
بسم الله الرحمن الرحيم

دانشگاه اصفهان
دانشکده مهندسی کامپیوتر
گروه مهندسی فناوری اطلاعات

گزارش پروژه کارشناسی
رشته مهندسی کامپیوتر، گرایش فناوری اطلاعات

عنوان پروژه

# طراحی و پیاده‌سازی یک سیستم یادگیری تقویتی چندعاملی مشارکتی–رقابتی به‌منظور انجام بازی والیبال

استاد راهنما:
**دکتر مرجان کائدی**

پژوهشگر:
**رضا پورمحمدی**

شهریور ۱۴۰۲

**دانشگاه اصفهان**

**دانشکده مهندسی کامپیوتر**

**گروه مهندسی فناوری اطلاعات**

**پروژه کارشناسی رشتهٔ مهندسی کامپیوتر گرایش فناوری اطلاعات**

**آقای رضا پورمحمدی**

**تحت عنوان**

# "طراحی و پیاده‌سازی یک سیستم یادگیری تقویتی چندعاملی مشارکتی–رقابتی به‌منظور انجام بازی والیبال"

در تاریخ / / ۱۴۰۲ توسط هیأت داوران زیر بررسی و با نمره به تصویب نهایی رسید.

۱- استاد راهنمای پروژه:

دکتر امضا

۲- استاد داور:

دکتر امضا

امضای مدیرگروه

## باسپاس از

خدای متعال که هیچگاه فراموشم نکرد.

پدرم که به من ایثار و مادرم که به من عشق را آموخت. آنان که هیچگاه، ذره‌ای اندازه آنچه که شایسته آن اند عاشقشان نخواهم بود و این بزرگ‌ترین حسرت زندگی من است.

استاد عزیز و دلسوزم، دکتر کائدی که طی این مسیر بدون دانش و رهنمودهای ایشان میسر نبود.

## تقدیم به

مردم شریف، نجیب، پاک، صبور، جسور، ایستاده و آزاده کشور عزیزم ایران که در سخت‌ترین روزگار، زیباترین مردمان‌اند.

## چکیده

توسعه سیستم‌های چندعاملی هوشمند برای بازی‌ها، همواره یکی از حوزه‌های تحقیقاتی فعال در زمینه هوش مصنوعی بوده است. پیچیدگی وظایف و محیط‌ها در این گونه بازی‌ها، امکان حل آن‌ها از طریق عامل‌های از پیش برنامه‌ریزی شده را بسیار دشوار می‌سازد. به همین دلیل در طراحی چنین عامل‌هایی به استفاده از یادگیری تقویتی روی می‌آوریم.

نکته‌ای که باید به آن توجه کرد این است که طراحی چنین سیستم‌های چندعاملی مبتنی بر یادگیری تقویتی برای بازی‌ها، تنها دروازه‌ای است برای ورود این دانش به کاربردهای دنیای واقعی. سیستم‌های چندعاملی در دنیای کنونی ما اهمیت فزاینده‌ای پیدا کرده‌اند، چرا که بسیاری از سیستم‌های پیچیدهٔ دنیای واقعی نیاز به همکاری، رقابت و یا ترکیبی از این دو در بین چندین عامل دارند. به‌عنوان مثال می‌توان به خودروهای خودران، ربات‌ها، سیستم‌های حمل‌ونقل و فرایندهای تولید اشاره کرد که در آن‌ها چندین عامل باید برای دستیابی به یک هدف مشترک با یکدیگر همکاری و یا به‌منظور تصاحب یک منبع مشترک با هم رقابت کنند. طراحی سیستم‌های چندعاملی که بتوانند به طور مؤثر در محیط‌های پیچیده عمل کنند، یک چالش مهم باقی‌مانده است و رویکرد امیدوارکننده در مواجه با این چالش، استفاده از یادگیری تقویتی است که به عوامل اجازه می‌دهد تا از تعاملات خود با محیط بیاموزند و رفتار خود را بهینه کنند. والیبال یک بازی ایده‌آل برای طراحی یک سیستم چندعاملی و بررسی الگوریتم‌های یادگیری تقویتی است. این بازی شامل عواملی است که با یکدیگر به شیوهٔ مشارکتی-رقابتی، مشابه بسیاری از سیستم‌های دنیای واقعی، در تعامل هستند. این پروژه با توسعه یک سیستم یادگیری تقویتی چندعاملی مشارکتی-رقابتی برای بازی والیبال، بینش‌هایی را در مورد اینکه چگونه عوامل می‌توانند هماهنگی و برقراری ارتباط مؤثر با یکدیگر را در محیط‌های پویا و تصادفی بیاموزند، ارائه می‌دهد.

بنابراین، در این پروژه با طراحی یک بازی والیبال با استفاده از Unity و ML-Agents، یک سیستم چندعاملی مشارکتی-رقابتی ایجاد می‌شود که نه تنها شبیه‌سازی از بازی والیبال است، بلکه پلتفرمی مجازی خواهد بود برای مطالعه و پژوهش در مورد تأثیرات پارامترهای مختلف در این سیستم‌ها و همچنین آزمایشگاهی برای بررسی الگوریتم‌های یادگیری تقویتی در سیستم‌های چندعاملی.



**واژگان کلیدی:** سیستم‌های چندعاملی، یادگیری تقویتی، یادگیری تقویتی عمیق، ML-Agents

# فهرست مطالب

**عنوان** **صفحه**

# فهرست مطالب

**عنوان** **صفحه**

# فهرست مطالب

**عنوان** **صفحه**

# فهرست شکل‌ها

**عنوان** **صفحه**



# فهرست جداول

**عنوان** **صفحه**

## مخفف‌ها:

| | |
|---|---|
| RL | Reinforcement Learning |
| DRL | Deep Reinforcement Learning |
| PPO | Proximal Policy Optimization |
| MA-POCA | Multi-Agent Posthumous Credit Assignment |
| MARL | Multi-Agent Reinforcement Learning |

# فصل اول

# مقدمه

## ۱-۱- هدف پروژه

در جهانی که توسط تعاملات پیچیده و تصمیم‌گیری در محیط‌های پویا و اکثرا چندعاملی اداره می‌شود، همکاری مشترک و رقابت استراتژیک به‌عنوان یک حوزه جذاب برای پژوهش و بررسی شناخته می‌شود. این تعاملات تنها محدود به عامل‌های طبیعی همچون انسان‌ها نمی‌باشند و در سالیان گذشته، بانفوذ هرچه بیشتر عامل‌های هوشمند در زندگی روزمره انسان، نیاز به طراحی، ارزیابی و توانمندسازی الگوریتم‌ها و سیستم‌ها چندعاملی[1]، بیش‌ازپیش حس شده است. هدف از این پروژه سفر به قلمرو سیستم‌های چندعاملی و برداشتن گام‌هایی در توانمندسازی استفاده از یادگیری تقویتی عمیق[2] به‌منظور هوشمندسازی عامل‌ها در شناخت پیچیدگی روابط میان ویژگی‌ها و اجزای مختلف محیط و بهره‌برداری از آموخته‌های خود در جهت دستیابی به اهداف تعیین شده برای آن‌ها می‌باشد. توانمندسازی‌ای که امید است بیش‌ازپیش به کاربردهای دنیای واقعی راه یابد.

یکی از ابزارهایی که امکان تعلیم عامل‌های با استفاده از یادگیری تقویتی را به‌خوبی فراهم می‌کند، ML-Agents می‌باشد. این جعبه‌ابزار به‌خوبی با موتور بازی‌سازی Unity ادغام شده و شبیه‌سازی سناریوهای دنیای واقعی را به بهترین شکل میسر می‌سازد. علاوه بر این، این ابزار قدرتمند تعدادی از قدرتمندترین الگوریتم‌های یادگیری تقویتی عمیق را در اختیار پژوهشگران و بازی‌سازان می‌گذارد تا به‌وسیله آن‌ها عامل‌های هوشمند خود را تعلیم داده و یا به مطالعه سیستم‌ها یادگیری تقویتی و عامل‌های آن بپردازند.

[1] Multi-Agent Systems
[2] Deep Reinforcement Learning

در ایـــن پـــروژه نیـــز از ML-Agents اســـتفاده شـــده اســـت تـــا به‌وســـیله آن طراحـــی و پیاده‌ســـازی سیســـتم‌های چندعاملی موردمطالعه قـرار گیرنـد. بـه ایـن منظـور از بـازی والیبـال بـه‌عنوان دنیـای کـوچکی کـه مسـتلزم رقابـت و همکاری بین عامل‌هاست اسـتفاده شـده اسـت. در ایـن بـازی، عامل‌هـا (بازیکنـان) بـه‌منظور دسـتیابی بـه یـک هـدف مشـترک (پیـروزی) بایـد بـا هـم تیمـی خـود همکـاری کـرده و اسـتراتژی‌های مناسـبی را بـرای غلبـه بـر کنش‌هـای رقبـای خـود اتخـاذ کننـد. بنـابراین ایـن بـازی می‌توانـد بـه‌عنوان یـک محـیط مناسـب جهـت بررسـی ویژگی‌هـای سیســـتم‌های چنـــدعاملی، قابلیت‌هـــای جعبـــه‌ابزار ML-Agents و الگوریتم‌هـــای یـــادگیری تقـــویتی عمیـــق مورداستفاده قرار گیرد.

در واقع طراحـی و پیاده‌سـازی ایـن بـازی، یـک آزمایشـگاه مجـازی بـرای مطالعـه خصیصـه‌های مختلـف سیسـتم‌های چندعاملی و الگوریتم‌های یادگیری تقویتی مورداستفاده در آن‌ها را به وجود آورده است.

## ۱-۲- کاربرد پروژه

دامنـه ایـن پـروژه فراتـر از مرزهـای شبیه‌سـازی والیبـال بـوده و نتـایجی بـا اهمیـت گسـترده‌تر دارد. ایـن پـروژه بـه‌عنوان بسـتری بـرای بررسـی پیچیـدگی‌های کـار تیمـی و رقابـت اسـتراتژیک عمـل می‌کنـد. بسـتری کـه از نتـایج حاصـل از آن می‌تـوان بـرای کاربردهـای دنیـای واقعـی در زمینـه‌های متنـوعی ماننـد تعـاملات ربات‌هـا (ورزشـکار، امدادرسان و...)، خودروهای خودران، فرایندهای تولید، سیستم‌های حمل‌ونقل و... استفاده کرد.

مزیـت قابل‌توجـه انجـام تحقیقـات در چـارچوب یـک بـازی رایانـه‌ای در ایجـاد یـک محـیط آزمایشـی ایمـن و کنتـرل شده نهفته است. مشـابه بـا تقویـت مهارت‌هـای ضـروری در یـک شبیه‌سـازی قبـل از کـاربرد عملـی، ایـن پـروژه یـک کـاوش بـدون ریسـک و مقرون‌به‌صـرفه بـرای مطالعـه سیسـتم‌های چنـدعاملی و بـه‌ویژه الگوریتم‌هـای یـادگیری تقـویتی فـراهم می‌کنـد. عـلاوه بـر ایـن، ابزارهـای اسـتفاده شـده در ایـن پـروژه یعنـی ML-Agents و Unity، پتانسـیل شبیه‌سـازی سـناریوهایی دنیـای واقعـی را بادقـت بسـیار بـالا در بـر دارنـد. بنـابراین ایـن پـروژه گام‌هـای اولیـه‌ای در راسـتای توانمندسـازی ایجـاد مـدل‌های هوشـمند بـا اسـتفاده از یـادگیری تقـویتی بـرای عامل‌هـای دنیای واقعـی آن هـم در یـک فضـای شبیه‌سـاز بـر مـی‌دارد. بـه سـبب ایـن قابلیـت، هوشمندسـازی عامل‌هـا، بـه‌ویژه در محیط‌های پیچیده‌ای کـه مسـتلزم همکـاری و رقابـت بـین چنـد عامـل می‌باشـند، بـا هزینـه و زمـان بسـیار کمتـر و در شرایط کنترل‌شده‌تری ممکـن خواهـد بـود. در اصـل، ایـن پـروژه فراتـر از شبیه‌سـازی یـک بـازی والیبـال، گـامی به‌سوی پیشبرد درک ما از سیستم‌های چندعاملی و استفاده از آنها در زمینه‌های دنیای واقعی است.

## ۱-۳- ساختار پایان‌نامه

در ادامهٔ این پایان‌نامه، ابتدا به مفاهیم بنیادی زیربنای این پروژه پرداخته شده است. مفاهیمی که آشنایی با آن‌ها، پیش‌نیاز درک و آگاهی از آنچه که در این پژوهش اتفاق افتاده است می‌باشد. سپس در فصل سوم به بررسی برخی از پژوهش‌های و پروژه‌های سابق در زمینه سیستم‌های چندعاملی و ابزار ML-Agent پرداخته شده است. این بررسی، بینش‌هایی در مورد وضعیت سیستم‌های چندعاملی و کاربرد یادگیری تقویتی و یادگیری تقویتی عمیق در این زمینه ارائه می‌دهد. در فصل چهارم، گزارشی از نحوه و روند اجرای این پروژه به‌تفصیل ارائه شده است و برخی نتایج حاصل از پیاده‌سازی‌ها و آزمایش‌های انجام شده در فصل پنجم به تصویر کشیده و تشریح شده است. این بخش نتایج تجربی و یافته‌های حاصل از این پروژه را ارائه می‌دهد. در نهایت و در فصل پایانی، یافته‌های کلیدی و نتیجه‌گیری حاصل از این پژوهش، به همراه پیشنهاداتی برای تلاش‌های تحقیقاتی آینده ارائه شده است.

# فصل دوم
# مفاهیم

## ۲-۱- مقدمه

در این فصل به ارائه توضیح مختصر در مورد مفاهیم و ابزارهای مورداستفاده در این پروژه که آشنایی اولیه با آن‌ها لازمهٔ درک کار انجام شده در این پژوهش می‌باشد، پرداخته می‌شود. ابتدا توضیحاتی در مورد یادگیری تقویتی، بلوک‌های سازنده آن و مهم‌ترین مفاهیم و اهداف آن داده شده است. پس از آن مفهوم یادگیری تقویتی عمیق، علت نیاز به آن و چالش‌های اساسی‌اش مطرح شده است. در ادامه تعریف سیستم‌های چندعاملی و دو نوع اصلی از تعاملات موجود در این سیستم‌ها و همچنین مفهوم بازی با خود که یکی از تکنیک‌های یادگیری تقویتی است تشریح شده. در نهایت نیز توضیحاتی جهت آشنایی با موتور Unity و جعبه‌ابزار یادگیری ماشین آن، یعنی ML-Agent ارائه شده است.

## ۲-۲- یادگیری تقویتی

یادگیری تقویتی(RL) به‌عنوان یک الگوی اساسی یادگیری ماشین شناخته می‌شود که به چالش آموزش عامل‌های[1] هوشمند برای تصمیم‌گیری متوالی در محیط‌های مختلف می‌پردازد. برگرفته از اصول روان‌شناسی رفتاری، یادگیری تقویتی بر توانمندسازی عامل‌ها برای یادگیری از طریق آزمایش و سازگاری تمرکز دارد. این روش یادگیری ماشین، در هسته خود شامل تعاملات بین عوامل و محیط است. جایی که عامل‌ها

[1] Agents

کنش‌هایی[1] را انجام می‌دهند، نتایج را درک می‌کنند و بازخورد را به شکل پاداش[2] دریافت می‌کنند. این تعامل باهدف تجهیز عوامل به خط‌مشی‌هایی[3] است که پاداش‌های تجمعی[4] را در طول زمان بهینه می‌کنند و عامل‌ها را با اهداف تعریف شده برای آنها همسو می‌سازند [۱].

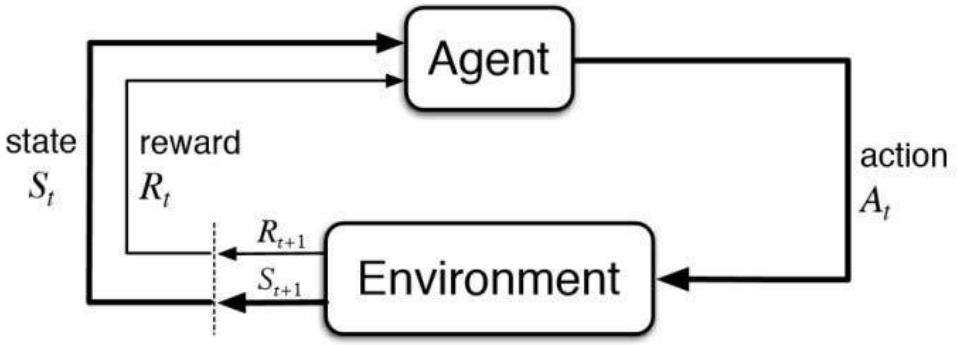
### ۲-۲-۱- اجزای پایه‌ای یادگیری تقویتی

در یادگیری تقویتی چندین جزء اساسی وجود دارد که بلوک‌های سازنده این رشته را تشکیل می‌دهند. آشنایی با این اجزا، لازمه درک هر سیستم دربردارنده یادگیری تقویتی می‌باشد.

- **عامل:** عامل تصمیم‌گیرنده مستقلی است که با محیط تعامل دارد. هدف عوامل دستیابی به اهداف یا وظایف خاص تعریف شده در محیط از طریق مجموعه‌ای از اقدامات می‌باشد.
- **محیط:** محیط شامل شرایطی است که عامل در آن تصمیم‌گیری و عمل می‌کند. محیط حالات و قوانین را تعریف می‌کند و زمینه یادگیری و تعامل عامل‌ها را فراهم می‌سازد.
- **حالت[5]:** حالت شرایط فعلی محیط را نشان می‌دهند. حالت‌ها تمام اطلاعات لازم یک عامل برای یک تصمیم‌گیری آگاهانه را در بر می‌گیرند.
- **کنش:** کنش‌ها اقداماتی هستند که توسط عامل‌ها برای انتقال بین حالت‌ها انجام می‌شود. اعمال یک عامل بر وضعیت و پاداش بعدی تأثیر می‌گذارد و به دنباله‌ای از کنش‌ها و واکنش‌ها منجر می‌شود.
- **پاداش:** پاداش‌ها به عوامل بازخورد فوری درباره مطلوبیت اقداماتشان ارائه می‌دهند. پاداش‌های مثبت، اقدامات مطلوب را تشویق می‌کنند. درحالی‌که پاداش‌های منفی، اقدامات نامطلوب را سرزنش می‌کنند.
- **خط‌مشی:** خط‌مشی‌ها راهبردهایی را تعریف می‌کنند که توسط آنها، عامل‌ها اقداماتی را در حالت‌های مختلف انتخاب می‌کنند. هدف RL اغلب شناسایی خط‌مشی بهینه‌ای است که پاداش تجمعی مورد انتظار را در طول زمان به حداکثر می‌رساند.

شکل ۱-۱ بیانی ساده از نحوه ارتباط میان این اجزا را نشان می‌دهد.

[1] Actions
[2] Reward
[3] Policies
[4] Cumulative Rewards
[5] State




شکل ۲-۱: معماری یک سیستم یادگیری تقویتی

### ۲-۲-۲- اکتشاف[1] و بهره‌برداری[2]

دوراهی اکتشاف-بهره‌برداری یک چالش اساسی در یادگیری تقویتی است که از عوامل می‌خواهد بین جستجوی دانش جدید و بهره‌برداری از دانش موجود تعادل برقرار کنند. اکتشاف شامل انتخاب اقداماتی است که ممکن است نتایج ناآشنا به همراه داشته باشد. این امر برای عوامل، در کشف اقدامات و حالات بهینه بالقوه، و همچنین اجتناب از تله بهره‌برداری مداوم از انتخاب‌های نیمه‌بهینه[3] بسیار مهم است. از سوی دیگر، بهره‌برداری بر استفاده از دانش فعلی عامل برای به حداکثر رساندن پاداش‌های فوری متمرکز است. بهره‌برداری از اقداماتی استفاده می‌کند که به‌خوبی شناخته شده‌اند و احتمالاً نتایج مطلوبی به همراه خواهند داشت [۲].

دستیابی به تعادل مناسب بین اکتشاف و بهره‌برداری برای یادگیری مؤثر بسیار مهم است. عواملی که صرفاً از اقدامات شناخته شده بهره‌برداری می‌کنند، ممکن است جایگزین‌های بهتری را از دست بدهند. درحالی‌که آنهایی که به طور مشخصی کاوش می‌کنند، ممکن است به پیشرفت‌های معناداری دست یابند.

### ۲-۲-۳- تخمین ارزش[4] و بهینه‌سازی خط‌مشی[5]

دو هدف اصلی در RL تخمین ارزش و بهینه‌سازی خط‌مشی است که هر کدام جنبه‌های متفاوتی از فرایند یادگیری را موردتوجه قرار می‌دهند.

[1] Exploration
[2] Exploitation
[3] Suboptimal
[4] Value Estimation
[5] Policy Optimization

- **تخمین ارزش:**

تخمین ارزش بر کمی‌کردن مطلوبیت جفت‌های حالت-اقدام (میزان پاداش تجمعی حاصل از انتخاب هر اقدام در حالتی خاص از محیط) بر حسب پاداش‌های تجمعی مورد انتظار تمرکز دارد. روش‌هایی مانند برنامه‌نویسی پویا[1]، تکنیک‌های مونت‌کارلو[2]، و یادگیری تفاوت زمانی[3] در تخمین ارزش جفت‌های حالت-اقدام و انتخاب عمل‌های بهینه‌تر استفاده می‌شوند.

- **بهینه‌سازی خط‌مشی:**

هدف بهینه‌سازی خط‌مشی تعیین بهترین استراتژی‌ای (سیاستی) است که یک عامل باید برای به حداکثر رساندن پاداش‌های تجمعی دنبال کند. تکنیک‌هایی مانند گرادیان‌های خط‌مشی[4]، عوامل را قادر می‌سازد تا مستقیماً خط‌مشی‌های خود را با تنظیم پارامترها در جهت دستیابی به پاداش بیشتر تقویت کنند.

## ۲-۳- یادگیری تقویتی عمیق

یادگیری تقویتی عمیق(Deep RL) یک توسعه پیشرفته از یادگیری تقویتی سنتی استفاده از قدرت شبکه‌های عصبی عمیق نشان می‌باشد. با ترکیب قابلیت‌های یادگیری شبکه‌های عصبی با چارچوب تصمیم‌گیری RL، یادگیری تقویتی عمیق پیشرفت‌های قابل‌توجهی را در پرداختن به وظایف و محیط‌های پیچیده و با ابعاد بزرگ که قبلاً برای روش‌های معمولی RL چالش‌برانگیز بودند، ممکن کرده است [۳].

انگیزه پشت Deep RL از درک این موضوع ناشی می‌شود که بسیاری از مشکلات دنیای واقعی شامل فضاهای حالت گسترده و پیچیده است. جایی که روش‌های سنتی جدولی برای به مشکل بر می‌خورند. شبکه‌های عصبی عمیق توانایی تقریب توابع پیچیده را ارائه می‌دهند و به عوامل اجازه می‌دهند از مجموعه‌داده‌های بزرگ یاد بگیرند و برای حالاتی که پیش از این مشاهده نشده‌اند عمل تعمیم‌دهی را انجام دهند.

[1] Dynamic Programming
[2] Monte Carlo
[3] Temporal Difference Learning
[4] Policy Gradients

### ۲-۳-۱- تقریب تابع در یادگیری تقویتی عمیق

تقریب تابع شامل مدل‌سازی روابط پیچیده بین ورودی‌ها (حالت‌ها) و خروجی‌ها (اقدامات) برای حالات مشاهده نشده می‌باشد و استفاده از شبکه‌های عصبی عمیق یکی از روش‌های آن است. در Deep RL، این شبکه‌ها از داده‌های مشاهده‌شده برای پیش‌بینی ارزش[1] یا خط‌مشی برای حالت‌های دیده نشده استفاده می‌کنند که به آن تعمیم‌سازی[2] گفته می‌شود. این امر به‌ویژه هنگام برخورد با فضاهای حالت با ابعاد بسیار بزرگ و یا پیوسته مفید است، جایی که تعیین ارزش برای هر حالت غیرممکن است. توانایی شبکه‌های عصبی در شناخت الگوهای پیچیده در داده‌ها، به عامل اجازه می‌دهد تا به طور مؤثر در محیط‌های پیچیده فعالیت کند و بر اساس بازنمایی‌های آموخته شده تصمیمات آگاهانه بگیرد [۴].

### ۲-۳-۲- چالش‌های یادگیری تقویتی عمیق

درحالی‌که یادگیری تقویتی عمیق قابلیت‌های قابل‌توجهی را ارائه می‌دهد، چالش‌هایی را نیز نسبت به روش‌های سنتی RL معرفی می‌کند که باید به آنها توجه شود.

- **بهره‌وری نمونه[3]:**

آموزش شبکه‌های عمیق به مقادیر قابل‌توجهی داده نیاز دارد که می‌تواند در سناریوهای دنیای واقعی بازدارنده باشد.

- **کاوش:**

یادگیری تقویتی عمیق به دلیل فضاهای حالت با ابعاد بالا با چالش اکتشاف-بهره‌برداری شدیدتر مواجه می‌شود و ایجاد تعادل بین اکتشاف و بهره‌برداری همچنان یک نگرانی حیاتی است.

- **تعمیم:**

مدل‌های Deep RL باید به‌خوبی برای موقعیت‌های نادیده تعمیم‌سازی را انجام دهند و این امر در عملکرد صحیح عامل بسیار حیاتی است.

[1] Value
[2] Generalization
[3] Sample Efficiency

### ۲-۳-۳- الگوریتم PPO

بهینه‌سازی خط‌مشی مبدائی[1] (PPO) یک الگوریتم یادگیری تقویتی پرکاربرد است که به کلاس روش‌های بهینه‌سازی خط‌مشی تعلق دارد [۵]. این الگوریتم به دلیل پایداری[2] و عملکرد قوی در آموزش عوامل برای کارهای مختلف محبوبیت پیدا کرده است. PPO چالش بهینه‌سازی خط‌مشی‌ها را به‌گونه‌ای حل می‌کند که اکتشاف (آزمایش اقدامات جدید) و بهره‌برداری (انتخاب بهترین اقدامات شناخته‌شده) را متعادل می‌سازد و درعین‌حال ثبات فرایند یادگیری را تضمین می‌کند.

PPO با جمع‌آوری مکرر تجربیات از طریق تعامل با محیط و استفاده از این تجربیات برای به‌روزرسانی خط‌مشی عامل عمل می‌کند. یکی از ویژگی‌های کلیدی آن این است که به‌روزرسانی خط‌مشی را به محدوده خاصی محدود می‌کند و از تغییرات بیش از حد بزرگ که می‌تواند منجر به بی‌ثباتی شود جلوگیری می‌کند. این مکانیسم همراه با سایر پیشرفت‌ها، به PPO اجازه می‌دهد تا در مقایسه با روش‌های گرادیان خط‌مشی قبلی، به فرایند یادگیری پایدارتر و کارآمدتری دست یابد. PPO در طیف وسیعی از حوزه‌ها کاربرد پیدا کرده است و به ابزاری اساسی در جعبه‌ابزار یادگیری تقویتی تبدیل شده است.

### ۲-۳-۴- الگوریتم MA-POCA

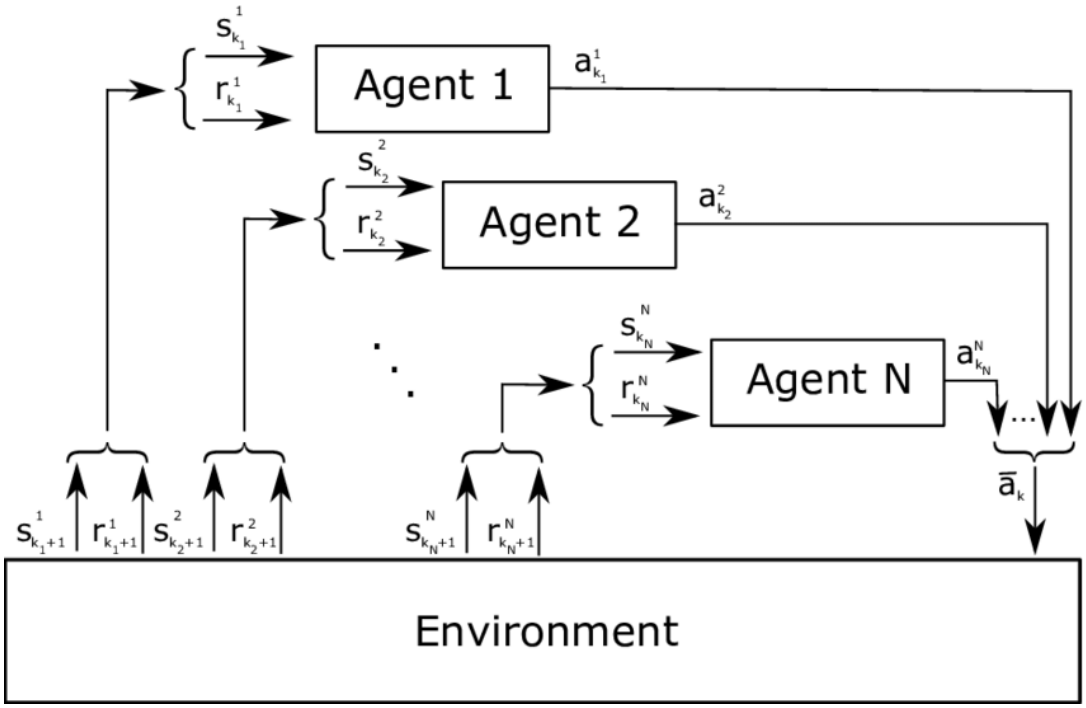
بهینه سازی خط‌مشی چند عاملی با اقدامات جمعی[3] (MA-POCA)، که اغلب به‌عنوان POCA شناخته می‌شود، یک الگوریتم یادگیری تقویتی پیشرفته است که به طور خاص برای سیستم‌های چندعاملی طراحی شده است [۶]. این الگوریتم به چالش‌های ناشی از سناریوهای چندعاملی مشارکتی می‌پردازد که در آن چندین عامل باید برای دستیابی به اهداف مشترک با یکدیگر همکاری کنند. POCA به‌ویژه برای مشکلاتی مناسب است که در آن عوامل باید اقدامات خود را برای به حداکثر رساندن پاداش‌های جمعی هماهنگ کنند و درعین‌حال تأثیر تصمیمات فردی خود را بر عملکرد کلی در نظر بگیرند.

POCA با قادر ساختن عوامل برای استدلال در مورد اقدامات خود به طور جمعی، هماهنگی و همکاری بین عوامل را افزایش می‌دهد و منجر به تصمیم‌گیری کارآمدتر و مؤثرتر در محیط‌های پیچیده چندعاملی می‌شود.

[1] Proximal Policy Optimization
[2] Stability
[3] Multi-Agent Posthumous Credit Assignment

## ۲-۴- سیستم‌های چندعاملی

سیستم‌های چندعاملی(MAS) یک حوزه مهم در هوش مصنوعی را تشکیل می‌دهند که به تعاملات، همکاری‌ها و رقابت‌هایی می‌پردازد که بین چندین عامل، در محیط‌های مشترک رخ می‌دهد. بر خلاف سناریوهای تک عاملی که در آن یک عامل تنها به‌صورت مجزا عمل می‌کند، سیستم‌های چندعاملی پویایی پیچیده ناشی از وابستگی متقابل و تأثیرات متقابل عوامل را در بر می‌گیرند. این سیستم‌ها زمینه مناسبی را برای بررسی چگونگی ترکیب تصمیمات فردی برای تولید نتایج جمعی و درک پیچیدگی‌های تعاملات مشارکتی و رقابتی فراهم می‌کنند. شکل ۱-۲ نمایشی از یک سیستم چندعاملی مبتنی بر یادگیری تقویتی و نحوه تعامل اجزای آن می‌باشد [۷].




شکل ۲-۲: معماری یک سیستم یادگیری تقویتی چندعاملی

### ۲-۴-۱- همکاری[1] و رقابت[2]

تعاملات چندعاملی را می‌توان به دو دسته اصلی تقسیم کرد: همکاری و رقابت. این تعاملات در مجموع رفتار جمعی سیستم را شکل می‌دهند.

[1] Cooperation
[2] Competition

- **همکاری:**

تعاملات مشارکتی بر همکاری بین عوامل برای دستیابی به اهداف مشترکی که ممکن است به‌صورت فردی قابل‌دستیابی نباشد تأکید می‌کند. عوامل درگیر در همکاری اغلب مبادله اطلاعات یا انجام فعالیت‌ها را برای ارتقای عملکرد خود به‌صورت جمعی انجام می‌دهند. این هم‌افزایی مشترک به عوامل اجازه می‌دهد تا از نقاط قوت مکمل استفاده کنند و با از پس چالش‌هایی که فراتر از توانایی‌های یک تک عامل هستند برآیند.

- **رقابت:**

در طرف دیگر، تعاملات رقابتی زمانی آشکار می‌شود که عوامل اهداف متضاد را دنبال کنند یا برای منابع محدود رقابت کنند. در سناریوهای رقابتی، عوامل به طور استراتژیک اقدامات خود را برای برتری بر دیگران و کسب مزیت بهینه می‌کنند. رقابت، عوامل را وادار می‌کند تا حرکات حریفان را پیش‌بینی کنند، منابع را به‌صورت استراتژیک تخصیص دهند، و تاکتیک‌هایی را برای دستیابی به اهداف شخصی و درعین‌حال در نظر گرفتن استراتژی‌های سایر عوامل، طراحی کنند.

## ۲-۵- بازی با خود[1] در یادگیری تقویتی:

بازی با خود تکنیکی در یادگیری تقویتی است که در آن یک عامل با رقابت با خودش یاد می‌گیرد. در این روش، به‌جای استفاده از حریفان خارجی یا مجموعه داده‌های ثابت برای آموزش، عامل با نسخه‌های قبلی خود تعامل دارد. این فرایند به عامل اجازه می‌دهد تا به‌تدریج عملکرد خود را از طریق رقابت با خود و یادگیری مکرر بهبود بخشد.

در این روش، عامل با یک خط‌مشی اولیه شروع می‌کند و بازی را انجام می‌دهد یا وظایف را بر اساس آن خط‌مشی کامل می‌کند. اقدامات گذشته خود را به‌عنوان اقدامات حریف در نظر می‌گیرد و با این رقیبان شبیه‌سازی شده تعامل دارد و تجارب به‌دست‌آمده از بازی در برابر اقدامات گذشته خود، برای یادگیری استفاده می‌شود. عامل داده‌ها را از این تعاملات جمع‌آوری می‌کند، از جمله اقدامات خود، اقدامات حریف (که اقدامات گذشته خودش هستند)، و پاداش‌های مربوطه. سپس عامل با استفاده از این داده‌ها خط‌مشی خود را به‌روز می‌کند تا استراتژی‌های خود را اصلاح کند. این فرایند تکراری همچنان ادامه می‌یابد و عامل در

[1] Self-play

مقابل سیاست‌های در حال تحول خود بازی می‌کند و از موفقیت‌ها و شکست‌های خود برای بهبود عملکرد خود در طول زمان درس می‌گیرد.

## Unity -۶-۲

یونیتی یک پلتفرم توسعه بازی جامع است که ابزارهایی را برای ایجاد محیط‌های مجازی ارائه می‌دهد. این موتور یک چارچوب همه‌کاره برای طراحی، شبیه‌سازی و تجسم سناریوهای متنوع ارائه می‌کند و آن را به یک پلتفرم ایده‌آل برای ایجاد محیط‌هایی تبدیل می‌کند که در آن عوامل می‌توانند در تعامل، یادگیری و تصمیم‌گیری باشند. توانایی یونیتی برای ایجاد جهان‌های سه‌بعدی واقع‌گرایانه با عناصر تعاملی، منبع ارزشمندی برای توسعه محیط‌های پیچیده و چالش‌برانگیز است که بسیار شبیه سناریوهای دنیای واقعی است. زبان برنامه‌نویسی مورد پشتیبانی این موتور C# است.

## ML-Agents -۷-۲

جعبه‌ابزار ML-Agents تلفیقی پیشگامانه از پلتفرم توسعه بازی یونیتی و تکنیک‌های پیشرفته یادگیری ماشین، به‌ویژه یادگیری تقویتی عمیق را ارائه می‌دهد [۸]. این ادغام دامنه‌ای از امکانات را برای آموزش و توانمندسازی عوامل برای یادگیری، تطبیق و تعالی در محیط‌های مختلف باز می‌کند.

ML-Agents با پر کردن شکاف بین توسعه بازی و یادگیری ماشین، فرایند توسعه عوامل هوشمندی را که قادر به تسلط بر وظایف پیچیده، شامل تصمیم‌گیری پیچیده، تعامل محیطی و همکاری و رقابت با سایر عوامل هستند، متحول می‌کند. جعبه‌ابزار ML-Agents به توسعه‌دهندگان این امکان را می‌دهد که محیط‌های مجازی ایجاد کنند که نه‌تنها از نظر بصری مجذوب‌کننده هستند، بلکه به‌شدت به سناریوهای دنیای واقعی نزدیک هستند. این محیط‌ها به‌عنوان زمینه‌های آموزشی برای عوامل عمل می‌کنند. فرصت‌هایی را برای تعامل در محیط‌های چالش‌برانگیز و برقراری ارتباط با سایر عوامل، و پاسخ به طیف وسیعی از محرک‌های پویا ارائه می‌دهند. الگوریتم‌های یادگیری تقویتی عمیق تعبیه شده در ML-Agents ، عوامل را قادر می‌سازد تا استراتژی‌هایی را یاد بگیرند که از رفتارهای مبتنی بر قوانین ساده فراتر می‌روند و آنها را برای تصمیم‌گیری آگاهانه در موقعیت‌هایی که نیاز به سازگاری، برنامه‌ریزی استراتژیک و تعاملات ظریف دارند، مجهز می‌کنند.

یکی از مزایای اصلی ML-Agents در ظرفیت آن برای افزایش کارایی یادگیری برای سناریوهایی جهان واقعی‌ای است که آزمایش مستقیم آن‌ها در دنیای واقعی ممکن است غیرعملی یا محدود باشد. ML-Agents با اجازه‌دادن به عوامل برای جمع‌آوری تجربیات فراوان در محیط‌های مجازی، فرایند یادگیری را تسریع می‌کند و به عوامل قدرت می‌دهد تا بر کارهایی که نیازمند کاوش و آزمون و خطای گسترده هستند، تسلط پیدا کنند. علاوه بر این، این جعبه‌ابزار توسعه مهارت‌های قابل‌انتقال را تسهیل می‌کند و عوامل آموزش‌دیده در محیط‌های مجازی را قادر می‌سازد تا رفتارهای آموخته‌شده خود را به برنامه‌های دنیای واقعی با سازگاری‌های مناسب تعمیم دهند. به بیان ساده، این جعبه‌ابزار محدودیت‌های تعلیم عامل‌های هوشمند برای کاربردهای دنیای واقعی را از طریق استفاده از محیط شبیه‌ساز ارائه شده توسط یونیتی، از پیشرو بر می‌دارد و امکان تعلیم عامل‌ها را با هزینه و زمان بسیار کمتر فراهم می‌سازد.

## ۲-۸- سیستم امتیازدهی ELO

سیستم امتیازدهی ELO روشی است که به طور گسترده برای ارزیابی و کمی‌کردن سطح مهارت نسبی بازیکنان در بازی‌های دو نفره مانند شطرنج و ورزش‌های رقابتی استفاده می‌شود. سیستم ELO مبتنی بر این ایده است که نتیجه مسابقه بین دو بازیکن به تفاوت در سطح مهارت آنها بستگی دارد. به بازیکنان یک امتیاز اولیه اختصاص داده می‌شود و پس از هر بازی، رتبه‌بندی آنها بر اساس نتیجه بازی تنظیم می‌شود. اگر بازیکنی در مقابل حریف قوی‌تر پیروز شود، امتیاز بیشتری نسبت به پیروزی بر حریف ضعیف‌تر کسب می‌کند. برعکس، باخت به حریف ضعیف‌تر منجر به از دست دادن امتیاز بیشتر می‌شود. این سیستم یک نمایش عددی از مهارت یک بازیکن ارائه می‌دهد که می‌تواند برای رتبه‌بندی استفاده شود.

سیستم امتیازدهی ELO به‌ویژه در زمینه بازی‌های self-play، جایی که یک عامل برای بهبود عملکرد خود با خودش رقابت می‌کند، قابل‌استفاده است. در این موارد ELO را می‌توان برای ردیابی پیشرفت عامل و ارزیابی نحوه تکامل عملکرد آن در طول زمان به کاربرد. هنگامی که عامل درگیر self-play می‌شود، تاریخچه‌ای از بازی‌ها را جمع‌آوری می‌کند و رتبه‌بندی‌های ELO به‌روز می‌شوند تا سطح مهارت کسب شده عامل را منعکس کنند. این امر به عامل اجازه می‌دهد تا به طور مداوم استراتژی‌های خود را تطبیق داده و بهینه‌سازی کند و اطمینان حاصل کند که همیشه با سطوح مناسبی از چالش مواجه است.

در یادگیری تقویتی، در بازی‌های خصمانه[1] و محیط‌های رقابتی‌ای که عوامل در آن بر سر یک منبع مشترک رقابت می‌کنند و در آن‌ها self-play وجود دارد، پاداش اختصاص‌داده‌شده به عامل‌های رقیب قرینه هم بوده

[1] Adversarial

و بنـابراین پـاداش تجمعـی در هـر قسـمت از بـازی برابـر صـفر خواهـد بـود. از ایـن رو نمی‌تـوان از پـاداش تجمعـی بـه‌عنوان معیـاری بـرای رصـد میـزان یـادگیری عوامـل اسـتفاده کـرد. در ایـن مـوارد سیسـتم امتیـازدهی ElO ایـن امکان را می‌دهد تا میزان یادگیری عوامل را از طریق نمایش ارتقای سطح مهارت آن‌ها مشاهده کرد.

## ۲-۹- جمع‌بندی

در این فصل بـا عمـده مفـاهیم زیربنـایی ایـن پـروژه آشـنا شـدیم. مفـاهیمی کـه از هـر یـک در برخـی از قسـمت‌های مختلـف پـروژه اسـتفاده شـده اسـت و شـناخت نسـبی آن‌هـا، لازمـهٔ درک کارهـای انجـام شـده در ایـن پـروژه می‌باشد.

# فصل سوم
# مرور کارهای پیشین

## ۳-۱- مقدمه

در فصل مقدمه به اهمیت سیستم‌های چندعاملی و استفاده از یادگیری تقویتی در آن پرداخته شد. پس از درک ویژگی‌های چنین سیستم‌هایی با مطالب ارائه شده در فصل ۲، در این فصل چندی از مهم‌ترین حوزه‌های پژوهشی مطرح در این شاخه و مثال‌هایی از کارهای انجام شده در آن‌ها بیان خواهد شد. پس از آن نیز برخی از پروژه‌ها و پژوهش‌های مشابه با این پروژه بررسی خواهند شد.

## ۳-۲- حوزه‌های پژوهشی

به علت پیچیدگی محیط در سیستم‌های چندعاملی و وجود تعداد زیادی از وظایف، برنامه‌ریزی عامل‌ها در این محیط سخت و در عمل غیرممکن است؛ بنابراین در چنین محیط‌هایی عامل‌ها باید بتوانند راه‌حل مسائل را خودشان یاد بگیرند. این ضرورت باعث می‌شود تا بخش مهمی از پژوهش‌ها در حوزه سیستم‌های چندعاملی با تمرکز بر یادگیری تقویتی باشد. در این بخش برخی از کارهای انجام شده در حوزه یادگیری تقویتی چندعاملی[1] یا MARL به طور مختصر بررسی می‌شود.

در بسیاری از کاربردها، عامل‌ها همکار یا رقیب یکدیگرند و به همین دلیل پژوهش‌های زیادی در این راستا صورت‌گرفته است. در یکی از پژوهش‌هایی که در این زمینه صورت‌گرفته، پژوهشگران با الهام‌گرفتن از انسان و مغز، توانسته‌اند توانایی عامل‌ها را در همکاری و رقابت افزایش دهند [۹]. وقتی که انسان به‌صورت

[1] Multi-Agent Reinforcement Learning

اجتماعی به تصمیم‌گیری می‌پردازد، نظریهٔ ذهن برای بهینه‌کردن تصمیم‌گیری، حالت ذهنی دیگر افراد را نیز در تصمیم‌گیری دخیل می‌کند. در این حالت، حالت ذهنی دیگر افراد بر اساس مشاهدات یا دانش قبلی در دسترس است. پژوهش ذکر شده، با الهام‌گرفتن از این نظریه، چنین حالتی را برای عامل‌ها در سیستم چندعاملی با یادگیری تقویتی پیاده‌سازی کرده است. در این سیستم، عامل‌ها در هر تصمیم‌گیری، رفتار دیگر عامل‌ها را نیز پیش‌بینی می‌کنند. دستهٔ دیگر از پژوهش‌ها که در این طبقه قرار می‌گیرند، پژوهش‌های مشابه با این پروژه‌اند که در بخش بعد مورد بررسی قرار می‌گیرند.

یکی دیگر از موضوعات مهم و پر طرفدار در این حوزه، مسئله اکتشاف می‌باشد. در یکی از مقالات مروری در این زمینه، پژوهشگران با اشاره این که یکی از چالش‌های یادگیری تقویتی عمیق و MLAR مسائل مربوط به اکتشاف است، به بررسی روش‌های مختلف اکتشاف پرداخته‌اند [۱۰]. یک پژوهش صورت‌گرفته دیگر در زمینه اکتشاف در مسائل یادگیری تقویتی عمیق چندعاملی، بیان می‌کند که بسیاری از روش‌های موجود برای اکتشاف، مشکل مشابهی دارند به‌طوری‌که برای عامل‌ها بسیار دشوار است که حالتی که ارزش اکتشاف دارد را پیدا کنند و اکتشاف آن‌ها با یکدیگر هماهنگ شوند [۱۱]. آن‌ها برای حل این مشکل، اکتشاف همکارانه چندعاملی را پیشنهاد داده‌اند.

پژوهش‌های متعددی نیز در زمینه طراحی سیگنال پاداش صورت‌گرفته است. یکی از این پژوهش‌ها، طراحی پاداش برای سیستم‌های همکارانهٔ چندعاملی مسیریابی بسته‌ها بررسی می‌کند [۱۲]. این پژوهش ابتدا نشان می‌دهد که پاداش‌های تیمی و فردی می‌توانند سیاست‌های نا بهینه تولید کنند. سپس به طراحی سیگنال پاداش مختلطی می‌پردازد که در آزمایش این مقاله بهترین نتیجه را کسب می‌کند.

تعداد زیادی از پژوهش‌ها نیز به کاربردهای دنیای واقعی مربوط می‌شوند. به طور مثال یکی از این پژوهش‌ها، به حل مسئلهٔ کنترل سیگنال ترافیک در معیار بزرگ پرداخته است [۱۳]. در پژوهشی دیگر کاربرد MARL در آیندهٔ اینترنت بررسی شده است این پژوهش بیان می‌کند که هرچند الگوریتم‌های استاندارد سنتی استفاده شده‌اند تا با درنظرگرفتن هر موجودیت شبکه به‌عنوان یک عامل، به آن این امکان را بدهند که تصمیم‌های بهینه بگیرد، این مدل‌ها در برخی از موارد نارسایی داشته‌اند زیرا نتوانسته‌اند مواردی مانند همکاری یا رقابت در موجودیت‌ها را پیاده‌سازی کنند و عامل‌ها در این روش‌ها با هر موجودیت دیگر مانند بخشی از محیط برخورد می‌کنند [۱۴]. درصورتی‌که استفاده از MARL سبب می‌شود که هر موجودیت شبکه بتواند سیاست بهینه خود را با بررسی محیط شبکه و سیاست دیگر موجودیت‌ها پیدا کند. این پژوهش باتوجه‌به نقش مهم MARL در افزایش بهینگی عملکرد موجودیت‌های شبکه، به‌صورت مروری کاربرد MARL را در نسل جدید اینترنت بررسی می‌کند.

## ۳-۳- کارهای مشابه قبلی

در ادامه چند نمونه از پژوهش‌های صورت‌گرفته در حوزه یادگیری تقویتی چندعاملی و با استفاده از ابزار ML-Agents، شرح داده شده است.

یکی از پژوهش‌های مشابه با این پروژه، پژوهشی است که در آن دو عامل با استفاده از self-play به یادگیری تنیس می‌پردازند [۱۵]. در این پژوهش از الگوریتم PPO استفاده شده است و چند حالت مختلف از پیکربندی آن تست و مقایسه شده است. همچنین یادگیری دوره‌ای[1] نیز برای عوامل پیاده‌سازی و نتایج آن بررسی شده است.

پژوهش دیگری به مقایسهٔ الگوریتم‌های PPO و SAC در یادگیری تقویتی سیستم‌های چندعاملی با استفاده از Ml-Agents پرداخته است [۱۶]. این پژوهش در قالب مسئله‌ای که در آن عوامل برای جمع‌آوری غذا همکاری و رقابت دارند به بررسی کارایی و سازگاری این الگوریتم‌ها در محیط‌های پویا می‌پردازد.

پژوهشی که بیشترین شباهت را از لحاظ محیط بازی به این پروژه دارد، یک شبیه‌ساز والیبال است [۱۷]. در این پژوهش تیم‌ها تنها ۱ عامل دارند فضای بازی تنها رقابتی است و همچنین کنش‌های تعبیه شده در آن، به‌مراتب کمتر از کنش‌های موجود در این پروژه است. همچنین قوانین بازی نیز بسیار ساده طراحی شده است.

## ۳-۴- جمع‌بندی

در این فصل چند حوزه داغ پژوهشی در زمینه یادگیری تقویتی چندعاملی و خلاصه‌ای بسیار کوتاه از برخی کارهای انجام شده در این حوزه‌ها ارائه شد. سپس ۳ نمونه از پژوهش‌های انجام شده در محیط Unity و با استفاده از ML-Agents ذکر شد. وجه تمایز این پروژه با اکثر قریب به‌اتفاق پروژه‌های پیاده‌سازی شده با استفاده از این ابزار، وجود همکاری علاوه بر رقابت می‌باشد. همچنین در این پروژه از الگوریتم MA-POCA که مناسب یادگیری تیمی می‌باشد استفاده شده و نتایج حاصل از آن با الگوریتم PPO مقایسه شده است.

[1] Curriculum Learning

# فصل چهارم
# طراحی و پیاده‌سازی بازی

## ۴-۱- مقدمه

در فصل گذشته، مفاهیم پیش‌نیاز برای درک کار انجام شده در این پروژه بیان شد. در این فصل شرحی از فعالیت‌های صورت‌گرفته و مسیر انجام پروژه ارائه می‌شود. در ابتدا ابزارهای استفاده شده و نحوه نصب ML-Agents و سپس نحوه طراحی و اجزای محیط یادگیری و همچنین تکنیک‌های استفاده شده برای افزایش سرعت یادگیری توضیح داده شده‌اند. در ادامه طراحی اجزای سیستم بازی از قبیل منطق آن، مشاهدات، کنش‌ها و سیگنال پاداش و پس از آن نیز مراحل تکاملی طراحی بازی و تعلیم عامل‌ها تشریح شده است. در انتها نیز علاوه بر توضیح تنظیمات پیکربندی استفاده شده برای نسخه نهایی بازی، در مورداستفاده از PPO به‌جای MA-POCA صحبت شده است.

## ۴-۲- ابزارهای استفاده شده

آنچه در پیاده‌سازی این پروژه از آن استفاده شده است، Unity و جعبه‌ابزار یادگیری ماشین آن یعنی ML-Agent می‌باشد. معرفی این دو ابزار پیش‌تر و در بخش مفاهیم ارائه شده است. نکته حائز اهمیت راه‌اندازی و نصب ML-Agnrts می‌باشد که در ادامه به طور مختصر بهترین روش این کار شرح داده شده است تا در صورت نیاز، توسعه‌دهندگان و پژوهشگران بتوانند از آن استفاده کنند. چرا که خیلی از روش‌های موجود ممکن است موجب نصب ناقص و یا اشتباه این جعبه‌ابزار شود و به علت وجود وابستگی‌های موجود بین نسخه‌های خاصی از کتابخانه‌های موردنیاز، دشواری‌هایی در فرایند نصب به وجود آید.

در این پروژه از Release 20 این جعبه‌ابزار استفاده شده است. نکته که در هنگام نصب باید به آن توجه کرد آن است که برخلاف راهنمای رسمی نصب این ابزار که پایتون نسخه ۳.۷.۲ را برای آن پیشنهاد داده است، این ابزار با نسخه‌های پایتون ۳.۱۰.x و بالاتر از آن سازگاری ندارد. در این پروژه از نسخه ۳.۹.۹ استفاده شده است.

برای نصب ML-Agnrts از روش پیشنهادی، ابتدا باید ریپازیتوری آن را که آدرس آن در پیوست ۱ قرار داده شده است، کلون کرد. سپس باید یک پروژه جدید در یونیتی تعریف کرد و پکیج com.unity.ml-agents را از مسیر Window -> Package Manager -> Add package from disk... و از پوشه‌ای که فایل‌های ML-Agnrts در آن کلون شده است نصب کرد. همین روند را برای پکیج com.unity.ml-agents.extensions نیز باید طی کرد. پس از طی مراحل مذکور باید به سراغ نصب PyTorch و پکیج‌های پایتون این جعبه‌ابزار رفت. برای جلوگیری از بروز هرگونه تضاد میان نسخه‌های نصب شده از کتابخانه‌های مختلف پایتون بر روی سیستم، باید از یک محیط مجازی برای نصب موارد مذکور استفاده کرد. پس از ایجاد و فعال‌سازی یک محیط مجازی، باید دستورات زیر را به همین ترتیب در خط فرمان وارد کرد.

```
pip3 install torch -f https://download.pytorch.org/whl/torch_stable.html
pip3 install -e ./ml-agents-envs
pip3 install -e ./ml-agents
```

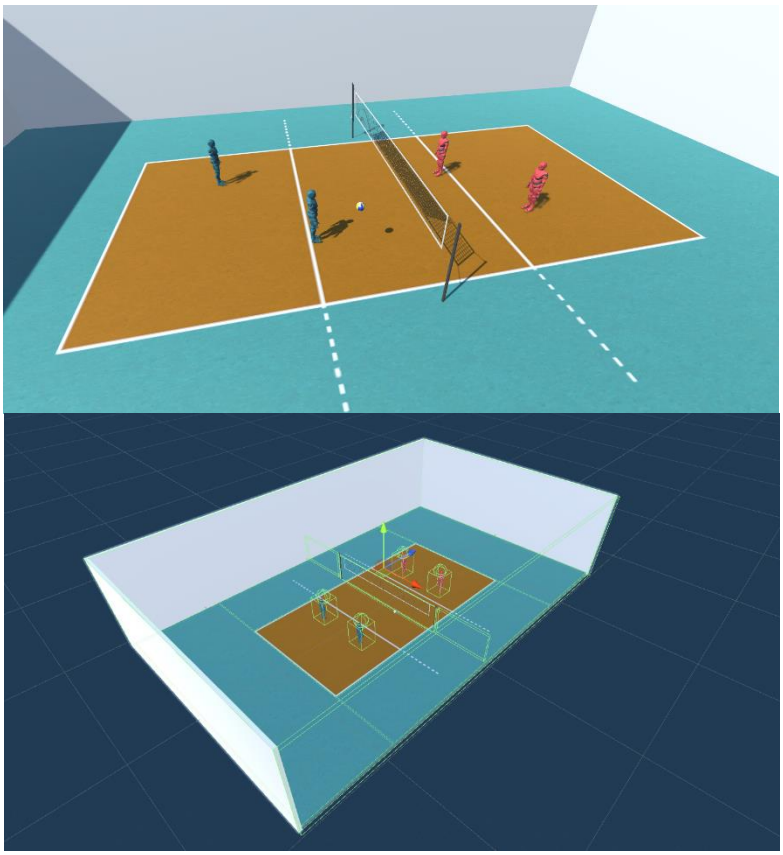
آدرس پکیج‌های ml-agents-envs و ml-agents نیز پوشه‌ای است که ریپازیتوری در آن کلون شده است.

ML-agents برای نمایش آمار و ارقام و نمودارهای مربوط به آموزش[1] عامل‌ها از Tensorboard استفاده می‌کند و این ابزار در طی مراحل قبلی به‌صورت خودکار اما نه به طور کامل نصب می‌شود. برای دسترسی کامل به‌تمامی نمودارها باید کتابخانه NumPy نصب شده را به نسخه جدیدتری به‌روزرسانی کرد. اما نسخه‌های به‌روزتر NumPy با پکیج‌های پایتون ML-agents سازگاری ندارند و این کار اصلاً توصیه نمی‌شود. بهترین روش ایجاد یک محیط مجازی دیگر و نصب TensorFlow و Tensorboard در آن است تا بتوان برای مشاهده نمودارها از این محیط جدید استفاده کرد.

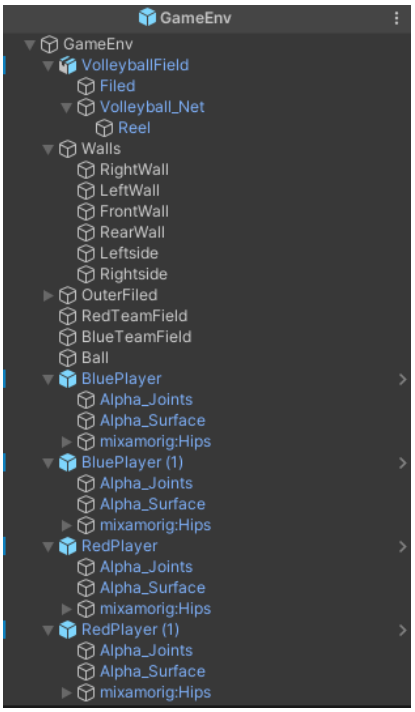
[1] Training

## ۴-۳- طراحی محیط یادگیری

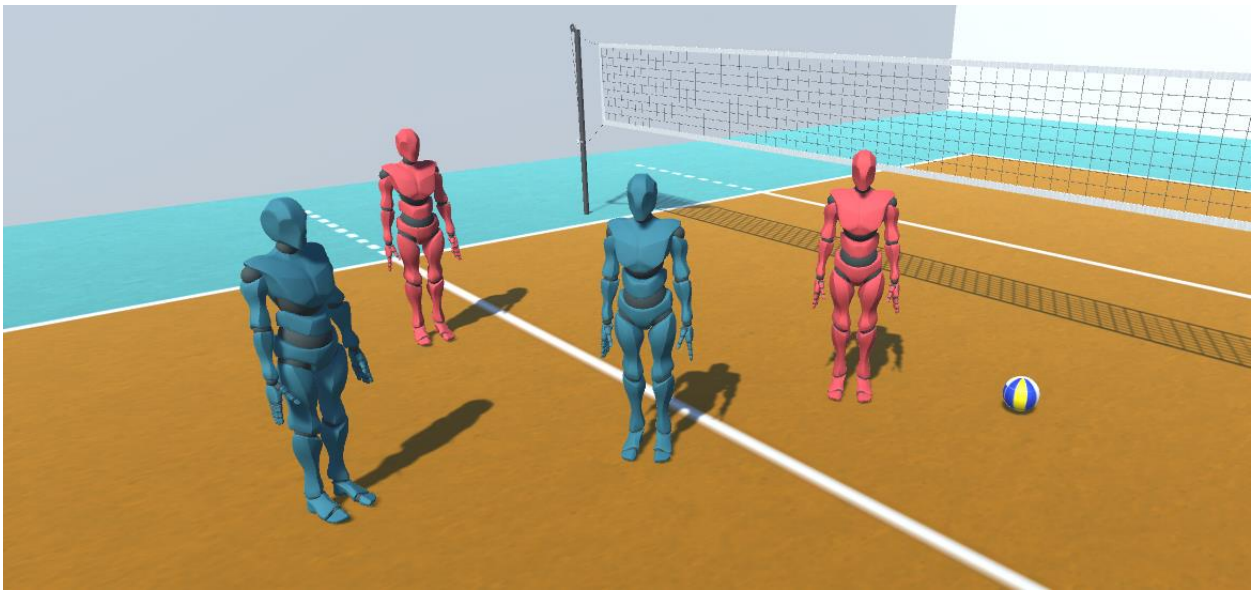
پس از نصب و راه‌اندازی ابزارهای موردنیاز، اولین گام در پیاده‌سازی پروژه طراحی محیطی است که عوامل در آن تعامل خواهند کرد. به این منظور از مدل‌های سه‌بعدی مختلفی برای اجزای مختلف زمین والیبال، بازیکنان و... استفاده شده است. این مدل‌های سه‌بعدی مدل‌ها خام بوده و برای شبیه‌سازی بازی والیبال مولفه‌های مختلفی مانند collliderها به‌منظور ایجاد فیزیک و تشخیص برخورد و همچنین rigidBody به‌منظور اعمال قوانین فیزیکی به آن‌ها اضافه و مناسب با کاربرد هر کدام طراحی شدند. در شکل ۴-۱ نمای ظاهری طراحی نهایی محیطی که عوامل در آن تعلیم داده شده‌اند. نمایش داده شده است. همچنین در شکل ۴-۲ اجزای مختلف این محیط مشخص است. شکل ۴-۳ نیز تصویری از عوامل (بازیکنان) بازی را نشان می‌دهد.



شکل ۴-۱: محیط یادگیری




شکل ۴-۲: اجزای محیط یادگیری



شکل ۴-۳: عامل‌های بازی

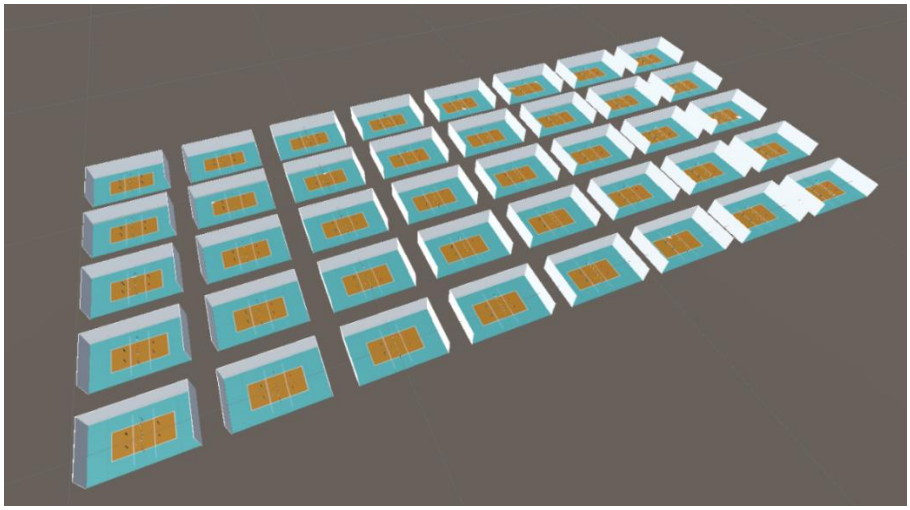
## ۴-۳-۱- افزایش سرعت یادگیری و یادگیری موازی[1]

از آنجایی که در یادگیری تقویتی داده‌های موردنیاز عامل برای یادگیری به‌صورت لحظه‌ای و از تعاملات عامل با محیط تولید می‌شوند، سرعت تولید و ورود این داده‌ها به الگوریتم مورداستفاده، رابطه مستقیم با سرعت یادگیری عامل دارد. به‌منظور این امر از تکنیک‌های مختلفی می‌توان استفاده کرد که یکی از آن‌ها یادگیری موازی می‌باشد.

یادگیری موازی در یادگیری تقویتی به یک رویکرد آموزشی اشاره دارد که در آن چندین نمونه یا کپی از یک عامل به طور هم‌زمان و به طور موازی یاد می‌گیرند. در روش یادگیری معمولی، یک عامل با انجام اقدامات در محیط، دریافت بازخورد به شکل پاداش، و تنظیم خط‌مشی خود بر اساس این تجربیات، یاد می‌گیرد. این فرایند اغلب به‌صورت سری اتفاق می‌افتد. در مقابل، یادگیری موازی شامل اجرای چندین نمونه از عامل به‌صورت موازی است که هر کدام به‌صورت مستقل با محیط تعامل دارند و تجربیات را به طور مستقل جمع‌آوری می‌کنند. سپس از این تجربیات برای به‌روزرسانی خط‌مشی مشترک استفاده می‌شود. یادگیری موازی می‌تواند به طور قابل‌توجهی روند آموزش را تسریع بخشد؛ زیرا به عامل اجازه می‌دهد تا داده‌های بیشتری را جمع‌آوری کند و از طیف وسیع‌تری از تجربیات در همان مدت‌زمان بیاموزد.

[1] Parallel Learning

در ایـن پـروژه نیـز از یـادگیری مـوازی اسـتفاده شـده اسـت و همـان‌طور کـه در شـکل ۴-۴ نمایش‌داده‌شـده، ۴۰ نمونه از محـیط والیبـال طراحـی شـده، به‌صـورت مسـتقل در محـیط Unity قـرار داده شـده اسـت. عامل‌هـای تمـامی ایـن محیط‌هـا هنگـام یـادگیری بـه یـک شـبکه عصـبی متصـل هسـتند. ایـن امـر بسـته بـه ویژگی‌هـای پردازنـده، سرعت یادگیری را چند برابر می‌کند که در این پروژه این افزایش سرعت حدود ۱۰ برابر بود.



شکل ۴-۴: استفاده از یادگیری موازی

برای افـزایش سـرعت یـادگیری عـلاوه بـر یـادگیری مـوازی، بـه علـت اسـتفاده از محـیط شبیه‌سـاز، می‌تـوان مقیـاس زمـانی را افـزایش داد و در ایـن پـروژه مقیـاس زمـانی ۲۰ برابـر دنیـای واقعـی منظـور شـد کـه ایـن امـر موجـب ۲۰ برابر شدن سرعت تولید نمونه[1] نسبت به عامل‌های مشابه در دنیای واقعی می‌شود.

بـه‌عبارت‌دیگر بـا اسـتفاده از یـادگیری مـوازی و افـزایش مقیـاس زمـانی، فراینـد یـادگیری ایـن عامل‌هـا ده‌هـا برابـر سریع‌تر نسبت به عامل‌هـای مشـابه در دنیـای واقعـی سـپری می‌شـود کـه ایـن امـر یکـی از مهم‌تـرین مزایـایی اسـت کـه ML-Agents ارائـه می‌کنـد. یعنـی بـه‌جای تعلـیم یـک عامـل واقعـی، می‌تـوان شبیه‌سـاز آن را بـا سـرعت بسـیار بیشتر تعلیم داد و از خط‌مشی به‌دست‌آمده برای عامل واقعی استفاده کرد.

## ۴-۴- طراحی سیستم بازی

پس از طراحـی محـیط، نوبـت بـه طراحـی بـازی و پیاده‌سـازی منطـق آن می‌رسـد. در طراحـی یـک عامـل یـادگیری تقـویتی، یکـی از مهم‌تـرین وظـایف، طراحـی مشـاهدات[2]، کنش‌هـای ممکـن بـرای عامـل و سـیگنال یـا تـابع پـاداش

[1] Sample
[2] Observations

می‌باشد. علاوه بر این، منطق بازی و قوانین حاکم بر آن نیز باید پیاده‌سازی شوند. در این پروژه سه اسکریپت با نام‌های Player، GameController و BallController برای شبیه‌سازی این محیط بازی پیاده‌سازی شده‌اند که شامل کدهای مربوط منطق بازی و همچنین توابع یادگیری تقویتی ML-Agents می‌باشند. نسخه‌های مختلف این کدها در ریپازیتوری پروژه که آدرس آن در پیوست ۱ قرار دارد، برای ورژن‌های مختلف طراحی شده از این شبیه‌سازی، قرار دارند. بنابراین در ادامه تنها توضیحات مختصری در مورد طراحی و پیاده‌سازی این موارد داده می‌شود.

### ۴-۴-۱- طراحی مشاهدات

مشاهدات عامل در هر گام[1] در واقع نمایانگر حالت محیط در آن گام می‌باشد. منظور از گام در یادگیری تقویتی، یک‌چرخه تعامل عامل با محیط می‌باشد. این چرخه شامل دریافت مشاهدات و پاداش از محیط و انتخاب کنشی مناسب با آن‌ها می‌شود. افزایش تعداد مشاهدات موجب افزایش تعداد ورودی‌های مدل یادگیری تقویتی مورداستفاده می‌شود. این امر، عمل کشف روابط بین اجزای مختلف تشکیل‌دهنده یک حالت از محیط و کنش مناسب با هر حالت را پیچیده‌تر و زمان‌برتر می‌کند. از طرف دیگر، مجموعه مشاهدات عامل باید تمام آنچه که وی برای یک تصمیم‌گیری آگاهانه و درست به آن نیاز دارد را در اختیار او قرار دهد؛ بنابراین برای دریافت بهترین خروجی از مدل تعلیم داده شده، یک مرز حیاتی بین میزان مشاهدات طراحی شده وجود دارد.

ML-Agents امکان استفاده از روش‌های مختلفی برای ارسال مشاهدات به مدل را فراهم می‌کند. در این پروژه از Vector Observation استفاده شده که به ما اجازه می‌دهد انواع مختلف تایپ داده‌ها را در یک آرایه قرار داده و این آرایه را به‌عنوان مجموعه مشاهدات ایجنت به مدل بفرستیم. در ادامه با مشاهدات طراحی شده در برای نهایی این بازی بیشتر آشنا می‌شویم.

### ۴-۴-۲- طراحی کنش‌ها

طراحی کنش‌ها نیز امری مهم و حیاتی در سرعت یادگیری و دقت مدل نهایی به‌دست‌آمده می‌باشد. کنش‌ها باید به صورتی طراحی شوند که امکان دستیابی عامل به هدف طراحی شده برای او را برایش فراهم کنند و از طرفی دیگر میزان آن‌ها به نحوی نباشد که موجب سردرگمی عامل شود.

[1] Step

در ML-Agents از دو نوع کنش می‌توان استفاده کرد. کنش‌های پیوسته[1] و گسسته[2]. برای هر کنش پیوسته، مدل یک عدد بین ۰ و ۱ باز می‌گرداند درحالی‌که برای هر کنش گسسته باید تعداد حالات ممکن را انتخاب کرد. به طور مثال اگر تعداد حالات ۱۰ انتخاب شود، مدل در هر گام عددی طبیعی از میان اعداد ۰ تا ۹ را بر می‌گرداند. این اعداد دریافتی از مدل باید توسط کدهای C# به پیاده‌سازی منطقی فعالیت‌های عامل در محیط تبدیل شوند. در ادامه با کنش‌های طراحی شده برای نسخه نهایی این بازی بیشتر آشنا می‌شویم.

### ۴-۴-۳- طراحی سیگنال پاداش

طراحی نحوه پاداش‌دهی به عامل‌ها نیز تأثیر بسزایی در سرعت و نحوه یادگیری آن‌ها دارد. ازآنجایی‌که سیگنال پاداش به‌عنوان معلم عامل عمل می‌کند، طراحی درست آن بسیار حیاتی است. به طور مثال، در یکی از ابتدایی‌ترین استیج‌های طراحی این بازی که در آن تنها یک عامل حضور داشت و باید یاد می‌گرفت تا توپ را به‌درستی از تور عبور دهد به‌نحوی‌که در زمین حریف قرار گیرد، عامل علاوه بر دریافت پاداش برای زمانی که توپ را به‌درستی به زمین حریف وارد می‌کرد، پاداش جزئی‌ای هم برای ضربه‌زدن به توپ کسب می‌کرد. هدف از این پاداش ترغیب عامل به قرار گرفتن در مسیر توپ بود، فارغ از اینکه دقت ضربهٔ او چه‌قدر خواهد بود. اما از آنجایی در این نسخه از شبیه‌سازی محدودیتی برای تعداد ضربه‌های عامل به توپ در نظر گرفته نشده بود، عامل از یک زمانی به بعد که یاد گرفته بود خود را به‌درستی به توپ برساند، شروع به ضربه‌زدن‌های متوالی به توپ و نگه‌داشتن آن در بالای سرش کرد که در مغایرت باهدف اصلی او یعنی فرستادن توپ به زمین حریف بود. این مثال نشان می‌دهد که تعریف دقیق سیگنال پاداش چه‌قدر ضروری است.

همان‌طور که پیش‌تر نیز بیان شد، برای درک دقیق‌تر نحوه طراحی مشاهدات، کنش‌ها و سیگنال پاداش باید به کدهای موجود در ریپازیتوری این پروژه رجوع شود.

[1] Continuous
[2] Discrete

## ۴-۵- روند طراحی بازی

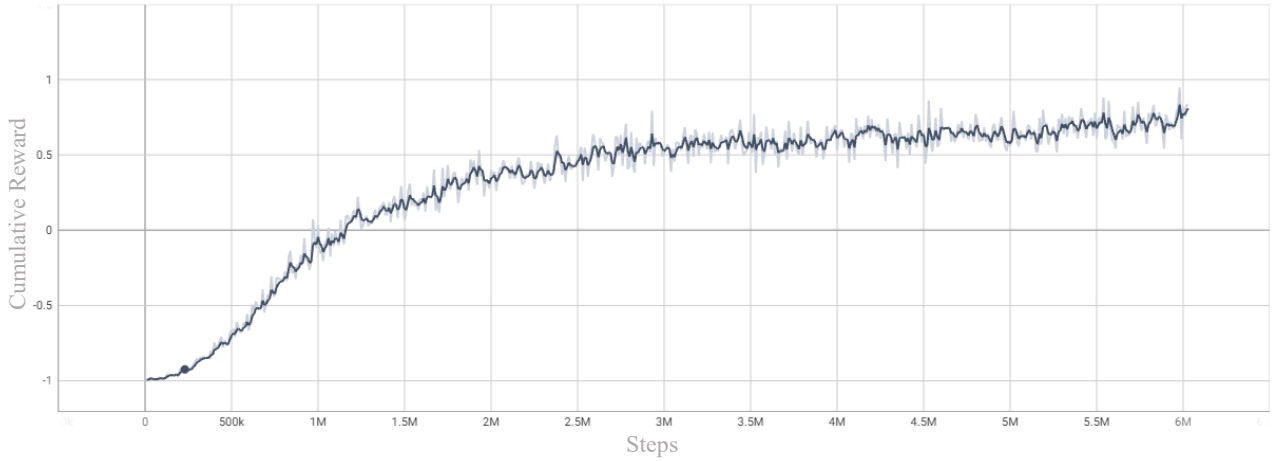
در مسیر پیاده‌سازی این شبیه‌ساز والیبال، طراحی بازی از حالت‌های ساده‌تر آغاز و به‌مرور تکامل یافت. در هر چرخه از تکامل بازی، یک مدل برای آن نسخه از بازی تعلیم داده شد که البته تعلیم این مدل‌ها پیش از رسیدن به یک نقطه نسبتاً مطلوب متوقف شد چرا که هدف اصلی پروژه نبوده‌اند و از آن‌ها تنها برای بررسی درستی منطق پیاده‌سازی تا آن مرحله از شبیه‌سازی بازی استفاده می‌شد. این امر به پروژه کمک کرد تا در صورت وجود نقص در پیاده‌سازی قوانین بازی، میزان مشاهدات عامل، کنش‌ها و سیگنال پاداش، این عیب‌ها سریع‌تر و در محیط‌های کنترل‌شده‌تری پیدا و برطرف شوند. چرا که در مراحل ساده‌تر بازی، مدت‌زمان موردنیاز برای تعلیم مدل چندین برابر کم‌تر از نسخه نهایی آن بود.

در ادامه ۳ ایستگاه اصلی توسعه این محیط شبیه‌ساز بازی والیبال به‌صورت مختصر شرح داده شده‌اند. کدها و مدل‌های مربوط به هر مرحله در ریپازیتوری پروژه موجود است.

### ۴-۵-۱- بازی تک عاملی

در ساده‌ترین محیط طراحی شده برای این بازی، تنها یک عامل در زمین حضور داشت که هدف آن تشخیص مسیر صحیح توپ و رساندن خودش به آن و سپس ضربه به توپ به‌نحوی‌که در زمین حریف قرار گیرد بود. مشاهدات این عامل شامل بردار موقعیت و بردار سرعت توپ و همچنین بردار دو بعدی موقعیت مکانی خودش بود. کنش‌های ممکن برای آن نیز در ۵ شاخه گسسته تقسیم می‌شدند: ۱-حرکت ۲-دفاع ۳-پاس ۴-اسپک ۵-جهت ورود ضربه به توپ. هر شاخه از کنش‌ها تعداد مختلفی حالت را در بر داشتند. حالت‌های مختلف دو شاخهٔ پاس و اسپک دارای نیروهای مختلفی در دو جهت عمودی و افقی می‌باشند که میزان قوس و شدت ضربه را مشخص می‌کند و در شاخه اسپک، نیروی عمودی در حالات مختلف به مراتب کمتر از حالات مختلف شاخه پاس بود. برای تعلیم عامل در این حالت از الگوریتم PPO استفاده شد. حداکثر پاداشی که عامل در این مرحله در هر قسمت[1] از بازی می‌توانست دریافت کند ۱.۳+ (۰.۳+ برای ضربه به توپ و ۱+ برای فرستادن توپ به زمین حریف) و حداقل آن ۱- (تماس توپ با زمین خود یا فرستادن آن به اوت) بود. نمودار میانگین پاداش تجمیعی این عامل برحسب تعداد گام‌های سپری شده در شکل ۴-۵ نمایش داده شده است.

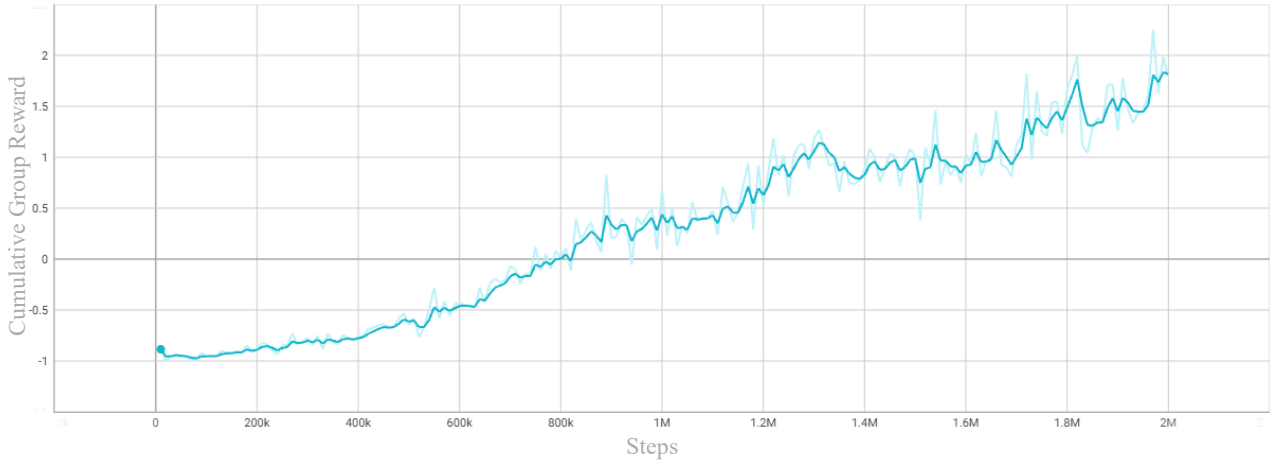
[1] Episode




شکل ۴-۵: نمودار میانگین پاداش تجمیعی بازی تک عاملی

### ۴-۵-۲- بازی مشارکتی

پس از پیاده‌سازی حالت قبلی و تعلیم موفق عامل، این بار یک عامل دیگر نیز به محیط اضافه شد. در این مرحله هر دو عامل در یک تیم قرارداشتن و برخی قوانین بازی مطابق با این موضوع تغییر کرد؛ اما کنش‌ها دست‌نخورده باقی ماند. همچنین موقعیت مکانی عامل هم‌تیمی نیز به مشاهدات هر یک از عوامل اضافه شد. با به‌وجودآمدن تیم، تغییراتی اساسی در کدهای بازی جهت مشخص‌کردن تیم عامل‌ها و محدودیت‌ها و قوانین جدید اعمال شده به‌واسطه تیم‌بندی، به وجود آمد. علاوه بر این، الگوریتم مورداستفاده برای تعلیم عامل‌ها به MA-POCA تغییر یافت. چرا که این الگوریتم اجازهٔ یادگیری تیمی را به عامل‌ها می‌دهد. حداکثر پاداش گروهی که عوامل در این مرحله در هر قسمت از بازی می‌توانست دریافت کند ۲+ (۱+ برای ضربه به توپ و ۱+ برای فرستادن توپ به زمین حریف) و حداقل آن ۱- (تماس توپ با زمین خود یا فرستادن آن به اوت) بود. نمودار میانگین پاداش تجمیعی گروهی به‌دست‌آمده توسط این عوامل برحسب تعداد گام‌های سپری شده در شکل ۴-۶ نمایش‌داده‌شده است. همان‌طور که مشاهده می‌شود در این مرحله تنها پس از حدود ۲ میلیون گام، عوامل به‌دقت بسیار خوبی در پیداکردن مسیر توپ و ضربه به آن به‌نحوی‌که وارد زمین حریف شود رسیده‌اند. در واقع دلیل آن هم استراتژی‌ای است که این دو عامل خیلی زود در پیش گرفتند. آن‌ها شروع به چسبیدن به یکدیگر کردن و این امر موجب افزایش فضایی شد که می‌توانند به توپ ضربه بزنند. در واقع یکی از اصلی‌ترین معضلات عوامل که یافتن مسیر دقیق توپ بود به‌سرعت بر طرف شد.



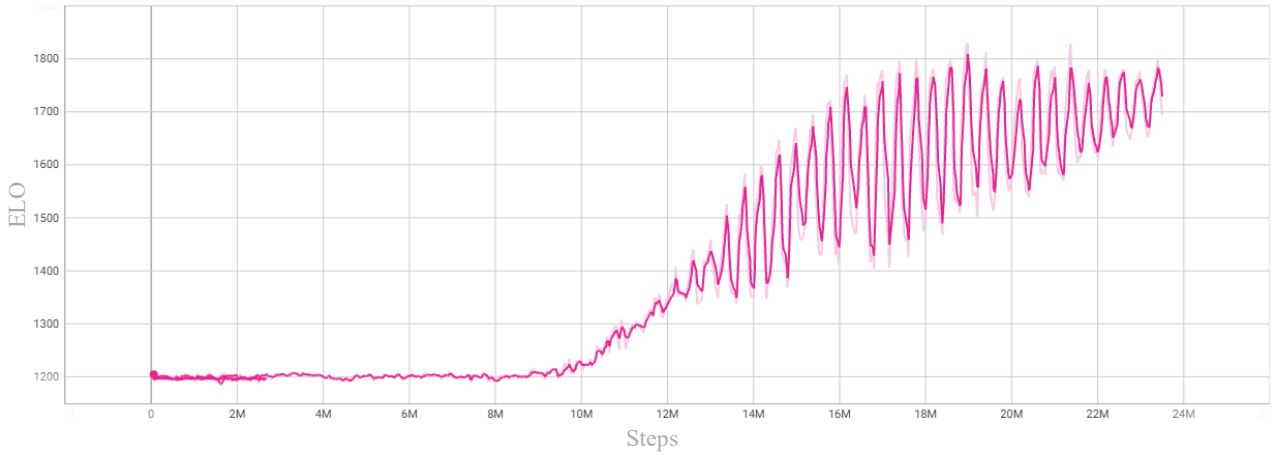

شکل ۴-۶: نمودار میانگین پاداش تجمیعی گروهی بازی مشارکتی

### ۴-۵-۳- بازی مشارکتی-رقابتی

در گام بعدی طراحی شبیه‌ساز بازی والیبال، تیم رقیب نیز به بازی اضافه شد. یعنی در این مرحله که قرار است مرحله نهایی باشد، محیط بازی شامل دو تیم دونفری می‌باشد. قوانین بازی تکمیل گردید و کدهای جدیدی برای اعمال محدودیت‌ها و شرایط بازی رقابتی اضافه شد. با اضافه‌شدن تیم رقیب، ازآنجایی‌که والیبال یک بازی متقارن است و تیم حریف نیز می‌تواند از خط‌مشی یکسان استفاده کند، self-play نیز به تنظیمات پیکربندی الگوریتم تعلیم‌دهنده که در این مرحله نیز MA-POCA می‌باشد، افزوده شد. همچنین در این گام، با بررسی‌های انجام شده در مراحل قبلی، تعدادی از کنش‌های اضافی در عین حفظ همان گستردگی قبلی حذف شدند. به طور مثال در مراحل قبل، عامل در شاخهٔ اعمال مربوط به حرکت، کنش پریدن را داشت. همچنین دارای شاخه‌ای برای دفاع روی تور بود. اما دفاع روی تور با همان پرش موجود در شاخه حرکت نیز قابل‌پیاده‌سازی بود، بنابراین این شاخه حذف شد. شاخه پاس و اسپک نیز با هم ادغام شدند و شاخه جدیدی به نام ضربه را به وجود آوردند که حالات مختلف آن به عامل امکان انجام عمل پاس با قوس و شدت‌های مختلف و عمل اسپک با شدت‌های مختلف را می‌دهد. مشاهدات عوامل نیز افزایش یافت و موقعیت مکانی بازیکنان تیم حریف نیز به آن اضافه شد. علاوه‌برآن برای هر عامل ۲ متغیر دیگر نیز به‌عنوان مشخصات هر حالت از محیط به بردار مشاهداتش اضافه شد که نشان‌دهنده قادر بودن عامل برای ضربه به توپ (یک عامل نمی‌تواند دو بار پشت‌سرهم به توپ ضربه بزند) و تعداد ضربات زده شده توسط تیم هر عامل (هر تیم حداکثر باید با دو ضربه توپ را به زمین حریف بفرستد) بودند.

در اولین تعلیم عامل‌ها در این حالت، بعد از گذشت نزدیک به ۲۴ میلیون گام، عوامل به‌خوبی شروع به ردوبدل کردن توپ بین تیم‌ها کردند و طول میانگین قسمت‌های بازی، به‌شدت افزایش یافت. اما نکته ناامیدکننده عدم وجود همکاری بین هم‌تیمی‌ها بود. در واقع بازیکنان هر تیم به‌خوبی خود را در مسیر توپ

قرار می‌دادند و مانع از زمین خورد آن می‌شدند و توپ را به زمین حریف، گاها با ضرباتی بسیار زیرکانه ارسال می‌کردند. ازآنجایی‌که در شرایط self-play، پاداش منبعی مشترک بین دو تیم است، یعنی برنده همان میزان پاداش مثبت می‌گیرد (+۱) که بازنده پاداش منفی (-۱) می‌گیرد، برای بررسی میزان پیشرفت یادگیری عوامل دیگر نمی‌توان از معیار میانگین پاداش گروهی به‌دست‌آمده استفاده کرد. چرا که مقدار آن در محیط همواره صفر است. در این شرایط، سیستم امتیازدهی ELO به‌خوبی می‌تواند میزان پیشرفت یادگیری عوامل را نشان دهد. نمودار شکل ۴-۷ امتیاز ELO این عوامل را نسبت به گام‌های سپری شده نشان می‌دهد.

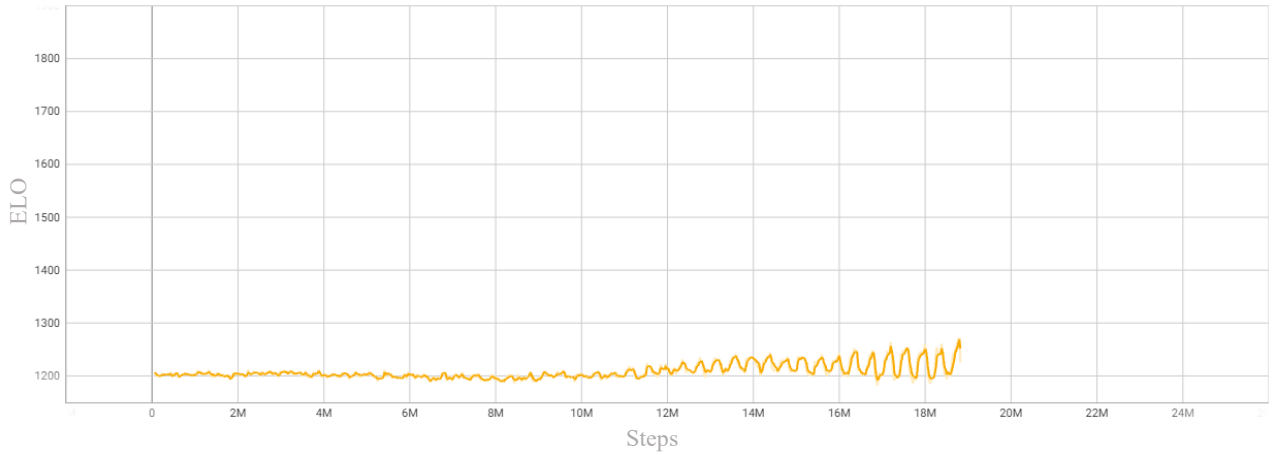



شکل ۴-۷: نمودار امتیاز ELO اولین مدل بازی مشارکتی-رقابتی

یکی از عواملی که موجب بازی‌های طولانی بین این دو تیم شده بود، عدم یادگیری آن‌ها در استفاده از ضربات اسپک بود. در نتیجه عوامل همواره توپ را با قوس‌های بلند و شدت کمتر به زمین حریف می‌فرستادند که این امر دریافت آن در زمین حریف را ساده‌تر می‌کند.

همان‌طور که گفته شد، این بازی فاقد پاس و فعالیت‌های همکارانه محسوس بود؛ بنابراین برای‌آنکه بتوان پاس‌کاری و همکاری‌های محسوس را به بازی اضافه کرد، یکی دیگر از شرایط واقعی بازی والیبال به این شبیه‌ساز اضافه شد و آن هم عدم امکان استفاده از ضربات پر قدرت در دریافت اول می‌باشد. در واقع بازیکن‌ها در دریافت اول تنها می‌توانند پاس‌هایی با قوس‌های بلند و شدت کم ارسال کنند؛ بنابراین یک بازیکن تنها درصورتی‌که می‌تواند در دریافت اول توپ را به زمین حریف ارسال کند که بسیار نزدیک تور باشد. در غیر این صورت، برای عبور توپ از تور، بازیکن اول باید توپ را به هم تیمی خود پاس دهد تا او توپ را به زمین رقیب بفرستد. پس از اعمال قوانین جدید در منطق برنامه، عامل‌ها مجدد تعلیم داده شدند که نتایج آن در شکل ۴-۸ نمایش‌داده‌شده است. همان‌طور که مشاهده می‌شود، به علت پیچیده‌تر شدن شرایط کسب امتیاز در بازی جدید، حتی بعد از گذشت ۱۹ میلیون گام، باز هم تغییر محسوسی در ELO

عامل‌ها به وجود نیامد. در اینجا فرایند تعلیم متوقف گردید تا راهکاری برای افزایش سرعت یادگیری پیدا و پیاده‌سازی شود.



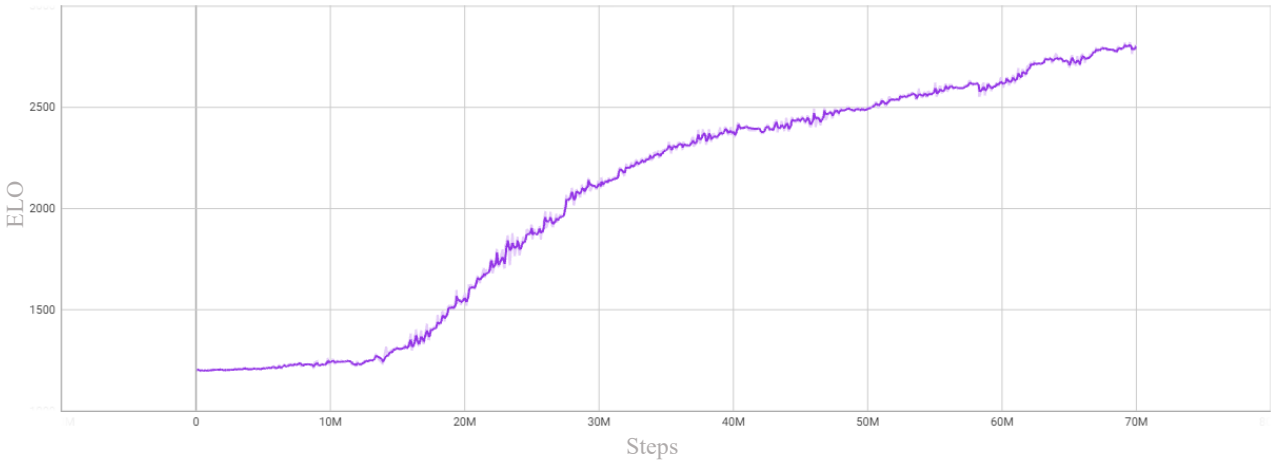

شکل ۴-۸: نمودار امتیاز ELO دومین مدل بازی مشارکتی-رقابتی

دو راهکار زیر برای افزایش سرعت یادگیری عوامل در حالت قبل پیاده‌سازی شد.

- **تغییر حالت اولیه**[1]: حالت اولیه در یادگیری تقویتی، به اولین حالت شروع هر قسمت (Episode) از محیط گفته می‌شود. یک تکنیک مورداستفاده از در RL به‌منظور یادگیری سریع‌تر عامل، استفاده از حالات اولیه‌ای است که عامل را در شرایط خاصی قرار می‌دهد که موجب کشف سریع‌تر خط‌مشی مناسب برای آن شرایط می‌شود. تا پیش‌ازاین، برای شروع هر قسمت از بازی، توپ به‌صورت رندوم در نقطه‌ای از زمین یکی از دو تیم رها می‌شد. اما در طراحی جدید توپ تنها در یک‌سوم انتهایی زمین رها می‌شود. این امر سبب می‌شود تا عامل‌ها نتوانند در اولین دریافت توپ را عبور دهند و خیلی زودتر به فکر همکاری و یادگیری بیافتند.
- **اضافه‌کردن پاداش فردی:** در طراحی جدید، برای ضربه‌زدن به توپ، چه ضربه اول باشد و چه ضربه دوم، به عامل ضربه‌زننده پاداش (۰.۱+) داده می‌شود. علت این کار آن است که تجربه فرایند پاس‌دادن به هم تیمی و سپس ضربه‌زدن به توپ توسط یار دوم و تبدیل‌شدن آن به امتیاز، فرایندی پیچیده است که عوامل برای تجربهٔ آن نیاز به تجربه نمونه‌های (Sample) بسیار زیادی دارند. این امر موجب می‌شود تا عوامل خیلی دیر یاد بگیرند که راه کسب پاداش گروهی، طی نمودن چنین مسیری است. اضافه‌شدن امتیاز فردی برای ضربه به توپ، مانند تقلبی برای یادگیری این مسیر عمل می‌کند.

[1] Initial State

نتیجه حاصل از تعلیم عوامل پس اعمال تغییرات ذکر شده، در شکل ۴-۹ نمایش‌داده‌شده است. این نسخه از بازی نسخه نهایی آن می‌باشد که ۷۰ میلیون گام و نزدیک به ۱۵ ساعت تعلیم داده شده است.




شکل ۴-۹: نمودار امتیاز ELO مدل نهایی بازی مشارکتی-رقابتی (گام-ELO)

در این نسخه از بازی، به‌خوبی شاهد همکاری بین عوامل هستیم. همچنین بر خلاف نسخه اولیه از حالت مشارکتی-رقابتی بازی که در آن طول هر دست از بازی بسیار طولانی بود، طول بازی‌های این نسخه کوتاه‌تر است که یکی از مهم‌ترین دلایل آن، استفاده عوامل از اسپک در ضربات دوم یک تیم است. این ضربات با احتمال بسیار در زمین حریف دریافتی را به همراه ندارند و به همین دلیل دست‌های بازی سریع‌تر به اتمام می‌رسند. نکته جالب‌توجه دیگر انجام عمل دفاع روی تور است، این کنش نه به‌کرات، اما هر چند دست یک‌بار اتفاق می‌افتد که عملی بسیار زیرکانه است.

در فصل بعدی یعنی فصل نتایج، آمار و ارقام بیشتری از نسخه نهایی بازی ارائه خواهد شد. همچنین این نسخه در ۳ پیکربندی متفاوت شبکه (تعداد لایه‌های مختلف) تعلیم داده شد تا علاوه بر انتخاب بهترین مدل تولید شده، تأثیر تعداد لایه‌های شبکه بر فرایند یادگیری نیز مورد ارزیابی قرار گیرد. نتایج این مقایسه نیز در فصل آینده آورده شده است.

## ۴-۶- پیکربندی نسخه نهایی بازی

در این بخش تنظیمات پیکربندی استفاده شده برای تعلیم ایجنت‌ها در نسخه نهایی بازی را به طور مختصر شرح داده و برخی پارامترهای آن و علت مقادیر انتخابی برای هر یک را بررسی می‌کنیم.

```
behaviors:
  TwoTeamsTwoPlayers-v2.2:
    trainer_type: poca
    hyperparameters:
      batch_size: 2048
      buffer_size: 20480
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: constant
    network_settings:
      normalize: true
      hidden_units: 512
      num_layers: 3
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    checkpoint_interval: 10000000
    keep_checkpoints: 8
    max_steps: 80000000
    time_horizon: 1000
    summary_freq: 10000
    self_play:
      save_steps: 50000
      team_change: 200000
      swap_steps: 2000
      window: 10
      play_against_latest_model_ratio: 0.5
      initial_elo: 1200.0
```

trainer_type در واقع مشخص‌کننده الگوریتم استفاده شده برای تعلیم می‌باشد که در این نسخه poca یا همان MA-POCA می‌باشد. پارامتر buffer_size نشان‌دهنده میزان تجربیاتی (Samples) است که باید قبل از انجام هر گونه یادگیری یا به‌روزرسانی مدل (خط‌مشی) جمع‌آوری شود و پارامتر batch_size تعیین می‌کند که چه میزان از تجربیات جمع‌آوری شده در بافر، در هر چرخه آموزش استفاده می‌شود؛ بنابراین مقدار buffer_size باید چندبرابر مقدار batch_size در نظر گرفته شود که در اینجا ۱۰ برابر است. درست است که مقادیر در نظر گرفته شده برای این دو پارامتر در این پروژه نسبتاً بزرگ بوده و موجب افزایش نیاز

به حافظه و منابع پردازشی می‌شود، اما مقدار زیاد این پارامترها سبب می‌شود تا هر به‌روزرسانی بر اساس نمونه‌ای بزرگ‌تر و پایدارتر از تجربیات باشد و فرایند همگرایی[1] روان‌تر باشد.

پارامتر learning_rate نشان‌دهنده نرخ به‌روزرسانی پارامترها یا وزن‌های شبکه عصبی می‌باشد. مقدار انتخاب شده برای این پارامتر، در بازهٔ معمول آن و به‌منظور ایجاد تعادل بین سرعت و پایداری یادگیری می‌باشد. همچنین پارامتر learning_rate_schedule بیانگر نحوه تغییر learning_rate در طول فرایند یادگیری می‌باشد که در این پروژه، ثابت و بدون تغییر تنظیم شده است. پارامتر beta در واقع ضریبی است برای تعیین میزان کنش‌های عامل که به‌منظور اکتشاف صورت می‌گیرند. میزان انتخاب شده برای این پارامتر، به‌منظور حفظ تعادل میان اکتشاف و بهره‌برداری می‌باشد. پارامتر epsilon نیز ضریبی است برای تعیین سرعت تکامل خط‌مشی. افزایش این پارامتر سرعت یادگیری را افزایش داده؛ اما آن را غیر پایدار می‌کند.

در قسمت تنظیمات شبکه، به علت اینکه بازی ما یک مسئله کنترل پیوسته و با ورودی‌های پیوسته می‌باشد، نرمال سازی فعال شده است. با توجه به تعداد مشاهدات و کنش‌ها، تعداد hidden_units برابر ۵۱۲ و با توجه به میزان پیچیدگی بازی، تعداد num_layers در ابتدا برابر ۳ تنظیم شد. سپس نسخه نهایی بازی با تعداد لایه‌های ۲ و ۴ نیز ترین شد تا تاثیر تعداد لایه‌ها سنجیده شود.

پارامتر gamma نشان‌دهنده آن است که عامل در تخمین پاداش تجمیعی در هر episode، تا چه میزان باید پاداش ناشی از تجربیات آتی خود را در نظر بگیرد که در این بازی به علت ماهیت آن و دریافت پاداش در انتهای episode، این مقدار زیاد است. پارامتر strength ضریبی است که الگوریتم به پاداش‌های دریافت شده از محیط می‌دهد که در اینجا ۱ می‌باشد.

max_steps نشان‌دهنده تعداد گام‌هایی است که پس از آن تعلیم مدل متوقف می‌شود که در اینجا ۸۰ میلیون گام تنظیم شده است. checkpoint_interval نیز بیانگر این است که بعد از چند گام یک نسخه از مدل تا آن لحظه فرایند تعلیم ذخیره‌سازی می‌شود که در این تنظیمات ۱۰ میلیون گام در نظر گرفته شده است. keep_checkpoints نیز تعداد این مدل‌های ذخیره شده در طی فرایند آموزش را نشان می‌دهد. درصورتی‌که اعداد این مدل‌ها از این مقدار بیشتر شود، قدیمی‌ترین مدل حذف و مدل جدید نگه‌داری می‌شود.

در انتها نیز تنظیمات مربوط به حالت self-play قرار دارد. team_change نشان‌دهنده تعداد گام‌هایی است که پس از آن جای تیمی که در حال یادگیری است با تیمی که در حال استفاده از خط‌مشی ثابت می‌باشد عوض می‌شود. save_steps نیز نشان‌دهنده آن است که هر چند گام یک‌بار یک تصویر[2] جدید از خط‌مشی

[1] Convergence
[2] Snapshot

بـه‌روز عامـل گرفتـه می‌شـود. از ایـن تصـاویر هنگـام اسـتفاده از خط‌مشـی ثابـت اسـتفاده می‌شـود. window نشـان‌دهنده تعـداد آخـرین تصـاویر ذخیـره شـده می‌باشـد و swap_steps نشـان می‌دهـد کـه تیمـی کـه در حـال اسـتفاده از خط‌مشـی ثابـت اسـت، هـر چنـد گـام یک‌بـار از یـک تصـویر جدیـد اسـتفاده می‌کنـد. اسـتفاده از تصـاویر متفاوت از خط‌مشـی‌های گذشـته، موجـب می‌شـود تـا یـادگیری تیمـی کـه در حـال بهبـود خط‌مشـی می‌باشـد، بهتـر و نسبت به سیاست‌های مختلف اتفاق بیفتد، نه‌تنها یک سیاست و استراتژی خاص.

### ۴-۷- استفاده از PPO به‌جای MA-POCA

همـان‌طور کـه پیش‌تـر و در فصـل مفـاهیم بیـان شـد، الگـوریتم MA-POCA توسـط تـیم توسـعه‌دهنده ML-Agents و بـه‌منظور اسـتفاده در یـادگیری تیمـی توسـعه‌یافته اسـت. بـه‌منظور بررسـی تـأثیر واقعـی ایـن الگـوریتم در یادگیری تعـاملات مشـارکتی، نسـخه نهـایی بـازی بـا همـان پیکربنـدی بیـان شـده در قسـمت قبـل، امـا ایـن بـار بـا الگوریتم PPO تعلیم‌یافتـه اسـت تـا بتـوان نتـایج آن را بـا نتـایج حاصـل از الگـوریتم MA-POCA مقایسـه کـرد. ایـن نتایج در فصل بعد بیان شده‌اند.

## ۴-۸- جمع‌بندی

در ایـن فصـل فعالیت‌هـای مختلـف صـورت‌گرفته در مسـیر پیاده‌سـازی ایـن پـروژه، از طراحـی محـیط یـادگیری گرفته تا طراحـی و پیاده‌سـازی منطـق بـازی و اجـزای مختلـف سیسـتم ماننـد کنش‌هـا و مشـاهدات شـرح داده شـد. مراحـل مختلـف طراحـی بـازی و چالش‌هـای هـر یـک بیـان شـد و نمودارهـای مختصـری بـرای درک سـیر یـادگیری عوامـل در نسـخه‌های مختلـف بـازی ارائـه شـد. در فصـل آتـی نتـایج حاصـل از ایـن پـروژه و فراینـدهای تعلـیم، دقیق‌تر بررسی خواهند شد.

# فصل پنجم
# نتایج

## ۵-۱- مقدمه

در فصل گذشته توضیحات نحوه پیاده‌سازی و انجام پروژه و همچنین فعالیت‌های صورت‌گرفته در خلال آن، به طور کامل ارائه شد. در این فصل به بررسی نتایج دو نمونه از ارزیابی‌های صورت‌گرفته در این پروژه پرداخته خواهد شد. مورد اول بررسی تأثیر تعداد لایه‌های شبکه در یادگیری مدل و مورد دوم بررسی تأثیر الگوریتم MA-POCA در یادگیری تیمی و مقایسه آن با الگوریتم PPO می‌باشد.

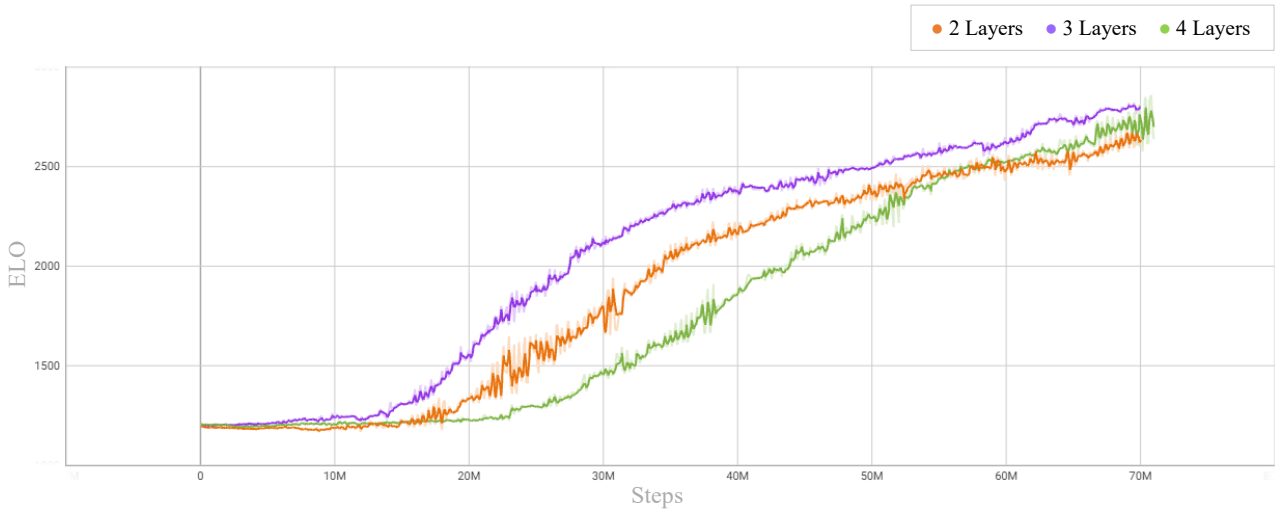
## ۵-۲- ارزیابی تأثیر تعداد لایه‌های شبکه

همان‌طور که در فصل پیش بیان شد، نسخه نهایی این شبیه‌ساز، با ۳ حالت مختلف شبکه تعلیم داده شد که در ادامه تفاوت‌های این ۳ حالت ارزیابی خواهند شد. این ۳ حالت در واقع دارای تعداد لایه‌های متفاوت می‌باشند و تعداد لایه‌های در نظر گرفته شده ۲، ۳ و ۴ است. هر یک از این ۳ حالت، ۷۰ میلیون گام آموزش داده شده‌اند.

### ۵-۲-۱- سرعت یادگیری

نمودار امتیاز ELO برای این ۳ حالت شبکه در شکل ۵-۱ نمایش‌داده‌شده است. همان‌طور که در شکل مشخص است، مدل دارای ۳ لایه در تمام زمان تعلیم، امتیاز بیش‌تری از دو مدل دیگر داشته است و در

نهایت و در انتهای این ۷۰ میلیون گام نیز، با امتیاز بیشتری فرایند تعلیم را خاتمه داده است. امتیاز بیشتر نشان‌دهنده کسب توانمندی‌های بیشتر توسط عوامل است. علاوه بر این، مدل ۳ لایه‌ای خیلی سریع‌تر از دو مدل دیگر شروع به یادگیری‌های محسوس و افزایش امتیاز کرده است. اتفاقی که بعد از حدود ۷ میلیون گام برای این مدل افتاده است، در مدل‌های ۲ و ۴ لایه‌ای به ترتیب بعد از حدود ۱۷ و ۲۳ میلیون رخ‌داده است.




شکل ۵-۱: نمودار مقایسه امتیاز ELO در سه مدل نسخه نهایی

شیب یادگیری نیز در این مدل از دو مدل دیگر بیشتر بوده است. نکته جالب‌توجه دیگر در این نمودار، پیشی‌گرفتن امتیاز مدل ۴ لایه از مدل ۲ لایه پس ۶۰ میلیون گام می‌باشد. به‌عبارت‌دیگر شیب فرایند یادگیری در مدل ۲ لایه به علت ساده‌تر بودن آن نسبت به مدل ۴ لایه، پس از مدتی افول پیدا کرده است. این امر برای مدل ۲ لایه نیز صادق است و چه‌بسا اگر فرایند تعلیم چند ده میلیون گام دیگر ادامه پیدا می‌کرد، نتیجه نهایی مدل ۴ لایه بهتر می‌بود. مدل ۴ لایه به علت پیچیده‌تر بودن، به زمان بیشتری برای تعلیم نیاز دارد و همچنین شیب یادگیری آن نیز کمتر ولی باثبات‌تر است.

### ۵-۲-۲- میانگین طول episode

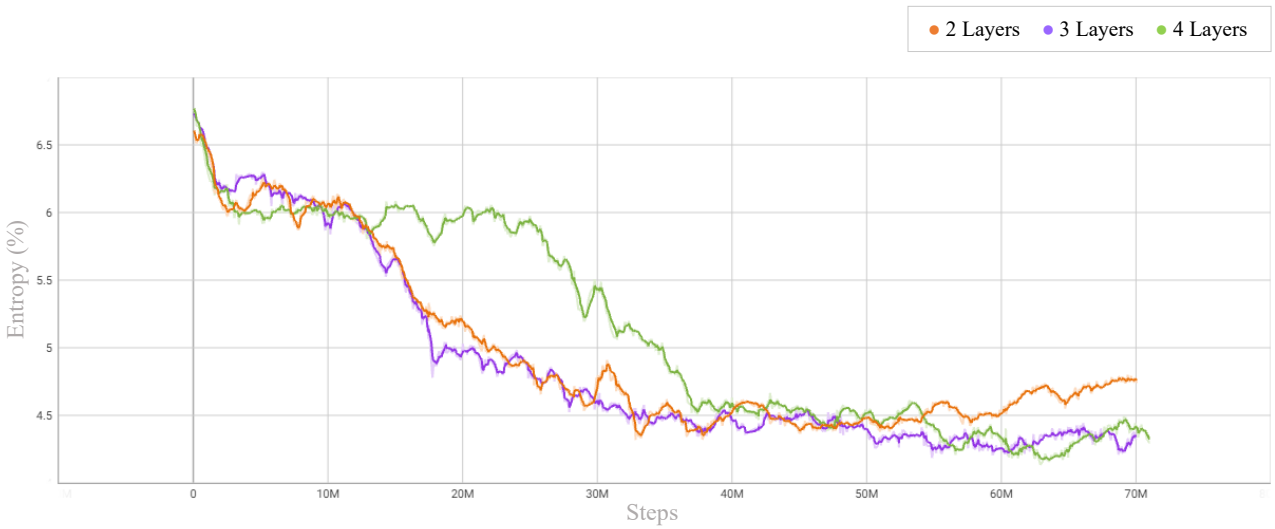
در این قسمت به سراغ مقایسه میانگین طول هر دست از بازی در این ۳ حالت از شبکه می‌رویم. مطابق نمودار شکل ۵-۲، در این معیار نیز مدل ۳ لایه‌ای در تمامی طول تعلیم، دارای بازی‌های طولانی‌تری نسبت به ۲ مدل دیگر داشته است. البته در انتهای فرایند آموزش نتایج این مدل و مدل ۲ لایه‌ای بسیار نزدیک به هم بوده است و میانگین طول هر دست بازی (در این شبیه‌سازی هر امتیاز یک دست محسوب می‌شود که همان episode تعریف شده برای این بازی نیز می‌باشد) تقریباً برابر با ۴ ثانیه شده است. این مقدار دو برابر

طول episodeهای مدل ۴ لایه‌ای می‌باشد. در واقع در جریان بازی قرارگرفتن و ردوبدل توپ بین دو تیم در مدل ۴ لایه‌ای به‌مراتب کمتر اتفاق می‌افتد. البته این امر نمی‌تواند لزوماً به معنی بهتر بودن خط‌مشی به‌دست‌آمده توسط مدل‌های ۲ و ۳ لایه‌ای باشد. به طور مثال ممکن است عامل دارای مدل ۴ لایه‌ای، کنش‌هایی را یادگرفته باشد که سریع‌تر او را به امتیاز برساند و مانع از دریافت توپ توسط تیم رقیب شده باشد. به همین دلیل برای مقایسهٔ این ۳ مدل تنها به آمار و ارقام اتکا نخواهیم کرد و توانمندی این مدل‌ها را در بازی رو در رو نیز خواهیم سنجید.


شکل ۵-۲: نمودار مقایسه میانگین طول episode در سه مدل نسخه نهایی

### Entropy -۳-۲-۵

نمودار شکل ۵-۳ نمایانگر این است که تصمیم‌های گرفته شده توسط مدل و یا در واقع کنش‌های آن، چه میزان تصادفی‌اند. با گذشت زمان میزان تصادفی بودن کنش‌های عوامل باید کاهش یابد چرا که مدل به خط‌مشی بهینه نزدیک‌تر شده است و کمتر نیاز به کاوش در محیط دارد و می‌تواند بیشتر به بهره‌برداری از دانش کسب‌شده‌اش بپردازد.




شکل ۵-۳: نمودار مقایسه policy entropy در سه مدل نسخه نهایی

همان‌طور که در نمودار قابل‌مشاهده است، مدل ۴ لایه‌ای که فرایند یادگیری محسوس را پس از حدوداً ۲۳ میلیون گام آغاز کرده بود، پس از همین میزان گام هم شروع به کاهش انجام کنش‌های تصادفی کرده است.

### ۵-۲-۴- پاداش تجمیعی

همان‌طور که در فصل ۴ مطرح شد، به‌منظور کوتاه‌کردن مسیر یادگیری و اینکه عوامل سریع‌تر متوجه نیاز به قرارگیری در مسیر توپ شوند، از یک پاداش جزئی و انفرادی برای ضربه‌زدن به توپ استفاده شد. نمودار شکل ۵-۴ نمایانگر میانگین پاداش تجمیعی دریافتی عوامل یک تیم در طی مسیر یادگیری می‌باشد.


شکل ۵-۴: نمودار مقایسه میانگین پاداش تجمیعی عوامل هم‌تیمی در سه مدل نسخه نهایی

همان‌طور که مشاهده می‌شود، این نمودار بسیار مشابه نمودار طول episode می‌باشد چرا که میزان پاداش به‌دست‌آمده توسط عامل‌های یک تیم نشان‌دهنده تعداد ضرباتی است که به توپ زده‌اند و تعداد ضربات وارد شده به توپ، رابطه مستقیم با مدت‌زمان هر دست از بازی دارد.

استنتاجی که از این نمودار می‌توان کرد آن است که در مدل‌های ۲ و ۳ عاملی، در انتهای فرایند تعلیم، در هر دست از بازی عوامل هر تیم به طور میانگین تنها کمی بیش‌تر از یک ضربه به توپ وارد کرده‌اند (پاداش هر ضربه به توپ ۰.۱ می‌باشد) و این مقدار در مدل ۴ لایه‌ای نصف می‌شود. علت پایین‌بودن این میانگین این است که در خیلی از دست‌ها، تیم‌ها موفق به دریافت توپ اول که به‌صورت رندوم در نقطه‌ای از زمینشان رها شده است نمی‌شوند و این امر موجب ۰ شدن میانگین پاداش تیمی می‌شود.

### ۵-۲-۵- رقابت مدل‌ها

تا به اینجا آمار و ارقام مختلفی مربوط به فرایند آموزش این ۳ مدل بررسی و مقایسه شدند. در ادامه این مدل‌ها در قالب مسابقات رو در رو مقایسه خواهند شد. در هر مسابقه عوامل هر تیم از یکی از مدل‌های تعلیم داده‌شده، استفاده کرده‌اند. مسابقه بین هر دو مدل ۵ راند تکرار شده است و برنده هر راند تیمی است که زودتر موفق به کسب ۱۰ امتیاز شده است. جداول ۵-۱، ۵-۲ و ۵-۳ نتایج این مسابقات را نشان می‌دهند.

جدول ۵-۱: نتایج رقابت‌های مدل‌های ۲ و ۳ لایه‌ای

| مدل / راند | ۲ لایه‌ای | ۳ لایه‌ای |
|---|---|---|
| اول | ۱۰ | ۹ |
| دوم | ۱۰ | ۷ |
| سوم | ۱۰ | ۸ |
| چهارم | ۱۰ | ۹ |
| پنجم | ۱۰ | ۴ |

جدول ۵-۲: نتایج رقابت‌های مدل‌های ۲ و ۴ لایه‌ای

| مدل / راند | ۲ لایه‌ای | ۴ لایه‌ای |
|---|---|---|
| اول | ۱۰ | ۴ |
| دوم | ۱۰ | ۵ |
| سوم | ۵ | ۱۰ |
| چهارم | ۱۰ | ۷ |
| پنجم | ۱۰ | ۳ |

جدول ۵-۳: نتایج رقابت‌های مدل‌های ۳ و ۴ لایه‌ای

| مدل / راند | ۳ لایه‌ای | ۴ لایه‌ای |
|---|---|---|
| اول | ۱۰ | ۸ |
| دوم | ۱۰ | ۶ |
| سوم | ۱۰ | ۶ |
| چهارم | ۱۰ | ۱ |
| پنجم | ۱۰ | ۳ |

نتایج به‌دست‌آمده نشان می‌دهد که در بازی رو در رو، مدل ۲ لایه‌ای بهتر از دو مدل دیگر عمل می‌کند. البته امتیازات مدل ۲ و ۳ لایه‌ای بسیار به هم نزدیک است. این شباهت در نمودارهای نمایش‌داده‌شده در قسمت‌های قبل نیز مشهود بود. از آن سو اما مدل ۴ لایه‌ای به‌مراتب ضعیف‌تر از دو مدل دیگر عمل می‌کند که این امر نیز در نمودارهای قسمت‌های قبلی مشخص بود.

### ۵-۲-۶- جمع‌بندی ارزیابی تاثیر تعداد لایه‌های شبکه

از نمودارها و نتایج حاصل از مسابقات رو در روی مدل‌ها، می‌توان نتیجه گرفت که در این پروژه، باتوجه‌به تعداد نه‌چندان زیاد مشاهدات که ورودی‌های مدل را تشکیل می‌دهند و همچنین میزان پیچیدگی روابط میان مشاهدات و کنش‌ها، ۲ یا ۳ لایه برای شبکه عصبی استفاده شده کافی است و تعداد بیشتر از این، علاوه بر افزایش زمان تعلیم، در نهایت به مدلی ناتوان‌تر در کشف خط‌مشی مناسب نیز تبدیل خواهد شد.

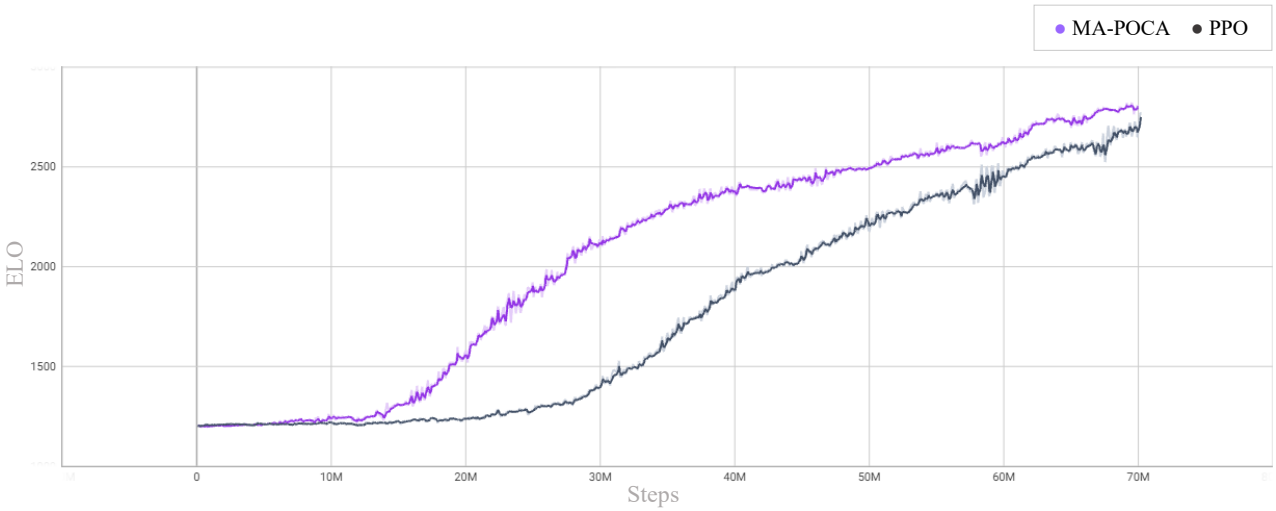
## ۵-۳- مقایسه PPO و MA-POCA

یکی دیگر از مواردی که در این پروژه به ارزیابی آن پرداخته شده است، بررسی تاثیر واقعی الگوریتم MA-POCA در فرایند یادگیری تیمی می‌باشد. همانطور که پیش‌تر بیان شد، MA-POCA الگوریتمی است که توسط توسعه دهندگان ML-Agents و به منظور توانمندسازی بیشتر یادگیری تیمی طراحی شده است. در واقع هدف از این الگوریتم تسهیل یادگیری فعالیت‌های مشارکتی به منظور دستیابی به یک هدف جمعی می‌باشد. اما اینکه این الگوریتم چه قدر به هدف خود نزدیک شده است را در ادامه و با مقایسه آن با الگوریتم PPO بررسی خواهیم کرد.

پیکربندی استفاده شده برای هر دو مدل تعلیم داده شد توسط این دو الگوریتم کاملاً یکسان و مشابه آنچه که در فصل قبل توضیح داده شد می‌باشد. شایان‌ذکر است که در این مقایسه از مدل ۳ لایه‌ای الگوریتم MA-POCA استفاده شده است و مدل تعلیم‌یافته با PPO نیز دارای ۳ لایه می‌باشد.

### ۵-۳-۱- سرعت یادگیری

نمودار شکل ۵-۵ سرعت و روند یادگیری این دو الگوریتم را با استفاده از امتیاز ELO کسب شده توسط آن‌ها در طی زمان یادگیری نشان می‌دهد. همانطور که مشخص است، مدل تعلیم یافته با MA-POCA نزدیک به ۱۰ میلیون گام سریع‌تر شروع به پیشرفت‌های محسوس کرده است و در تمامی زمان یادگیری نیز، همواره با اختلاف، موفق به ارتقای سطح توانمندی‌های خود شده است. البته با نزدیک شدن به گام‌های پایانی، این اختلاف کاهش‌یافته است.




شکل ۵-۵: نمودار مقایسه امتیاز ELO در مدل‌های PPO و MA_POCA

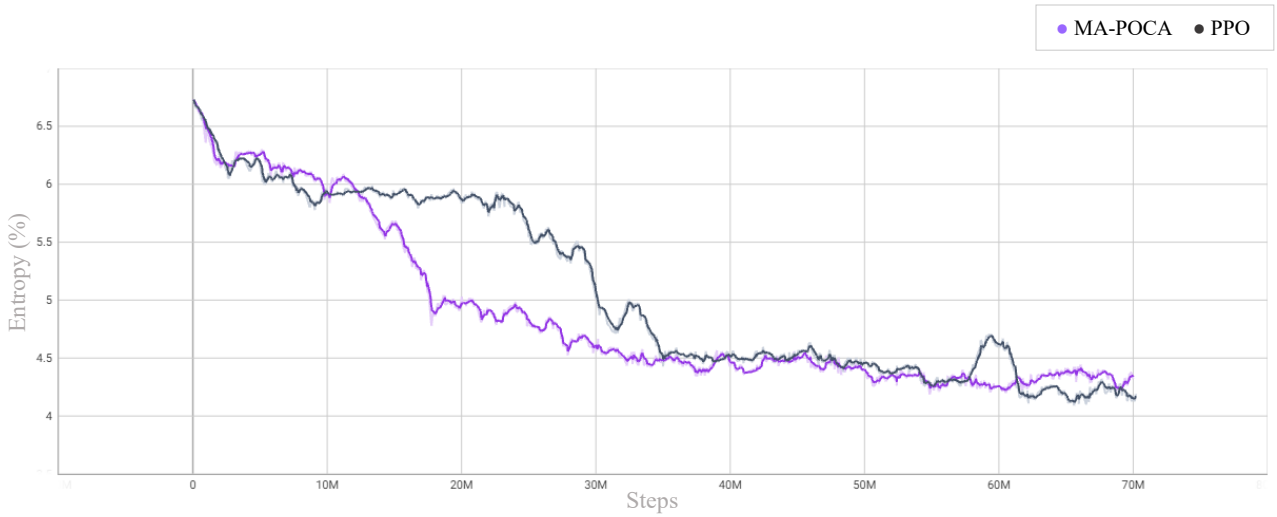
### ۵-۳-۲- میانگین طول episode

میانگین طول هر قسمت از بازی پس از ۷۰ میلیون گام، برای مدل تعلیم‌دیده با MA-POCA نزدیک به ۱.۵ ثانیه بیشتر است. این حقیقت نشان‌دهنده آن است که عامل‌هایی که با MA-POCA تعلیم‌دیده‌اند، بیشتر قادر به دریافت توپ و در جریان بازی نگه‌داشتن آن می‌باشند. میانگین قسمت‌های بازی در گام‌های مختلف در شکل ۵-۶ نشان‌داده‌شده است.


شکل ۵-۶: نمودار مقایسه میانگین طول episode در مدل‌های PPO و MA_POCA

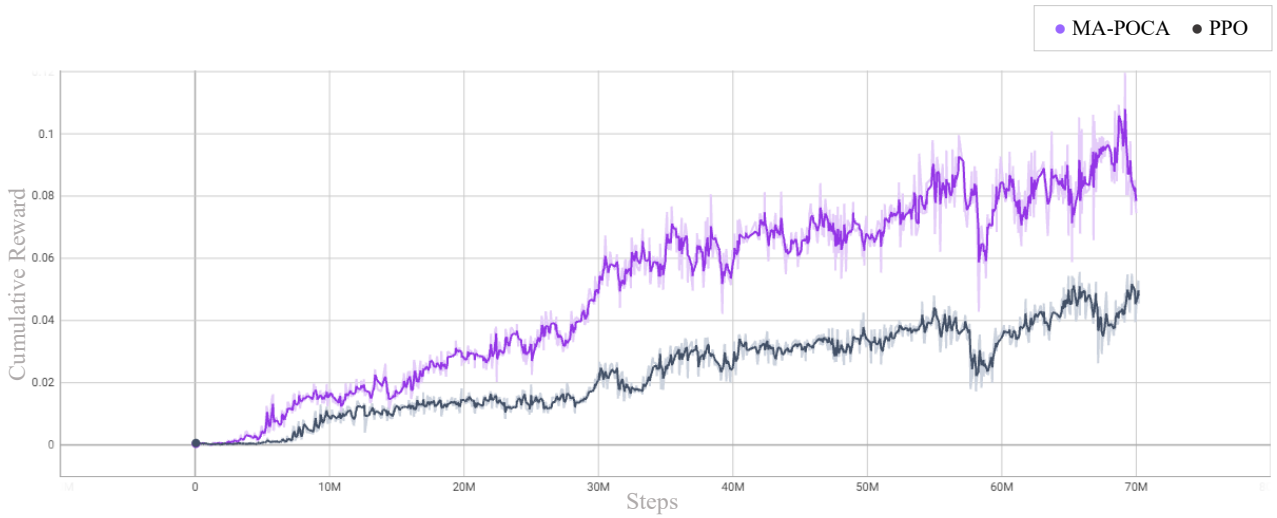
### ۵-۳-۳- Entropy

شکل ۵-۷ نشان‌دهنده درصد تصمیمات تصادفی اتخاذ شده توسط عامل به‌منظور اکتشاف در محیط می‌باشد. الگوریتم PPO همان‌طور که دیرتر آغاز به افزایش ELO کرد، حدود ۱۰ میلیون گام نیز دیرتر میزان کنش‌های رندوم خود را کاهش داده است. البته میزان تصمیمات تصادفی از یک‌زمان به بعد بسیار مشابه شده است.




شکل ۵-۷: نمودار مقایسه policy entropy در مدل‌های PPO و MA_POCA

### ۵-۳-۴- پاداش تجمیعی




شکل ۵-۸: نمودار مقایسه میانگین پاداش تجمیعی عوامل هم‌تیمی در مدل‌های PPO و MA_POCA

میانگین پاداش تجمیعی به‌دست‌آمده در طول فرایند یادگیری توسط عامل‌های یک تیم که در شکل ۵-۸ نمایش‌داده‌شده است، بیانگر آن است که عامل‌های تعلیم‌یافته توسط الگوریتم MA-POCA، به طور میانگین تقریباً دو برابر عامل‌های تعلیم‌یافته توسط PPO به توپ ضربه می‌زنند.

### ۵-۳-۵- رقابت مدل‌ها

آمار و ارقام بررسی شده، حاکی از برتری مدل تعلیم‌یافته با MA-POCA نسبت به مدل آموزش‌دیده با PPO می‌باشد. اما برای بررسی دقیق‌تر قدرت خط‌مشی ایجاد شده توسط این مدل‌ها، این دو در مقابل هم قرار داده شده‌اند و نتایج ۵ دور مسابقه بین این دو مدل در جدول ۵-۴ نشان‌داده‌شده است.

جدول ۵-۴: نتایج رقابت‌های مدل‌های PPO و MA_POCA

| مدل / راند | MA-POCA | PPO |
|---|---|---|
| اول | ۱۰ | ۵ |
| دوم | ۱۰ | ۴ |
| سوم | ۱۰ | ۴ |
| چهارم | ۱۰ | ۷ |
| پنجم | ۱۰ | ۴ |

### ۵-۳-۶- جمع‌بندی نتایج مقایسه PPO و MA-POCA

نتایج به‌دست‌آمده در بازی‌های رو در روی این دو مدل همانند نمودارهایی که پیش‌تر بررسی شد، نشان از برتری الگوریتم MA-POCA نسبت به PPO دارد. البته بررسی و مقایسه دقیق این دو الگوریتم و ارزیابی تأثیر MA-POCA در یادگیری تیمی و بهبود فعالیت‌های مشارکتی، نیاز به تست سناریوهای مختلف به همراه سنجش تمامی جوانب دارد. به طور مثال در این پروژه از پیکربندی یکسان برای هر دو الگوریتم استفاده شد. چه‌بسا با تغییر تنها یک یا چند پارامتر، PPO بتواند عملکرد بهتری از خودش نشان دهد. اما در قامت آزمایش انجام شده در محیط این پروژه، MA-POCA نشان داد که بسیار توانمندتر از PPO است.

## ۵-۴- جمع‌بندی

در این فصل در ابتدا مقایسه بین ۳ حالت مختلف شبکه عصبی ارائه شد. این ۳ حالت در واقع تعداد لایه‌های مختلف شبکه بودند. نتیجه این آزمایش نشان می‌داد که باتوجه‌به میزان مشاهدات عوامل از محیط در این پروژه که نمی‌توان آن را مقدار زیادی دانست و همچنین رابطه میان مشاهدات و کنش‌های مورد انتظار از

عامل، ایده‌آل‌ترین تعداد لایه شبکه ۲ خواهد بود درحالی‌که مدل دارای ۳ لایه نیز عملکرد بسیار نزدیک داشت.

دیگر آزمایش صورت‌گرفته در این پروژه بررسی عملکرد الگوریتم MA-POCA در یادگیری تیمی بود. به این منظور این الگوریتم با الگوریتم PPO مقایسه شد و نتایج به‌دست‌آمده نشان‌دهنده عملکرد بهتر MA-POCA بود.

# فصل ششم
# نتیجه‌گیری و پیشنهادات

## ۶-۱- نتیجه‌گیری

آنچه در این پروژه پیاده‌سازی شد، نه‌تنها یک شبیه‌ساز بازی والیبال، بلکه محیطی است برای انجام مطالعات و پژوهش‌های مرتبط با سیستم‌های چندعاملی و به‌ویژه استفاده از یادگیری تقویتی عمیق در این سیستم‌ها. ماهیت مشارکتی-رقابتی بازی والیبال این امکان را می‌دهد تا خصیصه‌های همکاری و رقابت نیز به‌خوبی در این سیستم قابل‌بررسی و ارزیابی باشند. پیاده‌سازی محیط این بازی به‌گونه‌ای است که می‌توان به‌راحتی پارامترهای سیستم از قبیل تعداد عامل‌های هر تیم را تغییر داد و این انعطاف‌پذیری در کنار قابلیت‌های وسیع ML-Agents و Unity این امکان را می‌دهد تا کارهای مطالعاتی مختلف، در یک فضای شبیه‌سازی‌شده نزدیک به واقعیت و با صرف هزینه و زمان بسیار کمتر صورت گیرد، همان‌طور که دو مثال از این کارهای مطالعاتی نیز در همین پروژه صورت گرفت و نتایج آن در فصل ۵ شرح داده شد.

این پروژه و سایر پروژه‌های مشابه می‌توانند گامی به‌سوی استفاده از یادگیری تقویت عمیق در کاربردهای دنیای واقعی و به‌ویژه سیستم‌های چندعاملی آن باشند. ML-Agents و Unity در کنار هم ابزاری بسیار قدرتمند را شکل می‌دهند که یکی از مهم‌ترین مزیت‌های آن می‌تواند شبیه‌سازی عوامل و سیستم‌های دنیای واقعی به‌منظور تعلیم مدل‌های یادگیری تقویتی عمیق برای آن‌ها در یک فضای آزمایشگاهی و کنترل شده باشد.

یقیناً در سال‌های آینده و با گسترش استفاده از یادگیری تقویتی در عامل‌های دنیای واقعی، از این ابزار یعنی ML-Agents بیشتر خواهیم شنید.

## ۶-۲- پیشنهادات برای کارهای آینده

دو چالش اساسی یادگیری تقویتی و یادگیری تقویتی عمیق که این پروژه نیز با آن‌ها مواجه شد، سرعت یادگیری و بهینگی محدود مدل‌های ایجاد شده می‌باشد. رفع این دو چالش می‌توانند به‌عنوان زمینه‌ای مهم برای انجام پروژه‌ها و پژوهش‌های آتی در نظر گرفته شود.

همان‌طور که پیش‌تر بیان شد، در این پروژه از دو تکنیک برای افزایش سرعت یادگیری عوامل استفاده شد. بااین‌حال زمان موردنیاز برای تعلیم برخی از مدل‌ها به بیش از ۱۵ ساعت می‌رسید. یافتن راهکارهایی برای افزایش سرعت یادگیری، چه از طریق بهینه‌سازی الگوریتم‌های یادگیری تقویتی از منظر نیاز به داده و نمونه‌های جمع‌آوری شده از محیط و چه از طریق افزایش نرخ ورودی‌های مدل می‌تواند گام بزرگی در توانمندسازی هرچه بیشتر این شاخه از هوش مصنوعی باشد.

زمینه دیگری که بسیار جای کار دارد، بهبود الگوریتم‌های موجود و یا توسعه الگوریتم‌های قوی‌تر است. مشکلی که بسیاری از پروژه‌ها و مسائل یادگیری تقویتی با آن دست‌وپنجه نرم می‌کنند، افزایش توانمندی عامل‌ها تنها تا سطحی محدود است. موردی که در این پروژه نیز با آن مواجه شدیم و درحالی‌که شیب یادگیری عوامل تقریباً ثابت شده بود، اما همچنان خطاهایی از آن‌ها سر می‌زد.

## پیوست ۱:

آدرس ریپازیتوری جعبه ابزار ML-Agents:

https://github.com/Unity-Technologies/ml-agent

آدرس ریپازیتوری پروژه (کدها و مدل‌های تمامی نسخه‌های شرح داده‌شده از بازی در پایان‌نامه، در این ریپازیتوری قرار دارد):

https://github.com/Pourmohammadi/Multi-Agent-DeepRL-Volleyball-Game

## منابع:


[1] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Coggan, M. (2004). Exploration and exploitation in reinforcement learning. *Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University*.

[3] Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. "Deep reinforcement learning: A brief survey." *IEEE Signal Processing Magazine* 34, no. 6 (2017): 26-38.

[4] Xu, Xin, Lei Zuo, and Zhenhua Huang. "Reinforcement learning algorithms with function approximation: Recent advances and applications." *Information Sciences* 261 (2014): 1-31.

[5] Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).

[6] Cohen, Andrew, Ervin Teng, Vincent-Pierre Berges, Ruo-Ping Dong, Hunter Henry, Marwan Mattar, Alexander Zook, and Sujoy Ganguly. "On the use and misuse of absorbing states in multi-agent reinforcement learning." *arXiv preprint arXiv:2111.05992* (2021).

[7] Busoniu, Lucian, Robert Babuska, and Bart De Schutter. "Multi-agent reinforcement learning: A survey." In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1-6. IEEE, 2006.

[8] Juliani, Arthur, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy et al. "Unity: A general platform for intelligent agents." *arXiv preprint arXiv:1809.02627* (2018).

[9] Zhao, Zhuoya, Feifei Zhao, Yuxuan Zhao, Yi Zeng, and Yinqian Sun. "A brain-inspired theory of mind spiking neural network improves multi-agent cooperation and competition." *Patterns* (2023).

[10] Hao, Jianye, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, Peng Liu, and Zhen Wang. "Exploration in deep reinforcement learning: From single-agent to multiagent domain." *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[11] Liu, Iou-Jen, Unnat Jain, Raymond A. Yeh, and Alexander Schwing. "Cooperative exploration for multi-agent deep reinforcement learning." In *International Conference on Machine Learning*, pp. 6826-6836. PMLR, 2021.

[12] Mao, Hangyu, Zhibo Gong, and Zhen Xiao. "Reward design in cooperative multi-agent reinforcement learning for packet routing." *arXiv preprint arXiv:2003.03433* (2020).

[13] Wang, Xiaoqiang, Liangjun Ke, Zhimin Qiao, and Xinghua Chai. "Large-scale traffic signal control using a novel multiagent reinforcement learning." *IEEE transactions on cybernetics* 51, no. 1 (2020): 174-187.

[14] Li, Tianxu, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Qihui Wu, Yang Zhang, and Bing Chen. "Applications of multi-agent reinforcement learning in future internet: A comprehensive survey." *IEEE Communications Surveys & Tutorials* 24, no. 2 (2022): 1240-1279.

[15] Ospanov, Bakhtiyar. "Training intelligent tennis adversaries using self-play with ML agents." (2021).

[16] Bayona Latorre, Andrés Leonardo. "Comparative study of SAC and PPO in multi-agent reinforcement learning using unity ML-agents." (2023).

[17] Göllner, Sabrina. "A competitive 3D-Volleyball-Game using Reinforcement Learning with Unity ML-Agents."