance. Specifically, animators utilize professional helmet and softwares to record facial videos of one actor and manually retarget expressions into game characters. However, professional systems are too expensive and still time-consuming to produce large amounts of facial animations in game production.

In recent years, with the development of deep learning algorithms, it is possible for researchers to automatically synthesize realistic facial animations from the driving audio as input signals[32, 52, 74, 114, 115]. In these works, deep neural networks are leveraged to learn complex mapping from simultaneous audio to facial expressions. Deep networks produce impressive animations with low cost and short time, thus greatly improving the efficiency of game production.

Deep learning based methods mainly rely on three kinds of frameworks, including CNN-based framework [26, 52, 62, 74, 109, 135], LSTM-based framework [103, 114, 115, 171] and Transformer-based framework [38, 157], to model cross-modal correlations between audio and facial animations. These works confirm the verification of the above three basic frameworks.

In the CNN-based works, Taylor et al. [135] and Li et al. [74] utilize speech phonemes to reduce the timbre gap between different subjects. Their method is proven to be effective and generalized for subjects. Cudeiro et al. [26] and Karras et al. [62] further prove the generalization of deep speech feature [45] in both speech timbre and cross-lingual. In the LSTM-based works, [171] synthesizes animations with different speaking styles and Sadoughi and Busso [115] add extra generative adversarial network (GAN) loss [25]. In transformer-based works, [38] first utilize Transformer [140] to capture long-time dependencies in animation synthesis. Xing et al. [157] add a discrete motion prior to improve the realism of synthetic animations.

The animation generation is essentially a regression problem, which leans the mapping from audio features to animation parameters. Therefore, regression losses, including L1 loss and L2 loss, are always taken to guide the training of the above various frameworks [26, 38, 62, 76, 103, 109, 135, 157, 171], particularly for lip movements. Additionally, to improve the realism of synthetic animations, a GAN loss

[42] is utilized to restrict animations to have a similar performance to humans. To improve the smoothness of synthetic animations, temporal difference loss [62, 171] is proposed to reduce jitters between frames.

In the academic community, researchers focus on smoothness, realism, emotional richness, and diversified styles in facial animation synthesis. However, in the practical applications of game production, some extra game-specific views need to be considered, including disentangled animation generation and cross-character animation generation.

Disentangled animation generation of facial expression requires a method to synthesize animations of the lower face (mouth) and upper face (eyebrow and eyelid), respectively. In facial expression, different facial regions have different motion patterns. For instance, mouth movements have high frequency and depend on speech content. Upper face movements have more close correlations with speech rhythm. In the academic papers [26, 38, 62, 109, 157], they usually utilize principal-component-analysis-based (PCA-based) basis or vertex displacements to control the expression of 3D facial model. These controllers involve the full face region and have no semantic information about facial regions, so they do not disentangle the lower face and upper face. In game characters, facial expressions are controlled by interpretable facial blendshape [72]. Each channel of blendshape controls semantic local facial movement. Li et al. [74] disentangle the animation synthesis in game production and utilize different networks to model different patterns in the lower face and upper face. For lower face animation, they use a residual neural 1D network to predict mouth movements from speech phoneme features. In upper face animation, they use an encoder-decoder 1D network to predict eyebrow motions from time-aligned word. With the help of disentangled generation, their method is used in games *Justice Online* and *Revelation Online*<sup>14</sup>.

Cross-character animation generation requires the method to synthesize animations that can be shared for different game characters at a low cost. In the academic papers [26, 38, 109, 157], different characters have the same PCA-based expression controller and they usually carry experiments on the same recorded dataset[26]. However, in game production, characters always have character-specific blend shape with different numbers and different semantics. It is very expensive and time-consuming to recording training data specific to each character. Edwards et al. [35] solve this problem by hand-crafted expression retargeting and rule-based phoneme-to-viseme mapping. Specifically, they first design several rules to model the mapping between speech phoneme and viseme label according to mouth co-articulation. Then, to realize viseme retargeting, they manually sculpt character-specific viseme shapes for each new game character. With

<sup>14</sup> <https://ro.my.games/>

**Fig. 10** Applications of speech/text driven facial animation synthesis in game production: (a) “Justice Online” and “Revelation Online” (b) “Cyberpunk 2077”



the help of the above two steps, their method is widely used in the game “Cyberpunk 2077”. Figure 10 shows the above methods in game production.

**Body animation** It is a challenging problem in the game industry to produce realistic body movements of virtual characters. Recovering the actions performed by subjects is the most common approach to producing skeleton animations. Motion capture (Mocap) is the typical technique in animation production, which inevitably contains positional noise and tracking errors. Therefore some works [17, 51] manage to denoise the markers with deep learning for better motion capture performance. Besides that, recent advances in deep learning have attempted to simplify the process of reconstructing detailed body poses and movements from videos directly, which has the potential to automatically extract body animations from performance video. In fact, directly copying the rotation value of joints will cause sliding and clipping due to the large discrepancy in bone size. Moreover, character retargeting has been still a challenge in body animation.

Another approach for dynamic action production is to synthesize controllable motion for a virtual character from weak control signals, such as desired path, joint positions and kinematic velocity [126, 147]. The published attempts include stylized motion prediction, music-to-dance generation, text-to-motion generation and so on [28, 80, 106].

Compared with facial expression animation, the development of body animation has been relatively slow, largely due to the difficulty of obtaining body motion data and the complexity of its diversity.

## 4 Interaction data

In game playing, a significant characteristic is the interaction between user behavior and the game environment, which is also different from other video content products. While players employ their individual behaviors or strategies as inputs to the game environment, the game environments display corresponding feedback for players and further achieve user engagement. Along with the development of the game industry, player behaviors have been allowed to have diverse inputs. For example, players can interact with NPCs by entering speech text or manipulating behaviors of their characters. They also expect real-time and believable feedback with conversations or behaviors from NPCs, where the rule-based

game feedback cannot meet the demand for engagement. In recent years, deep learning has also been introduced to enhance the interactivity of games. Specifically, players can freely chat with NPCs, and NPCs will give different responses according to their individual profiles and player inputs. In the aspect of behaviors, NPCs or game bots can rely on decision-making techniques to observe the players’ status and actions, and take the most appropriate responding actions for producing plausible interaction (e.g. a support-hero bot heals the hero of a player which has low health points in MOBA games). Furthermore, due to the autonomous acting and learning capabilities endowed by the decision-making techniques, the bots for automatic game testing can be far superior to script-based bots in terms of efficiency and cost.

### 4.1 Conversations

In MMORPGs, NPCs play various roles, such as advancing the game’s storyline, engaging in battles, and offering trade goods, which greatly enrich the game’s world. However, their interactions with players are limited by pre-programmed behaviors and text scripts designed by game developers, resulting in a lack of deep engagement. It is widely recognized that conversation is a natural way for players to interact with each other in games, allowing them to sense each other’s personality and emotions. Therefore, enabling players to engage in free-form conversations with NPCs can greatly improve their in-game immersion experience and make their interactions more engaging [130].

To achieve this, conversational AI techniques can be utilized to equip NPCs with the ability to chat with players and enhance their human-like qualities. Additionally, by programming NPCs with unique personas, they can offer players diverse responses when presented with the same input. This diversity can significantly improve the player’s overall experience and make the game more interesting. For instance, the NVIDIA Avatar Cloud Engine (ACE) for Games aims to revolutionize games by introducing intelligence to NPCs through AI-powered natural language interactions. As the Kairos demo<sup>15</sup> shown in Fig. 11, the player can chat with the NPC Jin, who runs a ramen shop. With the help of conversational AI, Jin responds to natural language queries realistically and consistently with the narrative backstory.

<sup>15</sup> <https://www.youtube.com/watch?v=5R8xZb6J3r0>



**Player:** Hey Jin, how are you?  
**Jin:** Unfortunately not so good.  
**Player:** How come?  
**Jin:** I am worried about the crime around here. It's gotten bad lately. My ramen shop got caught in the crossfire.  
**Player:** Can I help?  
**Jin:** If you want to do something about this, I have heard rumors that the powerful crime lord Kumon Aoki is causing all kinds of chaos in the city. He may be the root of this violence.  
**Player:** I'll talk to him, where can I find him?  
**Jin:** I have heard he hangs out in the underground fight clubs on the city's east side. Try there.  
**Player:** OK, I'll go.

**Fig. 11** The [Kairos](#) demo powered by NVIDIA ACE. In this demonstration, the non-playable character (NPC) named *Jin* is able to freely engage in conversation with players using conversational AI techniques

Moreover, Character AI<sup>16</sup> provides users with the ability to define a persona and chat with the agent, which has high potential for customizing NPCs within games.

Conversational AI is often denoted as a function  $P$  that maps a given post sentence  $X = \{x_1, x_2, \dots, x_i\}$  to an appropriate response sentence  $Y = \{y_1, y_2, \dots, y_j\}$ , i.e.,  $P(Y | X)$ , where  $x_i$  is the  $i$ -th token of the post sentence and  $y_j$  is the  $j$ -th token of the response sentence, *token* means the basic unit (e.g., *art*, *ificial*, *intelligence*) after cutting a sentence (e.g., *artificial intelligence*). If a conversational AI system has additional information input (e.g., persona, knowledge graphs, and dialogue context) denoted as  $K$  to generate a response, the mapping can be denoted as  $P(Y | X, K)$ .

Existing conversational AI can be broadly categorized into task-oriented and open-domain conversational. The task-oriented conversation aims to assist users in solving specific problems, such as booking restaurants or flights [16]. Open-domain conversation can be further divided into retrieval-based [59] and generation-based [142] ones. Retrieval-based conversation selects a response that best matches the user's input from a set of candidate responses, while generation-based conversation generates responses based on the dialogue context. While retrieval-based conversation only selects responses from the existing corpus, generation-based

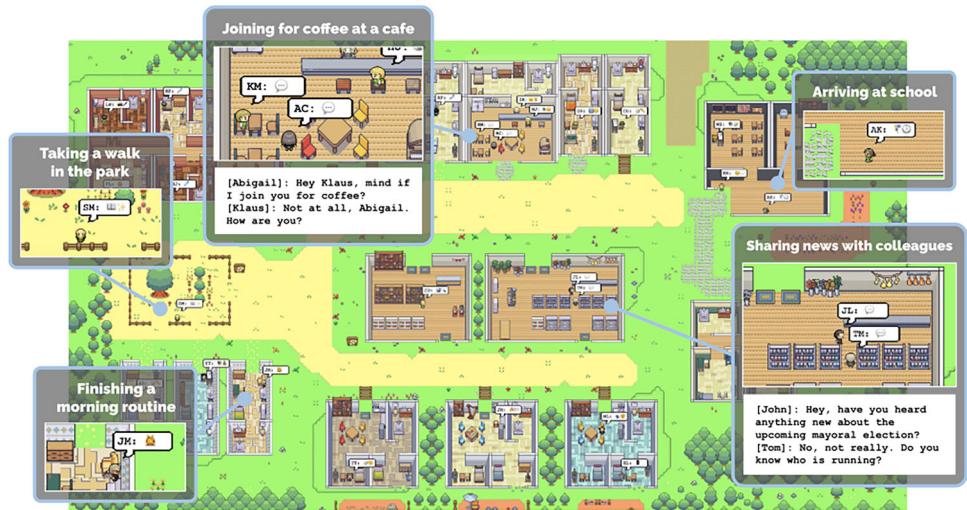
conversational AI can create new content with greater diversity, which is also more conducive to providing a higher degree of freedom in-game scenarios. Generation-based conversation typically employs an encoder-decoder architecture, which can be implemented using various networks such as Transformers [140], RNNs [117], etc. The encoder encodes the context and additional information, while the decoder generates responses that are returned to the user. However, in the early years, generation-based conversation suffers from poor contextual relevance and consistency due to limited training data and model size.

Fortunately, the performance of generation-based conversation systems has significantly improved due to the rapid development of auto-regressive pre-trained models. With a large number of parameters and massive training text data, these models can greatly improve the contextual relevance and consistency of the generated text. DialogGPT [167] was the first to fine-tune GPT models with 147 million dialogue instances and achieved a better conversational experience. Since then, it has become popular to improve conversational AI by fine-tuning pre-trained models with larger scale parameters over a wider range of dialogue corpus, such as Meena (867M pairs, 2.6B parameters) [1], PLATO-XL (2B pairs, 11B parameter) [4], LaMDA (1.56T words of public dialog data and web text, 137B parameters)[137], BlenderBot (175B parameters)[122]. These models demonstrate better context correlation, context consistency, and diversity of responses, leading to more natural and human-like NPCs and enriching the overall gaming experience for players.

Finally, it is worth noting that the emergence of ChatGPT or GPT-4 has recently sparked a surge in generative AI innovation. By fine-tuning large-scale pre-trained models (e.g., GPT-3 [13], LLaMA [139]) with chat-style data, ChatGPT-like models can act as human assistants and complete various requests from users, such as knowledge Q&A, event planning, and role-playing. This not only improves the effectiveness of conversational AI but also demonstrates the potential of integrating ChatGPT-like models to explore creative and novel features within games. One such example is the recent work [98] illustrated in Fig. 12, which investigates how generative agents can create believable simulacra of human behavior for interactive applications using GPT-4. The work demonstrates generative agents in a sandbox environment where twenty-five agents engage in their daily lives, including planning their schedules, sharing news, building relationships, and coordinating group activities. Players can also observe and intervene in the agents' lives as they see fit. Moreover, VOYAGER [145], shown in Fig. 13, is the first large-language-model-powered (LLM-powered) embodied lifelong learning agent in Minecraft that continuously explores the game world, acquires diverse skills, and makes novel discoveries without human intervention. It explores the use of black-box queries to enable VOY-

<sup>16</sup> <https://beta.character.ai/>

**Fig. 12** The illustration in [98]. It demonstrates generative agents by populating a sandbox environment with twenty-five agents



AGER to interact with GPT-4 without the need for fine-tuning model parameters. By leveraging this approach, VOYAGER develops skills that are temporally extended, interpretable, and compositional, leading to a rapid increase in its overall abilities and mitigating the risk of catastrophic forgetting. Empirically, VOYAGER exhibits strong lifelong learning capabilities within the context and performs exceptionally well in Minecraft gameplay.

Despite the significant advancements in conversational AI in recent years, there are still some challenges to be addressed. The first challenge is implementing safety mechanisms for filtering out inappropriate dialogue text. Pre-trained models are trained on a diverse range of Internet corpus and may generate text containing sensitive topics such as race, pornography, and politics. The second challenge is continuous learning. Conversational AI expects to continuously learn from users' feedback to improve their chat capabilities, which is also an important direction to enhance user experience.

## 4.2 Behaviors

Apart from elements such as graphics, gameplay, and storylines, intelligent agents (sometimes are also called intelligent bots, game AI bots, or AI bots) also play a significant role in shaping the players' game experience. Controlled by certain programs, these agents have action capabilities and internal behavioral goals, and can interact with players, generate various game events, and influence the entire game world. For example, in sports games such as *FIFA*<sup>17</sup> and *NBA 2K*<sup>18</sup>, the player's teammates (controlled by internal programs) need to coordinate with the player-controlled character. In single-

player games such as *Elden Ring*<sup>19</sup>, and *God of War*<sup>20</sup>, the minions and bosses constantly challenge the player and provide very exciting battle experience. In *GTAs*<sup>21</sup>, hundreds of NPCs can interact with the player-controlled character and generate various interesting events. Even in multiplayer online games, intelligent agents or game AI bots still exist. For example, in battle royale games like *PUBG*<sup>22</sup>, the agents are often used to supplement the number of players in case that there are not enough human players in a game. In MMORPGs, players are provided with away-from-keyboard service for their characters.

This section focuses on the subset of AI techniques for creating intelligent agents, namely the decision-making techniques, and how they can be utilized in the video game domain. Generally, four major techniques are commonly used for building intelligent agents currently, which are

- **Reinforcement Learning (RL):** a learning technique that enables an agent to learn playing games from the interactions between the game environment by trial-and-error [129].
- **Imitation Learning (IL):** a technique that can help an agent imitate the behaviors of expert game players, and is almost equal to supervised learning [57].
- **Evolutionary Algorithm (EA):** an optimization algorithm based on biological evolution principles, which can produce game playing agents by evolving a population of them according to a certain evaluation metric generation by generation [151].
- **Game Theory:** a mathematical theory that studies the optimal strategy of multiple agents in situations of com-

<sup>17</sup> [https://en.wikipedia.org/wiki/FIFA\\_\(video\\_game\\_series\)](https://en.wikipedia.org/wiki/FIFA_(video_game_series))

<sup>18</sup> [https://en.wikipedia.org/wiki/NBA\\_2K](https://en.wikipedia.org/wiki/NBA_2K)

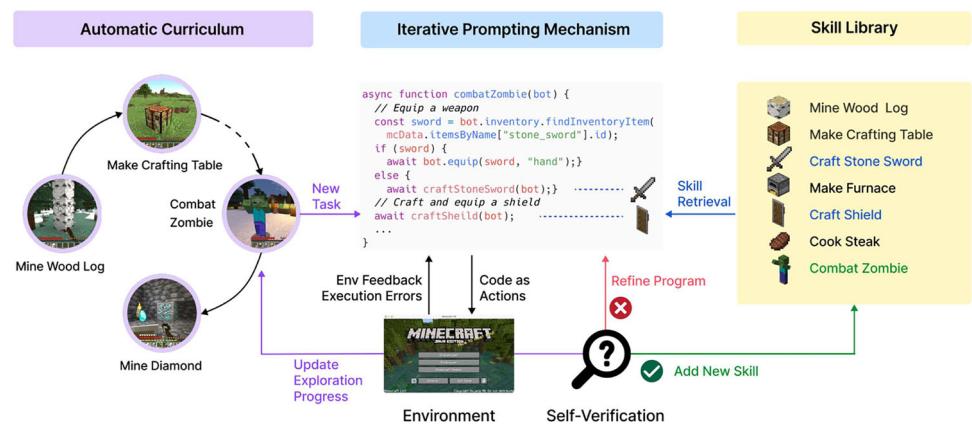
<sup>19</sup> [https://en.wikipedia.org/wiki/Elden\\_Ring](https://en.wikipedia.org/wiki/Elden_Ring)

<sup>20</sup> [https://en.wikipedia.org/wiki/God\\_of\\_War\\_\(franchise\)](https://en.wikipedia.org/wiki/God_of_War_(franchise))

<sup>21</sup> [https://en.wikipedia.org/wiki/Grand\\_Theft\\_Auto\\_V](https://en.wikipedia.org/wiki/Grand_Theft_Auto_V)

<sup>22</sup> [https://en.wikipedia.org/wiki/PUBG:\\_Battlegrounds](https://en.wikipedia.org/wiki/PUBG:_Battlegrounds)

**Fig. 13** The figure in [145] demonstrates VOYAGER. It comprises three essential elements: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space. Please zoom in for details



petition, cooperation, or mixture of both. The core idea of game theory is to simulate the decision-making process and results among all parties of agents in order to seek the optimal strategy for each party [144].

Each of these techniques benefits the domain from the perspectives of sequential decision-making, data fitting, optimization, problem modeling and solving, respectively. The following example of MOBA game agents can be used to give an intuitive understanding of them. The game playing of an agent (called hero) in MOBA games contains thousands of actions taken in order along the game, which is a typical sequential decision-making problem and can be solved by reinforcement learning. As an alternative, imitation learning can be utilized to train more human-like agents if human players' behavior data is available. Evolutionary algorithms are able to produce a group of agents with different playing strategies or behavior styles, so that can meet different human players' satisfaction. Finally, since MOBA games involve teamwork and competition, the mathematical models and solution concepts from game theory are necessary for modeling the agents' cooperative interactions within a team and competitive interactions between teams.

The decision-making techniques introduced above can all effectively improve the skill level of intelligent agents, even making them more skillful than human experts. Well-known milestones achieved by these techniques and their combinations include the superhuman agents for board games such as the AlphaGo family for *Go* [123, 124], Libratus [11] and Pluribus [12] for *Texas Hold'em*, Suphx for *Mahjong* [73], DeepNash for *Stratego* [102], and those for complex video games, such as the OpenAI Five agent for *Dota2* [6], the GT Sophy agent for *Gran Turismo* [156], and the AlphaStar agent for the full game of *StarCraft II* [143]. However, while a lot of academic research works are motivated by producing very skillful agents, overly powerful agents may not meet the actual needs of game players in game industry. For players, what they care about is having a very good

experience when playing the game, which generally contains the entertainment-level experience and the quality-level experience. The entertainment-level experience of a game is about whether players can get fun from it, and the quality-level experience refers to the objective quality of the game itself, such as crash rate, game balance, and game safety.

The intelligent agents in games can also influence the above two levels of gaming experience by playing the role of fun makers and game testing workers, respectively. Firstly, from the perspective that agents are entities which provide fun of interacting, the players' enjoyment and satisfaction highly depend on whether the agents exhibit human-like behaviors and diverse behavioral patterns. The agents with high degree of human-likeness and behavioral diversity can offer players more engaged and comprehensive feeling of interactions, such as a greater sense of achievement, and a continuous feeling of freshness. Secondly, from the perspective that agents are tools which enable autonomous decision-making and action execution, they can help game developers conduct automatic game testing and guarantee the quality of games for players. Since intelligent agents can possibly perform better than normal players in games, using such never-stopping workers for game testing can largely reduce the workload of game developers and may discover relatively hidden game defects. The following three subsections will introduce how the modern decision-making techniques can be adopted for generating agents that can improve both the entertainment-level and quality-level player experience, including human-like agents, agents with behavioral diversity, and game testing agents.

It should be pointed out that an agent for a specific demand usually cannot be built solely by any single technique. Researchers or developers often need to integrate various techniques to develop intelligent agents for games. For example, while RL is good at improving the skill level of agents, it is difficult for RL to optimize agents' behaviors towards some subjective goals, such as human-likeness. Imitation learning can be used to learn more human-like agents by fitting

the behavior distribution from human demonstrations, but often cannot create a very skillful agent due to unbalanced data and accumulative errors. Normally, simply using RL or IL can only generate one type of agent at a time, whereas evolutionary algorithms are a more general paradigm for generating multiple intelligent agents simultaneously. However, evolutionary algorithms still require reinforcement learning or other techniques as their basic optimizer. For game tasks with the existence of multiple agents, these techniques should often rely on the methods and solution concepts from game theory to deal with the interactions and relations between agents.

#### 4.2.1 Human-like agents

Intelligent agents usually act as players' partners, opponents, or even substitutes in games. So being human-like is a very natural and necessary requirement. Both games and game players can benefit from intelligent agents that act similarly or equally to human players. One typical example is battle royale games, such as *PlayerUnknown's Battlegrounds (PUBG)* and *Naraka Bladepoint*. A battle royale game involves dozens of players who start with minimal equipment and then must defeat all opponents while avoiding being trapped outside a shrinking safe zone. The major fun of battle royale games lies in the interesting or surprising events generated from the interactions between a large number of players. However, in the absence of enough human players during some specific periods (e.g., late at night), the game queuing time can be significantly longer and the players' gaming experience will become extremely poor. In this case, filling a game with human-like agents can shorten unnecessary queuing time without being noticed by players, and therefore retain the entertainment of the game. A second example where human-like agents can help is MOBA games, which require players to cooperate with different types of teammates to defeat the enemy team. A common issue occurring in MOBA games is that some teammates may quit the game early due to network disconnection or other factors. Without control, the unmovable characters of the quitting players may lead to the loss of the game. In such situations, human-like agents are able to take over the characters and keep the game going on without being discovered.

Recently, researchers have made efforts to study how to create human-like agents from human data. Imitation learning, which can directly learn how human players play games, is the simplest method for achieving this goal. As a type of supervised learning, imitation learning can be conducted using any possible learning model (e.g., neural networks, decision trees), with either hand-crafted input features [43, 164] or in an end-to-end learning manner [19, 46]. For example, Pelling and Gardner [100] developed two imitation learning agents for AI bots in the FPS game *Unreal*

*Tournament 2004* based on support vector machine (SVM) and probabilistic models [100]. And both agents performed well in the 2009 2K *BotPrize* competition [48], one of the best-known Turing-test-like competitions.

Since high-quality data from top experts is often inadequate, using imitation learning alone to train an agent may lead to low competitiveness of the agent. As mentioned in previous sections, reinforcement learning can easily raise the competitiveness or strength of an agent. Based on such fact, the literature Dossa et al. [33] proposed to combine RL and IL, making the two techniques complement each other. Concretely, the authors utilized RL and IL to train two teacher policies respectively, and then used the policy distillation method [113] to generate a skillful and human-like agent from the two teachers.

Compared to imitation learning, learning from human feedback (LfHF), or rather reinforcement learning from human feedback (RLHF), offers a more direct approach for achieving human-likeness. Typically, RLHF consists of two major steps, namely learning a reward model based on human feedback and then conducting RL training using that reward model. As the first work of RLHF, Christiano et al. [22] proposed to learn a reward model which can fit human preferences on a set of behavior sequences [22]. By using RL to learn a policy that maximizes the fitted rewards, an agent whose behavior is aligned with the human preferences can be obtained. Another successful application of RLHF, though not for games, is the astonishing chatbot ChatGPT of OpenAI. Note that ChatGPT got a large performance improvement by RLHF based on the former supervised-learning processes pre-training and small fine-tuning. This means that, in order to achieve a high level of human-likeness for game agents, RLHF can also be integrated with imitation learning. To promote RLHF research in the field of human-like agents, the leading AI conference *NeurIPS* held the *MineRL BASALT 2022 Competition* [88], where participants were encouraged to explore how RLHF methods can be used to address Almost Lifelike Tasks of the game Minecraft<sup>23</sup>. In the competition, two outstanding teams proposed methods that combined imitation learning and RLHF, and won the research prize.

Although remarkable progress has been made in developing human-like agents, there are still a number of challenges that need to be addressed. One of the primary challenges lies in evaluating the degree of human-likeness achieved by the agents. Currently, researchers often rely on human evaluators to provide feedback on the human-likeness. However, such a evaluation method can be subjective and inefficient. It is necessary to design appropriate metrics that can objectively evaluate the human-like performance of the agents. However, this is also a very difficult task due to the complexity of human-like behaviors and the variations across differ-

<sup>23</sup> <https://en.wikipedia.org/wiki/Minecraft>

ent game scenarios. As a result, achieving a comprehensive evaluation of human-like performance across multiple game scenarios remains a significant challenge. Another challenge that needs to be addressed is the efficiency of the methods used for learning human-like behaviors. Most current methods require a huge amount of human behavior data or feedback data. However, for newly launched games that have no available human data, manually producing the training data (e.g., by crowd-sourcing or the game developers) can be extremely time-consuming and laborious. This highlights the need for exploring new methods with minimal human intervention.

#### 4.2.2 Behavioral diversity

In the theory of biological evolution, biodiversity not only supports human and societal needs, but also maintains the long-term development of the organism itself. In the video game domain, the behavioral diversity of the intelligent agents in a game is also a core needs of players and important to the long-term development of the game. In games where the actions or strategies of players play a key role in gameplay, the agents having behavioral diversity are particularly meaningful. For example, when playing sports games like the *FIFA* series and the *NBA 2K* series, players always hope that their opponent agents can adaptively change their playing tactics as what really happens in the real-world sports games. Another example that can show the importance of behavioral diversity is the two super StarCraft agents AlphaStar [143] and StarCraft Commander (SCC) [148]. Due to the various policies learnt by these two agents, they had overcome the cycle of counter strategies in StarCraft II and defeated top professional players in live matches.

Specifically, the main function of the agents' behavioral diversity is to provide players with a more diverse gaming experience. Firstly, more diverse agents allows players to participate more actively in the game because they can interact with the agents in a wide variety of ways, without feeling the game monotonous or boring. The various behaviors of intelligent agents can arouse players' interest and curiosity, stimulate their exploration desire, and thus increase their enthusiasm. Secondly, the behavioral diversity of intelligent agents can make the game more challenging. This is because the agents can have different acting strategies, and players need to constantly try new counter-strategies to deal with different situations. Thirdly, behavioral diversity enables intelligent agents to make some unexpected but interesting movements, thus can add more fun to the game. Lastly, it is observed that human players may behave in different ways at different stages when playing games, thus behavioral diversity is also a crucial step towards human-like agents in a sense.

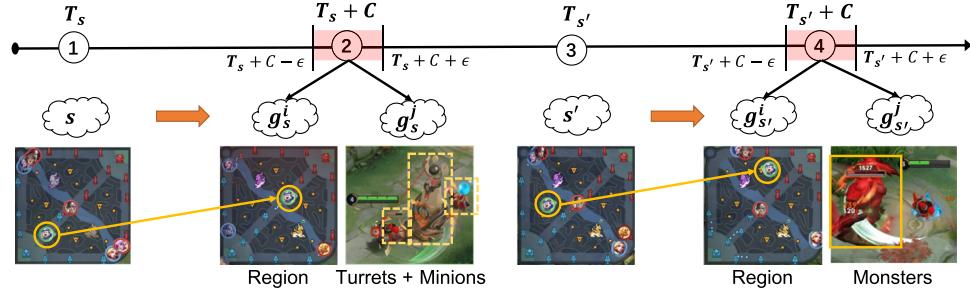
For generating intelligent agents with diverse behaviors, one common approach is the quality-diversity (QD) algorithm [27], which can be treated as an instantiation of the better-known evolutionary algorithm. Under the generic population-based optimization framework of EA, the QD algorithm works by iteratively selecting and re-producing elite agents from a population. The so-called elite agents mean that they can simultaneously maximize both the diversity and quality objectives. On the one hand, the quality objective requires the agents to evolve towards a certain level of intelligence. On the other hand, the reproduction mechanisms (e.g., crossover, mutation) together with the diversity objective ensure that diverse agents can be constantly generated and selected. As a kind of derivative-free approaches, the advantage of the QD algorithms is that they can be utilized to optimize any objective function. Also, QD can adopt any possible method as its optimizer, such as basic evolutionary strategies [71, 92, 99], clustering [149], and even reinforcement learning [23]. However, due to the high computational cost and sample inefficiency of the evolutionary techniques adopted, such QD algorithms are difficult to be applied in very complex video games.

Compared with the QD algorithm family, a much simpler approach is to utilize a gradient-based algorithm for the optimization of behavioral diversity. Usually, this often requires integrating the diversity goal to the original objective function (e.g., loss function or reward function) of the problem. The added objective of behavioral diversity can be an intrinsic reward function [82, 172], a diversity-driven regularization term [37, 53, 125], or a combination of the both [21]. But different from the QD algorithms, most of these approaches only treat the diversity goal as a necessary constraint to promote the optimization of the original objective. For example, the CDS algorithm [21] requires the behavioral diversity of the agents in a team for getting better coordinating performance, and the policy space response oracle (PSRO) algorithm family [3, 82, 101] utilizes behavioral diversity to ensure the rapid discovery of a Nash equilibrium strategy.

Currently, the approaches mentioned above only provide theoretical methodologies for achieving behavioral diversity, but have rarely been applied in the game industry. Perhaps the most relevant work of behavioral diversity to video games is the macro-goals guided (MGG) framework [40] proposed for the popular MOBA game *Honor of Kings*<sup>24</sup>. As shown in Fig. 14, MGG endows the agents with behavioral diversity by learning from how human players choose the location targets in a map. As mentioned above that behavioral diversity can help with improving human-likeness, it also holds true on the contrary.

Achieving the behavioral diversity of intelligent agents in video games is still a great challenge. The reasons are in

<sup>24</sup> [https://en.wikipedia.org/wiki/Honor\\_of\\_Kings](https://en.wikipedia.org/wiki/Honor_of_Kings)



**Fig. 14** An example of macro-goal label extraction in the MOBA game *Honor of Kings* [40]. The macro-goals at  $T_s$  is to move into the opposite middle lane and gain gold resources from turrets and minions, while at

$T_{s'}$  is to move to the upper jungle area and grab the monsters to level up.  $C$  is the delta of frame numbers between the state and its extracted macro-goals

two folds. First, currently there is not a simple and universal method for the game industry. Although the QD algorithms [23, 71, 92, 99, 149] provide a general framework, they can only be implemented in some small-scale environments. Other approaches can only be applied in some specific scenarios and are difficult to be transferred to other tasks. For instance, the MGG framework [40] needs to collect a large amount of human data and sort out human strategies to manually design the macro goals for MOBA games. For cases where human data is not available or other types of video games, MGG may not be suitable. Second, currently there is no unified standard for measuring behavioral diversity in both the academia and industry. Some works adopt policy entropy or population diversity, while others work through qualitative visual metrics. These metrics seem to be relatively one-sided, and a comprehensive set of evaluation criteria needs to be released urgently.

#### 4.2.3 Game testing

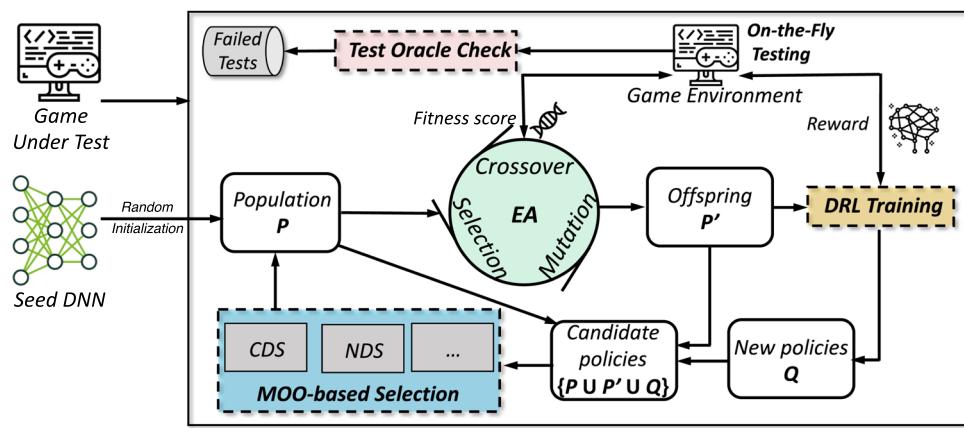
For game players, the intelligent agents in a video game can serve as interactive entities and provide fun to them. For game developers, such agents can be very useful tools for automatic game testing, even though they are not designed for this originally. The key challenge of game testing is that bugs or unexpected situations are only triggered when the game is being played, and may be hidden in very obscure places. Thus, finding those bugs requires a certain level of game playing skills and experiences. While traditional testing method like manually playing or using a script is time-consuming, labor-intensive, and inefficient, intelligent agents can automatically and tirelessly run through every part of a game, offering possibility to save human effort.

As a matter of fact, training intelligent agents for automatic game testing has no essential difference between training agents for game playing. Also, as game playing is a typical sequential decision-making process, most existing works of automatic game testing are based on reinforcement

learning. For example, Bergdahl et al. [5] experimented a deep reinforcement learning algorithm in an FPS game and found that RL can be good complement of scripted test [5]. They also showed that the advantage of intelligent agents exhibited more clearly in edge cases such as exploitation detecting and difficulty evaluation. Ariyurek et al. [2] utilized DRL and Monte-Carlo tree search to define two types of tester agents for defect finding, both of which were shown to be able to compete with experienced human testers in three games with 12 levels that contain 45 bugs.

It should be noted that game testing is not only about using an agent to complete the tasks designed by game developers, but also involves more complex playing tasks. For example, coverage testing for a MOBA game may require an agent to go through all possible positions in a map. And game balance testing for a sports game often needs to verify all game characters' ability to see whether some of them are overwhelming or underwhelming. Wu et al. [155] performed the first attempt of RL-based regression testing for MMORPGs in the game industry [155]. By training an intelligent RL agent which can capture the divergent dynamics of two game versions, potential regression bugs that result in the inconsistency are more likely to be discovered. Zheng et al. [170] did a more comprehensive work of using intelligent agent for game testing [170]. Firstly, they analyzed thousands of real bugs from commercial online games and classified the bugs into five categories. The bug categories include crash bugs, stuck bugs, logical bugs, game balance bugs and user experience bugs. Secondly, they developed an automated framework named *Wuji* to test the games and discover different types of bugs. As shown in Fig. 15, the framework combines DRL, evolutionary algorithms and multi-objective optimization (MOO) to perform an on-the-fly testing. At the same time, *Wuji* also constantly let the agents learn to play the game. The population in the big loop of EA are optimized by a DRL algorithm. Then, the MOO module selects a better set of agents as the next generation, in order to explore and test more areas of the game.

**Fig. 15** The workflow of the automated game testing framework *Wuji* [170], which integrates deep reinforcement learning, evolutionary algorithm, and multi-objective optimization



Although some exciting progress has been made, the application of intelligent agent in automatic game testing still faces some challenges. Firstly, commercial games are often highly complex and diverse. Particularly for large-scale multiplayer online games, the complexity is even higher, requiring stronger solutions. Training agents that can well play such games is not easy, let alone training them to conduct comprehensive and detailed game testing. Secondly, the tester agents created by techniques such as RL and EA need to respond to changes in the game environment and update their testing policies. However, changes in games are often frequent and even a small change can lead to unpredictable problems. This definitely poses some difficulties for agent training and updating. Lastly, but not the least, the training process of the tester agents may require a lot of time and computing resources, especially for complex video games. This can cause test engineers to spend a long time waiting for the training to complete, affecting testing efficiency.

## 5 Player data

In game playing, player data tracks the in-game player behavior such as activities, trajectories, relations, and so on. Based on the rich player data that records each action of players, most of the academic and industrial efforts are made into two typical tasks, i.e., player profiling such as cheat detection, and player experience improvement such as matchmaking systems. Cheat detection is a typical player profiling task of identifying whether the players violate the game rules, e.g., using outside-game tools or robot scripts, to gain illegal advantages. Matchmaking systems are designed for improving the player experience by arranging good and reasonable matches, especially for player-versus-player gameplays. For example, a good matchmaking system should guarantee that the competence of the two opposite sides is close enough to avoid boring one-sided competition. For cheat detection and matchmaking systems, traditional approaches usually adopt

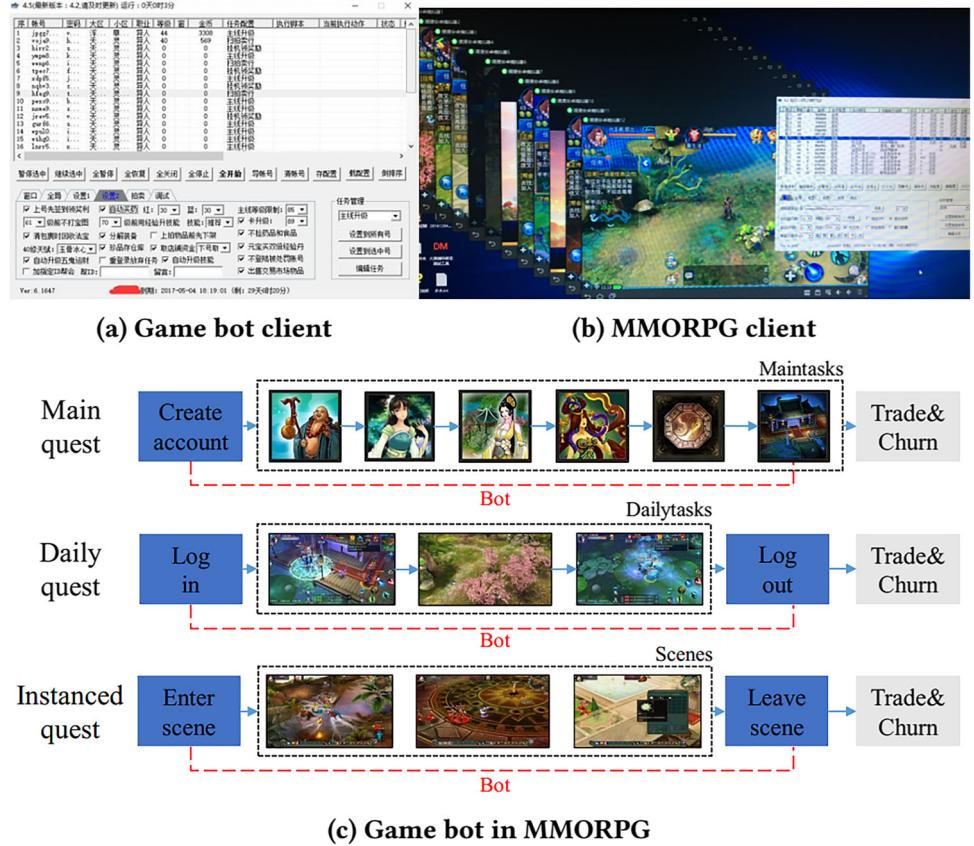
hand-crafted feature engineering followed by some specific machine learning models. However, these approaches face the growing data volume and computation complexity challenge. To address that challenge, deep learning-based methods are recently introduced for high-volume data analysis modeling. The following sections will specify two typical player data modeling tasks based on deep learning methods: 1) how to combine multiple sources of data to effectively detect player cheating; 2) how to optimize matchmaking systems to meet players' experience. Note that all the player data referred to in this paper are anonymized.

### 5.1 Cheat detection

While most of the game players enjoy the virtual world well together, someone yields to the temptation of outside-game tools for an easy win, and most notably, by illegally cheating. These cheating tools can mainly be divided into two categories, i.e., *plug-in programs* hacking the game clients to modify some critical configurations of gameplays, and *game bots* automatically interacting with games to collect resources [66, 131, 132, 152]. For example, plug-in-program cheaters can perform actions like limitless speeding and auto-aiming in FPS games; game-bot cheaters, as shown in Fig. 16, can build an army of dozens even hundreds of bots in MMORPG for simultaneously and constantly collecting game resources, which performs much more efficiently than normal players. Due to the rampant popularity of cheating tools, many unjustified rewards can be easily acquired within a short period, resulting in unfairness, reputation damages, and even the collapse of the economy in game [67, 69, 153]. Faced with aggressive game cheaters, who are often numerous and not easy to be identified, modern online games have to employ effective and efficient cheat detection solutions.

Game cheat detection can be formulated as one cognition task: an agent, either a human or AI-based model, infers the probability that one suspected player cheats given the

**Fig. 16** The illustration of game bot in MMORPG [131]. (a) The game cheaters configure the mode or patterns of game bots via client program. (b) Multiple instances can be handled by a game bot client. (c) Different game bots are usually designed for different tasks or game plays



observation about the player. Nowadays, large-scale online detection tasks have challenged the efficiency and effectiveness of manual detection due to its heavy dependence on her/his expertise in gameplay and cheating knowledge. Fortunately, AI-based models can screen each player and output the probability or score of suspicion much more quickly, which can even help to determine the subsequent operation measures like blocking the account, adjusting some game configurations for further observation, or exonerating the player as normal.

These AI-based models can be divided into two categories, i.e., supervised and unsupervised: the former usually prefers a regression or classification model based on known cheat cases and the latter adopts anomaly detection approaches to discover the potential cheats. In practice, the selection of AI-based methods often depends on the available data sources. This paper focuses on the three most typical data sources as shown in Table 1, i.e., player behaviors of various events, player trajectories tracing movement, and player relation of various connections.

Player behavior data is usually collected from the behavior log of each player, which depicts the specific events of each player from login to logout, including when, who, what, and how. Though behavior data is the most essential source for checking player cheats, the effectiveness is often restricted by

the completeness and correctness of the log system. Log systems have to deal with the trade-off between recording fewer events more efficiently and more events more effectively.

Player trajectory data records the sampled coordinates of each player to trace players' movement. Trajectory data is well-formed and easy to collect, but also requires models with high scalability considering millions even billions of data points per day.

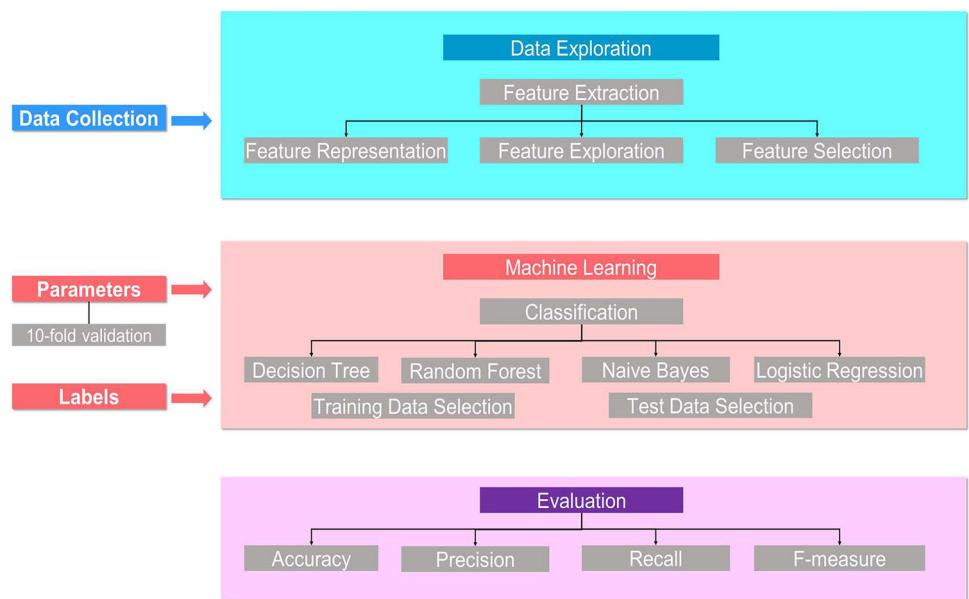
Player relation data is usually written to record the creation, change, or deletion of various connections between players. Relation data, though much harder to model and analyze, provides obviously one detection strategy to trace the upstream and downstream of cheat gangs inspired by “To Whom Is It A Benefit”.

**Behavior-based detection** As mentioned in Table 1, behavior data are usually players' logs of specific in-game events. Game bots, designed to collect resources quickly, behave in a couple of preset patterns that are quite different from the complicated and unpredictable patterns of human players. Some researchers attribute this discrepancy to the fact that game bots are profit-oriented, while humans are mostly fun-oriented, so game bots act in a tedious but efficient manner[61, 136]. Thus the main idea of behavior detection is to mine pattern differences between bot behaviors and human players' behaviors.

**Table 1** Three data sources for cheat detection

	Core Element	Format	Pros	Cons	Example
Behavior	Specific Events	TIME_STAMP, Player_ID, Event_ID, Other_Info	Easy to model	Event granularity trade-off	Player A logs in at 19:11 P.M., Player A enters the main scene at 19:15 P.M., Player A starts the daily quest task at 19:20 P.M.
Trajectory	Sampled Coordinates	TIME_STAMP, Player_ID, Coordinate_X, Coordinate_Y, Other_Coordinates, Other_Info	Easy to record	High-efficiency requirement	the character controlled by Player A reaches Position (355,- 125,26) of the main game scene at 19:11 P.M., the cursor con- trolled by Player A reaches Posi- tion (512, 100) of the computer monitor at 19:16 P.M., one fin- ger of Player A pushes Position (651,987) of the phone screen at 19:19 P.M.
Relation	Connections Between Players	TIME_STAMP, Player_1_ID, Player_2_ID, Rela- tion_Info	Good evidence	Hard to model and analyze	Player A accepts the invitation of friendship from Play B at 19:11 P.M., Player A and Player B join one team for the same game quest task at 19:14 P.M., Player A sells one rare weapon to Player B at the price of 5,000 game coins at 19:30 P.M.

**Fig. 17** Game bot detection based on behavioral features [61] including data collection, modeling, and evaluation procedures



One typical way to handle this challenge is to extract several discriminative hand-crafted behavior features and then build multiple machine learning models as a classifier to infer whether one player cheats or not [61]. As shown in Fig. 17, some indicative features can be carefully extracted like online duration, between-player chat count, the specifications of host machine hardware, and so on; then cheat detection can easily be built by using some highly efficient machine learning methods such as naive Bayes, logistic regression, random forest, and decision tree. While the hand-crafted features work very well, this case-by-case solution becomes less and less competitive and cost-effective for the various genres of games and gameplays.

To address the issues mentioned above, deep learning based approaches provide an alternative detection solution, which can effectively learn hidden representations for further classification or clustering from raw behavior data instead of specific feature engineering. McDonald et al. [85] adopt deep learning based VACnet in *Valve Anti-Cheat system*<sup>25</sup> for detecting cheating players in the well-known game *CS: GO* (as shown in Fig. 18). As shown in Fig. 19, Tao, Xu et al. [131, 158] propose a combined framework of supervised and unsupervised methods based on behavioral sequences. The supervised part focuses on efficiently identifying known bots, while the unsupervised part deals with the newborn bots or variants. In order to identify game bots more accurately, attention-based method is also employed to capture user patterns as shown in Fig. 19, e.g., some suspected behavior sequences contain repetitive actions of collecting game items for a long time.

**Trajectory-based detection** Game trajectory data, as shown in Table 1, which is much more common and can be easily collected in online games, consists of three types:

- Move trajectory records the path of the player's avatar within the game map [18, 64, 96].
  - Mouse trajectory records the path of the mouse in the game client in PC online games [105, 169].
  - Touch trajectory records the path of the player's finger on the device screen in mobile online games [75, 128].

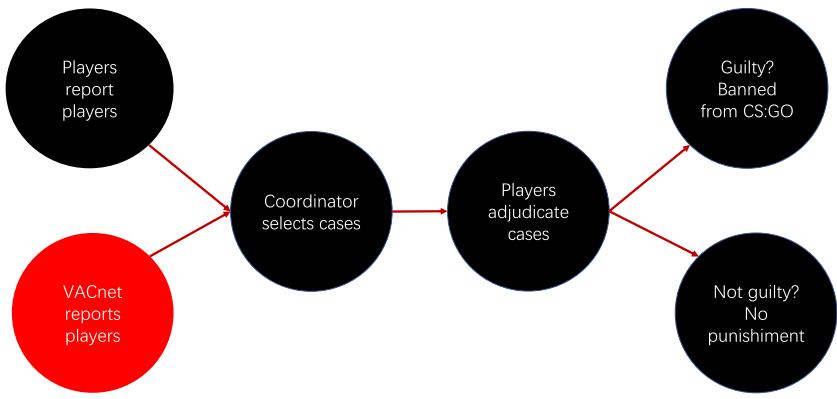
Similar to the behavior-based detection above, the core problem is how to effectively represent or mine features that can distinguish bots from humans. Two major approaches exist for trajectory-based detection: the former depends on feature engineering with expert experiences while the latter adopts deep learning models to learn good representation from raw trajectory data.

Feature engineering methods [18, 64, 96] usually design effective features by analyzing the differences in trajectories between the game bots and the normal players, and work on using feature-based methods to detect game bots. For example, these features cover the moving speed, direction, turning points, and angles between movements and so on; feature-based methods often utilize classical machine learning methods like SVM, naive Bayes, logistic regression, and so forth.

Considering the huge scale of trajectory data, high-efficiency methods like deep learning models, instead of case-by-case feature engineering, are more and more prevailing. Since each trajectory is comprised of movements, i.e., from one position to another, the trajectory data can be viewed as directed graphs in which each movement stands

<sup>25</sup> <https://help.steampowered.com/en/faqs/view/571A-97DA-70E9-FF74>

**Fig. 18** Valve presents the deep learning based VACnet [85], which boosts the performance of Valve Anti-Cheating system



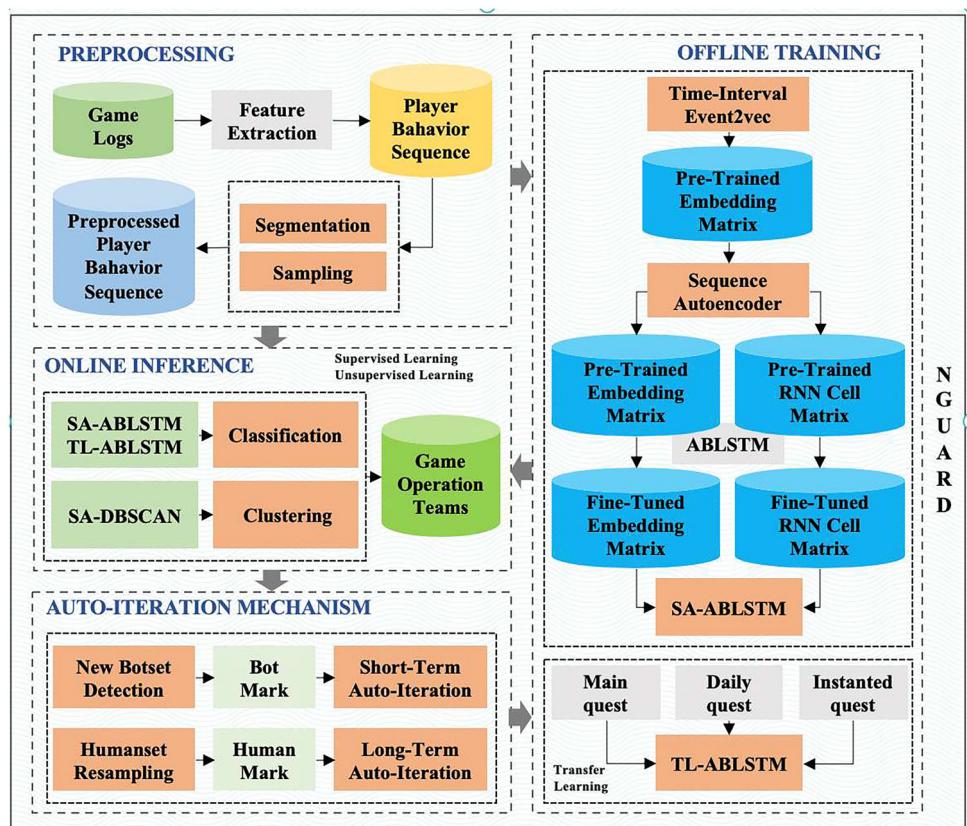
for an edge, and both the source and destination are nodes. Based on the graph construction, Qi et al. [105] apply a graph neural network-enhanced model to capture the players' complex behavioral patterns for game bot detection, e.g., the trajectories of game bots are more prone to densely distribute on some positions such that there exist a very large number of edges (movements) between these nodes (positions).

Another prevailing method of modeling trajectory data is based on spatial time series because each trajectory is actually made up of sequential coordinates with ordered timestamps. Following the effective pretrain-and-finetune fashion, Zhao et al. [169] propose a trajectory-based pre-trained model

named T-Detector for game bot detection. The whole framework firstly picks two typical trajectory data, i.e., in-game character trajectories and mouse trajectories, and builds general pre-trained encoders for these two spatial time series. The T-detector encoder can be easily pretrained with several kinds of models (e.g., multi-layer perceptron and self-attention-based models) to represent these trajectories for effectively identifying game bots.

Beyond the move and mouse trajectory data, touch data is more challenging because it contains information about finger operations and associations between different fingers. The following presents the approaches to address the two significant challenges in touch data scenarios, i.e., zero-shot

**Fig. 19** NGuard [131]: Behavioral sequence-based cheat detection model, which includes supervised, unsupervised, and model iteration



detection for unknown bots and few-shot detection for scarce labels.

To address unknown bots, one viable solution is to adopt contrastive strategies for building effective representations by using some task-specific data augment approaches of generating positive or negative samples. In the case of trajectory-based detection, some contrastive strategies can be employed by refining both finger-moving patterns and players' operation habits. Take an example of finger-moving patterns, when encountering a boss, most human players tend to move around slowly and carefully, trying to find its weakness. However, the game bots usually directly rush to the boss according to an established track and release a skill. Thus the first contrastive strategy can be implemented by constructing positive/negative samples according to finger-moving sequences of the same/different players.

In practice, the above finger-moving contrastive strategy cannot cover all game bots since some AI-powered game bots can behave very similarly to human players. To this end, players' operation habits are used for designing the second contrastive strategy, e.g., some players prefer releasing some combos of skills in similar game requests. Cheating players' operation habits are more similar even the same while the human players' operation habits are various due to his/her own traits. To exploit this distinction in operation habits between humans and game bots, the second contrastive strategy generates positive samples from operation records of the same player and negative samples from those of different players. Combining the aforementioned two contrastive strategies, Li et al. [75] introduce a Transformer-style game bot detection model, namely FingFormer, which can be effectively applied for zero-shot game bot detection based on touch data.

In practice, a few new bots can be found by chance, e.g., some players are willing to report bots they met. Though valuable and helpful, the super sparsity of the new labels poses a challenge of how to adapt the existing detection model for effectively identifying the newfound bots. Recently, few-shot learning methods, e.g., meta-learning, have gained solid progresses in several label-sparse fields like image classification, target detection, and so on. To this end, Su et al. [128] focus on the label scarcity problem and introduce a mobile game cheating detection framework based on meta-learning, namely FCDGame, to detect the cheats under the few-shot learning framework. FCDGame adopts a Cross-Pattern Meta-Learner for optimizing detection framework on known bots data then implements fine-tuning for the newly found few bots.

**Relationship-based detection** Apart from the previous two data sources for directly detecting cheats, the relation data (as shown in Table. 1) can also be used to discover the cheat gangs hidden in normal players. Regarding each account (player or bot) as one node and each relation (e.g.,

friendship, teammate, co-play, trade) between accounts as one edge, the main idea of relation-based detection is to discover suspected groups (e.g., edges, nodes, or sub-graph).

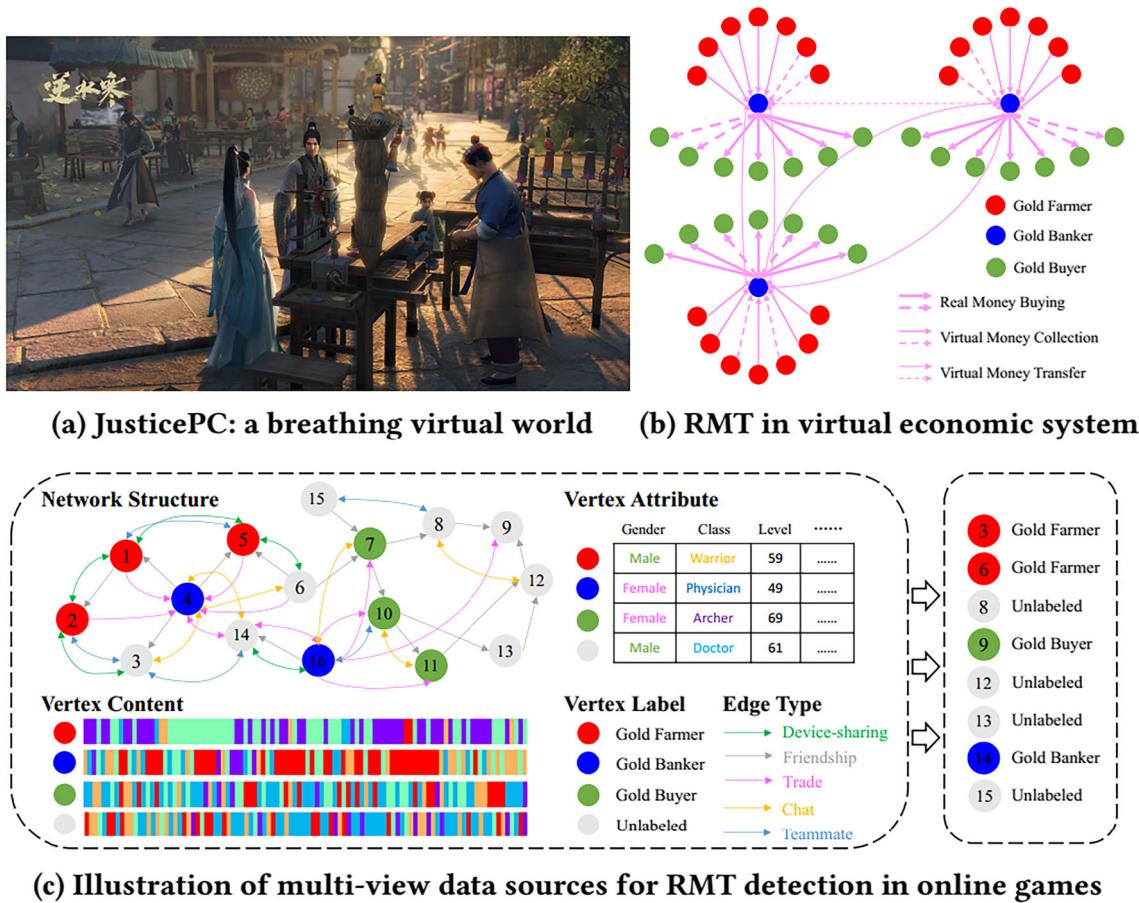
Along this line, Oh et al. [95] and Tao et al. [132] have explored social networks and found that game bots mostly exhibit a rather simple social relationship than normal players so that they can be identified by quantifying the social connection status. Typically, the high volume of transaction records in online games builds a huge dynamic network, where some abnormal nodes (i.e., players) and edges (i.e., interactions) would reveal cheating patterns, e.g., real money trading (RMT) [94, 132]. For example, these RMT cheaters usually get around the regular in-game trading system such as selling some valuable item at a highly below-the-market price and collecting the true money via outside-game payment.

However, these cunning RMT cheaters would try their best to cover up the abnormal trading records, resulting in that simple detection strategies like directly building a classifier do not work very well. One viable solution is to fully exploit the complementary effects from different aspects of player behavior by combining multi-source data, e.g., tabular user profiles, behavior sequences, and different kinds of relations like chat, teammate, friendship, and so on. In this way, as shown in Fig. 20, Tao et al. [132] adopt multi-view approaches based on the complementary multi-source data for effectively discovering these irregular transactions that disrupt the economic balance in online games. Besides, some researchers dive deep into detecting irregular trading abuse patterns such as rolling, which means alternatively repeating the selling and purchasing at different prices, and automatically discovering the relevant accounts. Along this line, Noh et al. [94] observe the abuse that RMTs enable through the auction house and propose a method to detect abuse using only a transaction log.

While AI-enhanced cheat detection methods have achieved sound progress in locating naughty players using known cheating techniques, there are still many unaddressed challenges in this endless battle.

On one hand, the substantial rewards, even illegal, have been driving cheat-making groups to constantly produce new-born loopholes or variants for walking around the current in-game regulation. Unsupervised, self-supervised, and even reinforcement learning based solutions are developed for attempting to maintain the power of the current solution. However, literally zero-shot scenarios and the lack of reliable evaluation have dramatically restricted the improvements of these methods. Some researchers have pointed out that data quality or human-in-the-loop labeling may relieve the cheat drift issue.

On the other hand, it is far from convincing to give only the predicted cheat scores, risks, or rankings for responsible game operation teams. Some high-stake actions like block-



**Fig. 20** MVAN [132]: Multi-view attention networks for real money trading detection in online games

ing cheating accounts require highly evidential facts instead of prediction results from modern black-box AI models as mentioned in Relation-based Detection. Recently, eXplainable AI (XAI) based methods have been adopted in the anti-cheat schema for generating high-importance features or human-understood rules [133, 134]. Having said that, there is obviously a long way to go for improving the effectiveness and efficiency of the current XAI methods.

## 5.2 Match making

In recent years, online games have attracted millions of players. For example, popular MOBA games such as *League of Legends* (LOL) or *Defense of the Ancients 2* (DOTA2) have become one of the most played genres, and they also become important platforms for esports tournaments, distributing millions of dollars in prizes. Among these games, one of the fundamental gameplay designs is PvP, a game mode in which multiple players directly engage in competition. For example, in DOTA2, two teams of five players compete against each other to destroy the enemy's base. For these PvP-centric online games, players usually spend the

majority of the game time in a series of matches, either one versus one or multiple versus multiple. Thus, as one of the core modules of these online games, matchmaking is the process of arranging multiple players into PvP matches. In competitive games and sports, reasonable and fair matchmaking mechanisms can bring players a more enjoyable and exciting game experience.

Let  $\mathcal{P} = \{p_i | i = 1, 2, \dots, N\}$  be the player pool that one could select from and  $N = |\mathcal{P}|$  be the number of all candidate players. Each player is associated with a profile vector  $x$ , which can describe various aspects of the player's ability such as precision, speed, and strategies, i.e.,  $\mathcal{F} = \{x_i \in \mathbb{R}^d | p_i \in \mathcal{P}\}$ . A team  $T = \{p_i \in \mathcal{P} | i = 1, 2, \dots, K\}$  is any (unordered) player combination selected from the player pool  $\mathcal{P}$ , and  $K$  denotes the team size (number of players in a team). Supposing that  $N$  players can form  $M$  matches (i.e.,  $N = 2K M$ ), our goal is to generate an optimal matching sequence  $\mathcal{M} = \{T_a^m, T_b^m\}_{m=1}^M$ , with the greatest gross player utilities measured by match level:

$$\mathcal{U}(\mathcal{M}) = \sum_m \mathcal{U}_m(T_a, T_b) \quad (1)$$

where  $\mathcal{U}_m(T_a, T_b)$  denotes the utility function when team  $T_a$  and  $T_b$  compete with each other in a match  $m \in \{1, 2, \dots, M\}$ . Here, the definition of utilities is general and could be implemented based on specific matchmaking goals, e.g., the balance of abilities between the two teams in a match.

Matchmaking arranges multiple players into PvP competitions. With many players requesting the gaming service and indicating their availability, the matchmaking system forms teams for queuing players and brings up every two teams into one combat.

A typical matchmaking system is actually a large-scale online planning system, which must quickly assign proper players to different teams for optimizing some utilities and objectives. To efficiently implement the whole matchmaking system, offline model training is usually explored for learning the utilities of different matches, i.e., what is a good match. Taking the above offline-learned utilities as objectives, the online planning workflow can be deployed as a large-scale combinatorial optimization system, i.e., search for the best team-ups among the available online game players. The following text will specify offline model training and then introduce online planning workflow.

**Offline model training** Offline model learning aims to grasp the relations among the individuals (heroes/players) and mine the implicit law behind the good or bad matches. Given a dataset with  $M$  observed matches, a population of players  $\mathcal{P} = \{p_1, \dots, p_N\}$  are involved. In each game, two disjoint teams competed with each other, which are denoted as  $T_{win}, T_{lose}$  by the outcome. For simplicity, each team has  $K$  members, where  $K = \|T_{win}\| = \|T_{lose}\|$  denotes the size of both teams. In this way, offline model training learns relations among individual players from the occurred records of  $T_{win}$  and  $T_{lose}$ , and estimates team-up utilities for any team-ups. The match utilities (good or not good) mainly depend on the final match outcome, e.g., matches ending in a tie are assumed to be more fascinating because both sides are perfectly matched in the same ability level. Thus in the stage of offline model training, the player ability is first learned from historic matches then match outcome prediction models are built for finding possible team-ups. Certainly, the player ability modeling and match outcome prediction can be implemented simultaneously. For better illustration, the player ability is first introduced and followed by match outcome prediction.

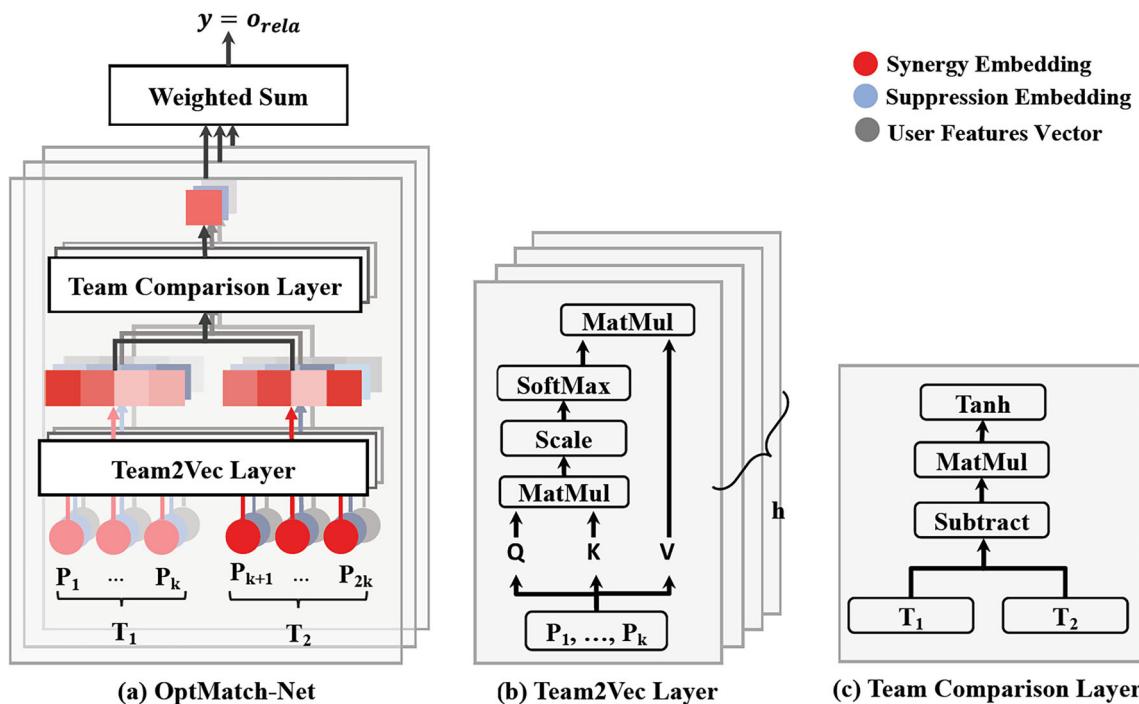
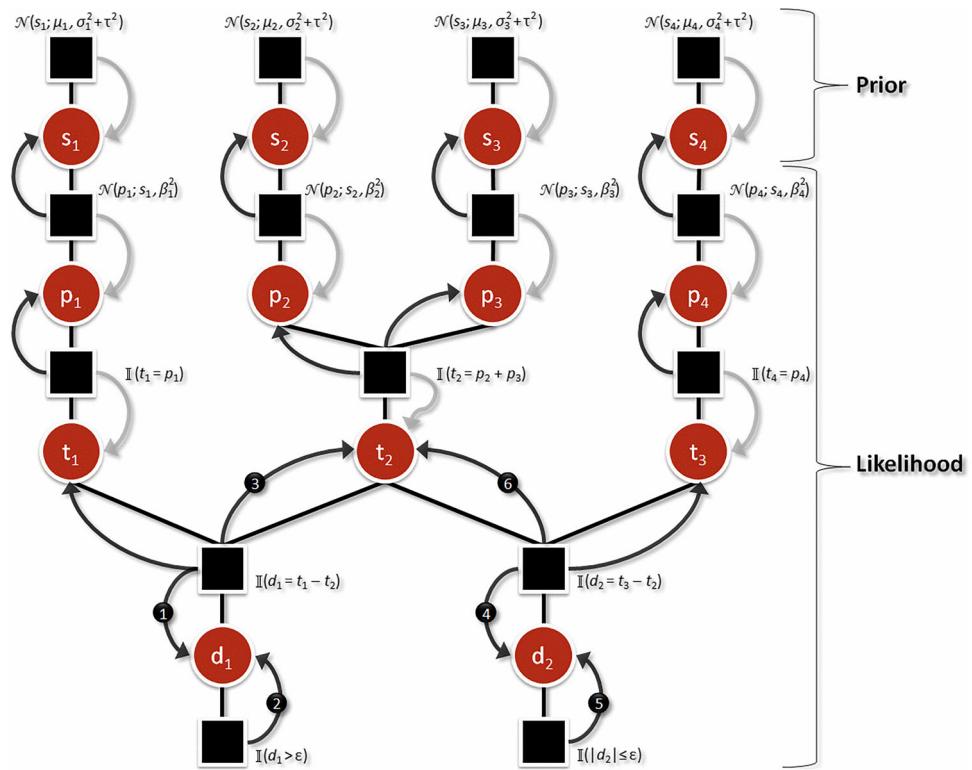
As the core part of offline model training, player ability modeling plays a very critical role in the whole matchmaking system. Many existing matchmaking algorithms originate from pairwise comparison [138]. The seminal work of the Bradley-Terry model [9], a probability model that can parameterize the ability of the two 1-versus-1 opponents. Bradley-Terry model can learn the defined player ability by fitting the true match outcome. However, in the multiplayer online games, game outcomes often refer to teams of play-

ers. All of the Bradley-Terry model, the Elo system, and the Glicko system mainly consider the 1-versus-1 scenario, e.g., Go, tennis, chess, and so on, instead of team-oriented multiple-versus-multiple games, e.g., basketball, and soccer. To address such challenges, several efforts have been made to extend the basic rating system [47, 56]. For example, the TrueSkill [47] from Microsoft and its extensions [90] generalizes the Elo system to games with flexible numbers of players and teams, in a principled Bayesian framework. TrueSkill is built upon a probabilistic graphic model shown in Fig. 21, which has been utilized with game matchmaking on Xbox Live platform.

Match outcome prediction is a straightforward, but effective, idea of modeling the utility of different team-ups, i.e., what is a good match. One main assumption is that the players prefer the more balanced matches, or say neck to neck, resulting in more competitive and excellent matches. Recently, many researchers adopt deep learning based methods for better representing player profiles like online time and historical win rate and in-game activities such as action sequence and gameplay requests. To effectively implement the in-match real-time win rate prediction, Wang et al. [146] combine Transformer to learn the players' state at each time step and predict the final win-or-loss result for both sides. Apart from rich record data, some gaming theories are also introduced for better prediction, Gong et al. [41] propose a deep learning based method named OptMatch-Net (Fig. 22) for effectively learning synergy and suppression between different game roles. Nonetheless, some researchers argue that player engagement, more than the game balance, can serve as a better team-up utility. To address this kind of team-up utility, Chen et al. [20] with colleagues from Electronic Arts reveal the correlation between win-or-loss and game player engagement for building a new matchmaking system named EOMM. Different from the previous outcome-orientated methods, EOMM manages to optimize the utility of multiple matches, e.g., how the player feels through three win-win-lose matches and whether she will leave or keep playing.

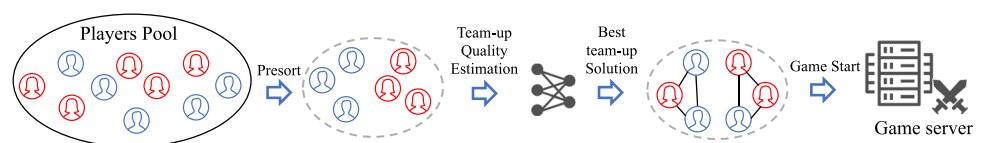
**Online planning workflow** Generally speaking, the online workflow can be regarded as one combinatorial optimization task where the objective function is defined by the aforementioned modeled utility offline. In practice, matchmaking is applied to a pool of players  $\mathcal{P}_t = \{p_1, \dots, p_n\}$  to find the optimal assignments for players by maximizing the total utility, as shown in Fig. 23. A heuristic method (i.e., pre-sort) is adopted to split players into several units, each with  $n_m$  players, by the players' ability ranking tiers. On a unit of  $2K$  players, several match-up candidates can be found based on some heuristics in [154], e.g., random pick, which are proved to be *ex-post Pareto efficient*. Then the gross utility of those  $2K$  players can be predicted for each match-up candidate with the team-up quality estimation network. Finally the

**Fig. 21** The factor graph of TrueSkill rating system [90], which probabilistically reparameterizes the Elo system by introducing suitable priors



**Fig. 22** The framework of OptMatch-Net [41], which can effectively learn the representation of team and accurately predict the outcomes for potential matches

**Fig. 23** The workflow of typical matchmaking systems [30], including selecting candidate players for various battle rooms, and balancing player arrangement for each match



candidate match-up with the maximum utility  $\mathcal{U}_m$  is chosen as the matchmaking assignment.

Recently, some researchers try their best to find one-stage global optimization solution, instead of the multi-stage heuristic, possibly sub-optimal, method as shown in Fig. 23. In fact, the progress in neural combinatorial optimization has shed some light on developing new effective solutions. Deng et al. [30] take one step further to fuse the two-stage matchmaking processes into one step by using a reinforcement learning based policy framework.

In the future, in addition to the win/loss balance of the match, more optimization objectives can be considered, such as minimizing the average waiting time of players. The more challenging direction is to consider long-term objectives, such as minimizing the players' churn.

## 6 Conclusion

In conclusion, this paper has provided a comprehensive review of the various deep learning applications in the game industry, focusing on game asset data, interaction data, and player data. It has highlighted the potential of deep learning in revolutionizing game development, experience, and operation by enabling more efficient and accessible content production, enhancing player experience, and improving game ecology. Specifically, this paper has discussed the use of deep learning in generating game assets, such as object images, 3D scene and avatar models, and facial animations. It has also explored the role of deep learning in processing interaction data, including conversation and behavior data, to create more vivid and interactive NPCs. Moreover, the paper has examined the application of deep learning in player data analysis for cheat detection, match making, and understanding player preferences.

Despite the significant progress made in applying deep learning to the game industry, challenges still remain. Further research is needed to optimize the efficiency and effectiveness of deep learning algorithms, as well as to explore novel applications that can further enhance the gaming experience. As the field of deep learning continues to evolve, it is expected that its applications in the game industry will become even more pervasive, ultimately revolutionizing the way games are designed, developed, and played. This paper serves as a valuable reference for future research and development efforts in the application of deep learning to the ever-growing and dynamic world of gaming.

It is important to recognize that deep learning is not a panacea, and there remain challenges in ensuring the quality, fairness, and ethical considerations of AI-generated content in games. Future research should focus on addressing these challenges, as well as exploring novel applications of deep learning in gaming, to fully harness the potential of AI in

creating more immersive, engaging, and personalized gaming experiences for all players.

**Acknowledgements** We would like to thank Jiajun Bu and Weixun Wang for their generous and helpful discussions, as well as their constructive suggestions.

**Funding** This work is supported by the Key Research and Development Program of Zhejiang Province (No. 2022C01011).

**Data Availability Statement** No datasets were generated or analysed during the current study

## Declarations

**Conflict of interest** All authors declare that they have no conflicts of interest.

**Ethics approval** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

1. Adiwardana D, Luong MT, So DR et al (2020) Towards a human-like open-domain chatbot. arXiv preprint [arXiv:2001.09977](https://arxiv.org/abs/2001.09977)
2. Ariyurek S, Betin-Can A, Surer E (2019) Automated video game testing using synthetic and humanlike agents. IEEE Trans Games 13(1):50–67
3. Balduzzi D, Garnelo M, Bachrach Y et al (2019) Open-ended learning in symmetric zero-sum games. In: International conference on machine learning, pp 434–443
4. Bao S, He H, Wang F, et al (2021) Plato-xl: Exploring the large-scale pre-training of dialogue generation. arXiv preprint [arXiv:2109.09519](https://arxiv.org/abs/2109.09519)
5. Bergdahl J, Gordillo C, Tollmar K et al (2020) Augmenting automated game testing with deep reinforcement learning. In: IEEE conference on games, pp 600–603
6. Berner C, Brockman G, Chan B, et al (2019) Dota 2 with large scale deep reinforcement learning. CoRR [arXiv:1912.06680](https://arxiv.org/abs/1912.06680)
7. Blanz V, Vetter T (1999) A morphable model for the synthesis of 3d faces. In: 26th Annual conference on computer graphics and interactive techniques, pp 187–194
8. Booth J, Roussos A, Ponniah A et al (2018) Large scale 3d morphable models. Int J Comput Vis 126(2–4):233–254
9. Bradley RA, Terry ME (1952) Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika 39(3/4):324–345
10. Brown K, Hamilton A (2016) Photogrammetry and star wars battlefield. In: GDC 2016: Game developer conference, San Francisco: UBM Tech
11. Brown N, Sandholm T (2018) Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. Science 359(6374):418–424
12. Brown N, Sandholm T (2019) Superhuman AI for multiplayer poker. Science 365(6456):885–890
13. Brown T, Mann B, Ryder N et al (2020) Language models are few-shot learners. Advances in neural information processing systems 33:1877–1901
14. Browne CB, Powley E, Whitehouse D et al (2012) A survey of monte carlo tree search methods. IEEE Trans Comput Intell AI Games 4(1):1–43

15. Cao C, Weng Y, Zhou S et al (2013) Facewarehouse: A 3d facial expression database for visual computing. *IEEE Trans Visual Comput Graph* 20(3):413–425
16. Chen H, Liu X, Yin D et al (2017) A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter* 19(2):25–35
17. Chen K, Wang Y, Zhang SH et al (2021) Mocap-solver: A neural solver for optical motion capture data. *ACM Trans Graph (TOG)* 40(4):1–11
18. Chen KT, Liao A, Pao HKK, et al (2008) Game bot detection based on avatar trajectory. In: International conference on entertainment computing, Springer, pp 94–105
19. Chen Z, Yi D (2017) The game imitation: Deep supervised convolutional networks for quick video game AI. *CoRR arXiv:1702.05663*
20. Chen Z, Xue S, Kolen J et al (2017) Eomm: An engagement optimized matchmaking framework. In: International conference on world wide web, pp 1143–1150
21. Chenghao L, Wang T, Wu C et al (2021) Celebrating diversity in shared multi-agent reinforcement learning. *Ad Neural Inf Process Sys* 34
22. Christiano PF, Leike J, Brown T, et al (2017) Deep reinforcement learning from human preferences. *Ad Neural Inf Process Sys* 30
23. Cideron G, Pierrot T, Perrin N, et al (2020) Qd-rl: Efficient mixing of quality and diversity in reinforcement learning. *CoRR arXiv:2006.08505*
24. Cohen MM, Massaro DW (1993) Modeling coarticulation in synthetic visual speech. In: Models and techniques in computer animation. Springer, pp 139–156
25. Creswell A, White T, Dumoulin V et al (2018) Generative adversarial networks: An overview. *IEEE Signal Proc Mag* 35(1):53–65
26. Cudeiro D, Bolkart T, Laidlaw C et al (2019) Capture, learning, and synthesis of 3d speaking styles. In: IEEE/CVF conference on computer vision and pattern recognition, pp 10,101–10,111
27. Cully A, Demiris Y (2017) Quality and diversity optimization: A unifying modular framework. *IEEE Trans Evol Comput* 22(2):245–259
28. Dabral R, Mughal MH, Golyanik V, et al (2023) Mofusion: A framework for denoising-diffusion-based motion synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9760–9770
29. De Kegel B, Haahr M (2019) Procedural puzzle generation: A survey. *IEEE Trans Games* 12(1):21–40
30. Deng Q, Li H, Wang K, et al (2021) Globally optimized matchmaking in online games. In: ACM SIGKDD conference on knowledge discovery; Data mining, pp 2753–2763
31. Deng Y, Yang J, Xu S, et al (2019) Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In: IEEE computer vision and pattern recognition workshops
32. Ding Y (2021) Full-body animation generation for expressive npcs. In: Game developers conference (GDC)
33. Dossa RFJ, Lian X, Nomoto H et al (2019) A human-like agent based on a hybrid of reinforcement and imitation learning. In: Proceedings of the 2019 international joint conference on neural networks (IJCNN’19), pp 1–8
34. Dyulicheva YY, Glazieva AO (2022) Game based learning with artificial intelligence and immersive technologies: an overview. In: Ceur workshop proceedings, pp 146–159
35. Edwards P, Landreth C, Fiume E et al (2016) Jali: an animator-centric viseme model for expressive lip synchronization. *ACM Trans Graph (TOG)* 35(4):1–11
36. Ekman P, Rosenberg E, Editors (1997) What the face reveals: Basic and applied studies of spontaneous expression using the facial action coding system (facs)
37. Eysenbach B, Gupta A, Ibarz J et al (2018) Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*
38. Fan Y, Lin Z, Saito J et al (2022) Faceformer: Speech-driven 3d facial animation with transformers. In: IEEE/CVF conference on computer vision and pattern recognition, pp 18,770–18,780
39. Galway L, Charles D, Black M (2008) Machine learning in digital games: a survey. *Artif Intell Rev* 29:123–161
40. Gao Y, Shi B, Du X et al (2021) Learning diverse policies in moba games via macro-goals. In: Ranzato M, Beygelzimer A, Dauphin Y, et al (eds) Advances in Neural Information Processing Systems, pp 16,171–16,182
41. Gong L, Feng X, Ye D et al (2020) Optmatch: Optimized matchmaking via modeling the high-order interactions on the arena. In: ACM SIGKDD International conference on knowledge discovery & data mining, pp 2300–2310
42. Goodfellow I, Pouget-Abadie J, Mirza M et al (2020) Generative adversarial networks. *Commun ACM* 63(11):139–144
43. Gorman B, Humphrys M (2007) Imitative learning of combat behaviours in first-person computer games. *Proceedings of CGAMES* pp 85–90
44. Gu X, Fan Z, Zhu S, et al (2020) Cascade cost volume for high-resolution multi-view stereo and stereo matching. In: IEEE/CVF Conference on computer vision and pattern recognition, pp 2495–2504
45. Hannun A, Case C, Casper J et al (2014) Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*
46. Harmer J, Gisslén L, del Val J et al (2018) Imitation learning with concurrent actions in 3d games. In: IEEE Conference on computational intelligence and games. IEEE, pp 1–8
47. Herbrich R, Minka T, Graepel T (2007) Trueskill™: a bayesian skill rating system. In: Advances in neural information processing systems, pp 569–576
48. Hingston P (2009) The 2k botprize. In: Lanzi PL (ed) IEEE Symposium on Computational Intelligence and Games
49. Ho J, Jain A, Abbeel P (2020) Denoising diffusion probabilistic models. *Ad Neural Inf Process Sys* 33:6840–6851
50. Ho J, Jain A, Abbeel P (2020) Denoising diffusion probabilistic models. *Ad Neural Inf Process Sys* 33:6840–6851
51. Holden D (2018) Robust solving of optical motion capture data by denoising. *ACM Trans Graph (TOG)* 37(4):1–12
52. Holden D (2019) A new era of performance capture with machine learning. In: Game developers conference (GDC)
53. Hong ZW, Shann TY, Su SY, et al (2018) Diversity-driven exploration strategy for deep reinforcement learning. *Ad Neural Inf Process Sys* 31
54. Hooshyar D, Yousefi M, Lim H (2018) Data-driven approaches to game player modeling: a systematic literature review. *ACM Comput Surv (CSUR)* 50(6):1–19
55. Hu EJ, Wallis P, Allen-Zhu Z et al (2021) Lora: Low-rank adaptation of large language models. In: International conference on learning representations
56. Huang TK, Lin CJ, Weng R (2004) A generalized bradley-terry model: From group competition to individual skill. *Ad Neural Inf Process Sys* 17:601–608
57. Hussein A, Gaber MM, Elyan E et al (2017) Imitation learning: A survey of learning methods. *ACM Comput Surv (CSUR)* 50(2):1–35
58. Jensen R, Dahl A, Vogiatzis G et al (2014) Large scale multi-view stereopsis evaluation. In: IEEE conference on computer vision and pattern recognition, pp 406–413
59. Ji Z, Lu Z, Li H (2014) An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*
60. Justesen N, Bontrager P, Togelius J et al (2019) Deep learning for video game playing. *IEEE Trans Games* 12(1):1–20

61. Kang AR, Jeong SH, Mohaisen A et al (2016) Multimodal game bot detection using user behavioral characteristics. SpringerPlus 5(1):1–19
62. Karras T, Aila T, Laine S et al (2017) Audio-driven facial animation by joint end-to-end learning of pose and emotion. ACM Trans Graph (TOG) 36(4):1–12
63. Kazhdan M, Hoppe H (2013) Screened poisson surface reconstruction. ACM Trans Graph (ToG) 32(3):1–13
64. van Kesteren M, Langevoort J, Grootjen F (2009) A step in the right direction: Botdetection in mmorpgs using movement analysis. In: Proc. of the 21st Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2009), pp 129–136
65. Knapitsch A, Park J, Zhou QY et al (2017) Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Trans Graph (ToG) 36(4):1–13
66. Kotkov D, Pandey G, Semenov A (2018) Gaming bot detection: a systematic literature review. In: International conference on computational social networks, Springer, pp 247–258
67. Kwon H, Mohaisen A, Woo J et al (2016) Crime scene reconstruction: Online gold farming network analysis. IEEE Trans Inf Forensics Security 12(3):544–556
68. Lattas A, Moschoglou S, Gecer B et al (2020) AvatarMe: Realistically renderable 3d facial reconstruction. In: IEEE/CVF Conference on computer vision and pattern recognition, pp 760–769
69. Lee E, Woo J, Kim H et al (2016) You are a game bot!: Uncovering game bots in mmorpgs via self-similarity in the wild. In: Ndss
70. Lee J, Marsella S (2006) Nonverbal behavior generator for embodied conversational agents. In: International conference on intelligent virtual agents, pp 243–255
71. Lehman J, Stanley KO (2011) Evolving a diversity of virtual creatures through novelty search and local competition. In: Annual conference on genetic and evolutionary computation, pp 211–218
72. Lewis JP, Anjyo K, Rhee T et al (2014) Practice and theory of blendshape facial models. Eurographics (State of the Art Reports) 1(8):2
73. Li J, Koyamada S, Ye Q et al (2020) Suphx: Mastering mahjong with deep reinforcement learning. CoRR arXiv:2003.13590
74. Li L, Wang S, Zhang Z et al (2021) Write-a-speaker: Text-based emotional and rhythmic talking-head generation. In: Proceedings of the AAAI conference on artificial intelligence, pp 1911–1920
75. Li W, Chu X, Su Y et al (2022) Fingformer: Contrastive graph-based finger operation transformer for unsupervised mobile game bot detection. In: ACM web conference, pp 3367–3375
76. Li X, Wu Z, Meng HM et al (2016) Expressive speech driven talking avatar synthesis with dblstm using limited amount of emotional bimodal data. In: Interspeech, pp 1477–1481
77. Lin J, Yuan Y, Shao T et al (2020) Towards high-fidelity 3d face reconstruction from in-the-wild images using graph convolutional networks. In: IEEE/CVF conference on computer vision and pattern recognition, pp 5891–5900
78. Lin J, Yuan Y, Zou Z (2021) Meingame: Create a game character face from a single portrait. In: Proceedings of the AAAI conference on artificial intelligence, pp 311–319
79. Lin J, Li L, Yuan Y et al (2022) Realistic game avatars auto-creation from single images via three-pathway network. In: 2022 IEEE Conference on Games (CoG), IEEE, pp 33–40
80. Lin J, Chang J, Liu L et al (2023) Being comes from not-being: Open-vocabulary text-to-motion generation with wordless training. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 23,222–23,231
81. Lin Z, Lin J, Li L et al (2022) High-quality 3d face reconstruction with affine convolutional networks. In: ACM international conference on multimedia
82. Liu X, Jia H, Wen Y et al (2021) Towards unifying behavioral and response diversity for open-ended learning in zero-sum games. Ad Neural Inf Process Sys 34:941–952
83. Loiudice A (2018) How milestone created the breathtaking biker's paradise that is strada della forra in upcoming ps4 racer ride 3. PlayStation Blog
84. Loiudice A (2019) A plague tale: Innocence overrun with megascans. Quixel <https://quixel.com/blog/2019/5/7/a-plague-tale-innocence-overrun-with-megascans>
85. McDonald J (2018) Robocalypse now: Using deep learning to combat cheating in 'counter-strike: Global offensive'. In: Game Developers Conference (GDC)
86. Mescheder L, Oechsle M, Niemeyer M et al (2019) Occupancy networks: Learning 3d reconstruction in function space. In: IEEE/CVF conference on computer vision and pattern recognition, pp 4460–4470
87. Miikkulainen R, Bryant BD, Cornelius R et al (2006) Computational intelligence in games. Principles and practice. Computational intelligence, pp 155–191
88. Milani S, Kanervisto A, Ramanauskas K et al (2023) Towards solving fuzzy tasks with human feedback: A retrospective of the minerl basalt 2022 competition. arXiv preprint arXiv:2303.13512
89. Mildenhall B, Srinivasan PP, Tancik M et al (2020) Nerf: Representing scenes as neural radiance fields for view synthesis. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I, pp 405–421
90. Minka T, Cleven R, Zaykov Y (2018) Trueskill 2: An improved bayesian skill rating system. Tech Rep
91. Mou C, Wang X, Xie L et al (2023) T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. arXiv preprint arXiv:2302.08453
92. Mouret JB, Clune J (2015) Illuminating search spaces by mapping elites. CoRR arXiv:1504.04909
93. Muñoz-Avila H, Bauckhage C, Bida M et al (2013) Learning and game ai. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
94. Noh Y, Jeong S, Kim HK (2021) Trading behind-the-scene: Analysis of online gold farming network in the auction house system. IEEE Trans Games 14(3):423–434
95. Oh J, Borbora ZH, Sharma D, et al (2013) Bot detection based on social interactions in mmorpgs. In: IEEE international conference on social computing, pp 536–543
96. Pao HK, Chen KT, Chang HC (2010) Game bot detection via avatar trajectory analysis. IEEE Trans Comput Intell AI Games 2(3):162–175
97. Park JJ, Florence P, Straub J et al (2019) Deepsdf: Learning continuous signed distance functions for shape representation. In: IEEE/CVF conference on computer vision and pattern recognition, pp 165–174
98. Park JS, O'Brien JC, Cai CJ et al (2023) Generative agents: Interactive simulacra of human behavior. arXiv preprint arXiv:2304.03442
99. Parker-Holder J, Pacchiano A, Choromanski K et al (2020) Effective diversity in population-based reinforcement learning. CoRR arXiv:2002.00632
100. Pelling C, Gardner H (2019) Two human-like imitation-learning bots with probabilistic behaviors. In: Proceedings of the 2019 IEEE conference on games (CoG'19), pp 1–7
101. Perez-Nieves N, Yang Y, Slumbers O et al (2021) Modelling behavioural diversity for learning in open-ended games. In: International conference on machine learning, PMLR, pp 8514–8524
102. Perolat J, De Vylder B, Hennes D et al (2022) Mastering the game of stratego with model-free multiagent reinforcement learning. Science 378(6623):990–996
103. Pham HX, Cheung S, Pavlovic V (2017) Speech-driven 3d facial animation with implicit emotional awareness: a deep learning

- approach. In: IEEE conference on computer vision and pattern recognition workshops, pp 80–88
104. Poznanski (2014) Visual revolution of the vanishing of ethan carter. *The Astronauts* <https://www.theastronauts.com/2014/03/visual-revolution-vanishing-ethan-carter>
  105. Qi X, Pu J, Zhao S et al (2022) A gnn-enhanced game bot detection model for mmorpgs. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 316–327
  106. Raab S, Leibovitch I, Li P et al (2023) Modi: Unconditional motion synthesis from diverse data. In: IEEE/CVF conference on computer vision and pattern recognition, pp 13,873–13,883
  107. Radford A, Kim JW, Hallacy C et al (2021) Learning transferable visual models from natural language supervision. In: International conference on machine learning, PMLR, pp 8748–8763
  108. Ramesh A, Dhariwal P, Nichol A et al (2022) Hierarchical text-conditional image generation with clip latents. arXiv preprint [arXiv:2204.06125](https://arxiv.org/abs/2204.06125)
  109. Richard A, Zollhöfer M, Wen Y et al (2021) Meshtalk: 3d face animation from speech using cross-modality disentanglement. In: IEEE/CVF International conference on computer vision, pp 1173–1182
  110. Risi S, Preuss M (2020) From chess and atari to starcraft and beyond: How game ai is driving the world of ai. KI-Künstliche Intelligenz 34:7–17
  111. Rombach R, Blattmann A, Lorenz D et al (2022) High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10,684–10,695
  112. Ruiz N, Li Y, Jampani V et al (2022) Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. arXiv preprint [arXiv:2208.12242](https://arxiv.org/abs/2208.12242)
  113. Rusu AA, Colmenarejo SG, Gulcehre C et al (2015) Policy distillation. CoRR abs/1511.06295 [arXiv:1511.06295](https://arxiv.org/abs/1511.06295)
  114. Sadoughi N, Busso C (2018) Expressive speech-driven lip movements with multitask learning. In: IEEE International conference on automatic face & gesture recognition, pp 409–415
  115. Sadoughi N, Busso C (2019) Speech-driven expressive talking lips with conditional sequential generative adversarial networks. IEEE Transactions on Affective Computing 12(4):1031–1044
  116. Schönberger JL, Zheng E, Frahm JM et al (2016) Pixelwise view selection for unstructured multi-view stereo. In: European conference on computer vision, Springer, pp 501–518
  117. Sherstinsky A (2020) Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Physica D: Nonlinear Phenomena 404(132):306
  118. Shi T, Yuan Y, Fan C et al (2019) Face-to-parameter translation for game character auto-creation. In: IEEE/CVF International conference on computer vision, pp 161–170
  119. Shi T, Zou Z, Shi Z et al (2020) Neural rendering for game character auto-creation. IEEE Trans Patt Anal Mach Intell 44(3):1489–1502
  120. Shi T, Zou Z, Song X et al (2020) Neutral face game character auto-creation via pokerface-gan. In: ACM International conference on multimedia, pp 3201–3209
  121. Shi T, Zuo Z, Yuan Y et al (2020) Fast and robust face-to-parameter translation for game character auto-creation. In: AAAI Conference on artificial intelligence, pp 1733–1740
  122. Shuster K, Xu J, Komeili M et al (2022) Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. arXiv preprint [arXiv:2208.03188](https://arxiv.org/abs/2208.03188)
  123. Silver D, Huang A, Maddison CJ et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489
  124. Silver D, Hubert T, Schrittwieser J et al (2017) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint [arXiv:1712.01815](https://arxiv.org/abs/1712.01815)
  125. Song Y, Wang J, Lukasiewicz T et al (2018) Diversity-driven extensible hierarchical reinforcement learning. CoRR [arXiv:1811.04324](https://arxiv.org/abs/1811.04324)
  126. Starke S, Zhao Y, Komura T et al (2020) Local motion phases for learning multi-contact character movements. ACM Trans Graph (TOG) 39(4):54–1
  127. Statham N, Jacob J, Fridenfalk M (2020) Photogrammetry for game environments 2014–2019: What happened since the vanishing of ethan carter. In: DIGRA 2020, Tampere, Finland, June 2–6, 2020
  128. Su Y, Yao D, Chu X et al (2022) Few-shot learning for trajectory-based mobile game cheating detection. In: ACM SIGKDD Conference on knowledge discovery and data mining, pp 3941–3949
  129. Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT press
  130. Takahashi T, Tanaka K, Oka N (2018) Adaptive mixed-initiative dialog motivates a game player to talk with an npc. In: International conference on human-agent interaction, pp 153–160
  131. Tao J, Xu J, Gong L et al (2018) Nguard: A game bot detection framework for netease mmorpgs. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 811–820
  132. Tao J, Lin J, Zhang S et al (2019) Mvan: Multi-view attention networks for real money trading detection in online games. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2536–2546
  133. Tao J, Xiong Y, Zhao S et al (2020) Xai-driven explainable multi-view game cheating detection. In: 2020 IEEE Conference on Games (CoG), IEEE, pp 144–151
  134. Tao J, Xiong Y, Zhao S et al (2023) Explainable ai for cheating detection and churn prediction in online games. IEEE Trans Games 15(2):242–251
  135. Taylor S, Kim T, Yue Y et al (2017) A deep learning approach for generalized speech animation. ACM Trans Graph (TOG) 36(4):1–11
  136. Thawonmas R, Kashifji Y, Chen KT (2008) Detection of mmorpg bots based on behavior analysis. In: Proceedings of the 2008 international conference on advances in computer entertainment technology, pp 91–94
  137. Thoppilan R, De Freitas D, Hall J et al (2022) Lamda: Language models for dialog applications. arXiv preprint [arXiv:2201.08239](https://arxiv.org/abs/2201.08239)
  138. Thurstone LL (1927) A law of comparative judgment. Psychol Rev 34(4):273
  139. Touvron H, Lavril T, Izacard G et al (2023) Llama: Open and efficient foundation language models. arXiv preprint [arXiv:2302.13971](https://arxiv.org/abs/2302.13971)
  140. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. Ad Neural Inf Process Sys 30
  141. Vicini D, Speierer S, Jakob W (2022) Differentiable signed distance function rendering. ACM Trans Graph (TOG) 41(4):1–18
  142. Vinyals O, Le Q (2015) A neural conversational model. arXiv preprint [arXiv:1506.05869](https://arxiv.org/abs/1506.05869)
  143. Vinyals O, Babuschkin I, Czarnecki WM et al (2019) Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature 575(7782):350–354
  144. Von Neumann J, Morgenstern O (2007) Theory of games and economic behavior. In: Theory of games and economic behavior. Princeton university press
  145. Wang G, Xie Y, Jiang Y et al (2023) Voyager: An open-ended embodied agent with large language models. arXiv preprint [arXiv:2305.16291](https://arxiv.org/abs/2305.16291)
  146. Wang K, Li H, Gong L et al (2020) Match tracing: A unified framework for real-time win prediction and quantifiable performance evaluation. In: Proceedings of the 29th ACM International conference on information & knowledge management, pp 2781–2788

147. Wang L, Hu K, Bai L et al (2023) Multi-scale control signal-aware transformer for motion synthesis without phase. arXiv preprint [arXiv:2303.01685](https://arxiv.org/abs/2303.01685)
148. Wang X, Song J, Qi P et al (2021) SCC: an efficient deep reinforcement learning agent mastering the game of starcraft II. In: the 38th International conference on machine learning, pp 10,905–10,915
149. Wang Y, Xue K, Qian C (2022) Evolutionary diversity optimization with clustering-based selection for reinforcement learning. In: Proceedings of the 10th international conference on learning representations (ICLR 2022). OpenReview.net
150. Ward I (2019) Call of duty: Modern warfare. PC, PS4, Xbox One
151. Whitley D (2001) An overview of evolutionary algorithms: practical issues and common pitfalls. Inf Softw Technol 43(14):817–831
152. Woo J, Kim HK (2012) Survey and research direction on online game security. In: Proceedings of the workshop at SIGGRAPH Asia, pp 19–25
153. Woo K, Kwon H, Kim Hc et al (2011) What can free money tell us on the virtual black market? In: Proceedings of the ACM SIGCOMM 2011 conference, pp 392–393
154. Wright M, Vorobeychik Y (2015) Mechanism design for team formation. In: AAAI Conference on artificial intelligence, pp 1050–1056
155. Wu Y, Chen Y, Xie X et al (2020) Regression testing of massively multiplayer online role-playing games. In: Proceedings of the 2020 IEEE international conference on software maintenance and evolution (ICSE 2020), IEEE, pp 692–696
156. Wurman PR, Barrett S, Kawamoto K et al (2022) Outracing champion gran turismo drivers with deep reinforcement learning. Nature 602(7896):223–228
157. Xing J, Xia M, Zhang Y et al (2023) Codetalker: Speech-driven 3d facial animation with discrete motion prior. arXiv preprint [arXiv:2301.02379](https://arxiv.org/abs/2301.02379)
158. Xu J, Luo Y, Tao J et al (2020) Nguard+: an attention-based game bot detection framework via player behavior sequences. ACM Trans Knowl Discov Data (TKDD) 14(6):1–24
159. Yannakakis GN, Togelius J (2018) Artificial intelligence and games. Springer
160. Yao Y, Luo Z, Li S et al (2018) Mvsnet: Depth inference for unstructured multi-view stereo. In: European conference on computer vision (ECCV), pp 767–783
161. Yariv L, Kasten Y, Moran D et al (2020) Multiview neural surface reconstruction by disentangling geometry and appearance. Ad Neural Inf Process Sys 33:2492–2502
162. Yariv L, Gu J, Kasten Y et al (2021) Volume rendering of neural implicit surfaces. Ad Neural Inf Process Sys 34:4805–4815
163. Yin QY, Yang J, Huang KQ et al (2023) Ai in human-computer gaming: Techniques, challenges and opportunities. Machine intelligence research pp 1–19
164. Zanetti S, Rhalibi AE (2004) Machine learning techniques for FPS in Q3. In: ACM SIGCHI International conference on advances in computer entertainment technology, pp 239–244
165. Zhang K, Riegler G, Snavely N et al (2020) Nerf++: Analyzing and improving neural radiance fields. arXiv preprint [arXiv:2010.07492](https://arxiv.org/abs/2010.07492)
166. Zhang L, Agrawala M (2023) Adding conditional control to text-to-image diffusion models. arXiv preprint [arXiv:2302.05543](https://arxiv.org/abs/2302.05543)
167. Zhang Y, Sun S, Galley M et al (2019) Dialogpt: Large-scale generative pre-training for conversational response generation. arXiv preprint [arXiv:1911.00536](https://arxiv.org/abs/1911.00536)
168. Zhao R, Li W, Hu Z et al (2023) Zero-shot text-to-parameter translation for game character auto-creation. arXiv preprint [arXiv:2303.01311](https://arxiv.org/abs/2303.01311)
169. Zhao S, Fang J, Zhao S et al (2022) T-detector: A trajectory based pre-trained model for game bot detection in mmorpgs. In: IEEE International conference on data engineering, pp 992–1003
170. Zheng Y, Xie X, Su T et al (2019) Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning. In: International conference on automated software engineering, IEEE, pp 772–784
171. Zhou Y, Xu Z, Landreth C et al (2018) Visemenet: Audio-driven animator-centric speech animation. ACM Trans Graph (TOG) 37(4):1–10
172. Zhou Z, Fu W, Zhang B et al (2022) Continuously discovering novel strategies via reward-switching policy optimization. In: International conference on learning representations. OpenReview.net

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

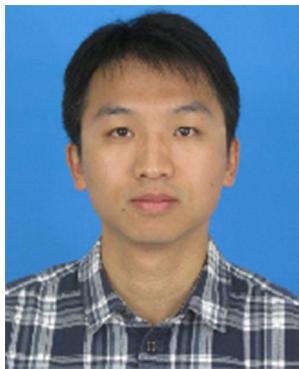
Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



**Zhipeng Hu** Vice president of NetEase Group. He graduated from Zhejiang University and once worked at MSRA. His major interest includes Game Design, Game AI, Computer Vision, and Graphics. He is also the co-founder of NetEase Fuxi AI Lab and has great insight into the cross-domain of game and AI. He has been responsible for developing several most successful and popular games in worldwide, such as Ghost and Justice.



**Yu Ding** is currently an artificial intelligence expert and the leader of virtual human group at NetEase Fuxi AI Lab, China. His research interests include virtual human, deep learning, image and video processing, talking-head generation, animation generation, multimodal computing, affective computing, nonverbal communication (face, gaze, and gesture), and embodied conversational agent. He received Ph.D. degree in Computer Science at Telecom Paris tech in Paris (France). He has published more than 50 papers in refereed journals and conference proceedings, such as IEEE TAC, CVPR, ICCV, AAAI, ACM MM, and IJCAI.



**Runze Wu** received the Ph.D. degree from University of Science and Technology of China. He is currently a Senior Researcher and the head of User Profiling Research Group in NetEase FUXI AI Lab, China. His research interests include user profiling, anomaly detection, causal inference, combinatorial optimization, deep learning, and various data mining and artificial intelligence applications across online games. He has published more than 50 papers in refereed journals and conference proceedings, such as IEEE TKDE, IEEE TMM, ACM TOIS, ACM TKDD, ACM SIGKDD, ACM MM, IJCAI, AAAI, TheWebConf, and ACM CIKM. He has served regularly on the program committees of conferences, including ACM SIGKDD, AAAI, IJCAI, TheWebConf, ACM CIKM and IEEE ICDM. Please find more information at <https://wu-runze.github.io/>



**Lincheng Li** received the B.S. and Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2011 and 2017 respectively. He is currently a researcher in Netease Fuxi AI Lab, Hangzhou, China. His research interests include computer vision, pattern recognition, and image and video processing.



**Rongsheng Zhang** is currently an artificial intelligence researcher at Netease Fuxi AI Lab, Hangzhou, China. He received the B.E. degree in communication engineering from Nanjing University, Jiangsu, China in 2015 and M.S. degree in communication engineering from Fudan University, Shanghai, China in 2018. His research interests include natural language process, multi-modal, pretrained model.



**Yujing Hu** received his Bachelor and Ph.D. degrees from the Department of Computer Science and Technology, Nanjing University in 2010 and 2015, respectively. Yujing joined Alibaba in 2016, where he worked on applying reinforcement learning to the domains of search and recommendation and proposed the first reinforcement learning framework for solving the ranking problem of a search engine. Yujing joined Netease Fuxi AI Lab in 2018, where he worked on the application and research of reinforcement learning in computer games. Yujing now is the head of the RL research group of NetEase Fuxi. Yujing has published more than 20 papers on top AI conferences and journals, such as NeurIPS, ICML, ICLR, IJCAI, AAAI, KDD, AAMAS, TCYB.



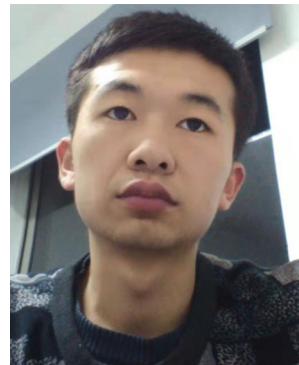
**Feng Qiu** is currently an Artificial Intelligence Researcher in Netease Fuxi AI Lab, Hangzhou, China. He received B.S. degree in Control Science and Engineering from Zhejiang University and B.E. degree in Automation from Harbin Institute of Technology. His research interests include deep learning, gesture recognition, pose estimation, affective computing and expression transfer.



**Zhimeng Zhang** received the B.E. degree and the M.S. degree from Shandong University, Shandong, China, in 2016 and 2019, respectively. He is currently a researcher working with Netease Fuxi AI Lab, Hangzhou, China. His current research interests include face reconstruction, animation generation and talking face generation.



**Kai Wang** Senior Researcher at FUXI AI Lab, NetEase, China. His current research topic is about recommender system using reinforcement learning, combinational optimization, deep learning, and various applications in online games. He has published many peer-reviewed research articles in top AI venues like ACM SIGKDD, AAAI, TheWebConf, and ACM CIKM.



**Yadong Xi** is currently an artificial intelligence researcher at Netease Fuxi AI Lab, Hangzhou, China. He received the B.E. degree and M.S. degree from Zhejiang University, Hangzhou, China in 2018. His research interests include natural language process, text generation, language model pretraining and dialogue system.



**Shiwei Zhao** received the M.S. degree from Zhejiang University, Hangzhou, China, in 2019. He is currently a researcher in NetEase Fuxi AI Lab, Hangzhou, China. His research interests include user profiling, anomaly detection, deep learning, and their application in games.



**Jiašhu Pu** is currently an artificial intelligence researcher at Netease Fuxi AI Lab, Hangzhou, China. He received the M.S. degree from the University of Sheffield in 2018. His research interests include Unsupervised Pre-Training and Representation Learning in Games.



**Yongqiang Zhang** received the M.S. and Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 2014 and 2018 respectively. He is currently a researcher in Netease Fuxi AI Lab, Hangzhou, China. His research interests include computer vision, 3D reconstruction, and neural rendering.



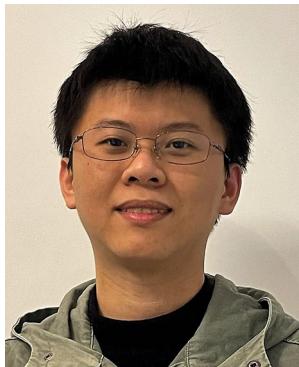
**Wei Zhang** received the B.E. degree in communication engineering from Nanjing University of Posts and Telecommunications, Jiangsu, China in 2017 and M.S. degree in electronic and information engineering from Zhejiang University, Zhejiang, China in 2020. She is currently a research scientist working with Netease Fuxi AI Lab, Hangzhou, China. Her current research interests include computer vision, expression embedding and facial affective analysis.



**Ji Jiang** received the M.S. degree from Peking University, Beijing, China, in 2022. He is currently a researcher in NetEase Fuxi AI Lab, Hangzhou, China. His research interests include reinforcement learning, game theory, robots and their application in games.



**Suzhen Wang** is currently an artificial intelligence researcher at Netease Fuxi AI Lab, Hangzhou, China. He received B.E. degree from Southeast University, Nanjing, China, in 2016, and the M.S. degree from Zhejiang University, Hangzhou, China in 2019. His research interests include computer vision, multimodal learning, image and video processing, animation generation and their application in games.



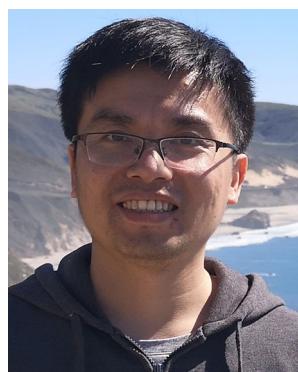
**Ke Chen** is currently an artificial intelligence researcher at Netease Fuxi AI Lab, Hangzhou, China. He received the B.E. degree and M.S. degree from Zhejiang University, Hangzhou, China in 2018. His research interests include reinforcement learning.



**Yan Song** is currently an artificial intelligence researcher at Netease Fuxi AI Lab, Hangzhou, China. He received the M.S. degree from Beijing University of Posts and Telecommunications in 2018. His research interests include reinforcement learning.



**Tianze Zhou** is an artificial intelligence researcher at Netease Fuxi AI Lab. He received the B.E. degree from Beijing Institute of Technology. The research interests of Tianze include machine learning and reinforcement learning for Game AI.



**Tangjie Lv** is the head of the AI team of NetEase Fuxi Lab. He received his PhD in computational mathematics from Peking University. After graduation, he worked for NetEase Games and Tencent AI Lab. At present, he is responsible for the AI research and technology development in games and other entertainment scenes. His research interests include reinforcement learning, machine learning, and game AI. He has led his team to achieve successful AI implementation in

several NetEase games.



**Jiarui Chen** is currently an artificial intelligence researcher at Netease Fuxi AI Lab, Hangzhou, China. He received the M.S. degree from the Department of Computer Science and Technology, Nanjing University in 2021. His research interests include reinforcement learning and multi-agent systems.



**Changjie Fan** is the Director of NetEase Fuxi AI Lab. He received his doctor's degree in Computer Science from University of Science and Technology of China. His research interest is in machine learning, including multiagent systems, deep reinforcement learning, game theory, and knowledge discovery.

## Authors and Affiliations

Zhipeng Hu<sup>1</sup> · Yu Ding<sup>2</sup>  · Runze Wu<sup>2</sup> · Lincheng Li<sup>2</sup> · Rongsheng Zhang<sup>2</sup> · Yujing Hu<sup>2</sup> · Feng Qiu<sup>2</sup> ·  
Zhimeng Zhang<sup>2</sup> · Kai Wang<sup>2</sup> · Shiwei Zhao<sup>2</sup> · Yongqiang Zhang<sup>2</sup> · Ji Jiang<sup>2</sup> · Yadong Xi<sup>2</sup> · Jiashu Pu<sup>2</sup> ·  
Wei Zhang<sup>2</sup> · Suzhen Wang<sup>2</sup> · Ke Chen<sup>2</sup> · Tianze Zhou<sup>2</sup> · Jiarui Chen<sup>2</sup> · Yan Song<sup>2</sup> · Tangjie Lv<sup>2</sup> · Changjie Fan<sup>2</sup>

Zhipeng Hu  
zphu@corp.netease.com

Runze Wu  
wurunze1@corp.netease.com

Lincheng Li  
lilincheng@corp.netease.com

Rongsheng Zhang  
zhangrongsheng@corp.netease.com

Yujing Hu  
huyujing@corp.netease.com

Feng Qiu  
qiufeng@corp.netease.com

Zhimeng Zhang  
zhangzhimeng@corp.netease.com

Kai Wang  
wangkai02@corp.netease.com

Shiwei Zhao  
zhaoshiwei@corp.netease.com

Yongqiang Zhang  
zhangyongqiang02@corp.netease.com

Ji Jiang  
jiangji@corp.netease.com

Yadong Xi  
xiyadong@corp.netease.com

Jiashu Pu  
pujiashu@corp.netease.com

Wei Zhang  
zhangwei05@corp.netease.com

Suzhen Wang  
wangsuzhen@corp.netease.com

Ke Chen  
chenke3@corp.netease.com

Tianze Zhou  
zhoutianze@corp.netease.com

Jiarui Chen  
chenjiarui@corp.netease.com

Yan Song  
songyan@corp.netease.com

Changjie Fan  
fanchangjie@corp.netease.com

<sup>1</sup> Zhejiang University, Hangzhou 310058, China

<sup>2</sup> NetEase Fuxi AI Lab, Hangzhou 310052, China