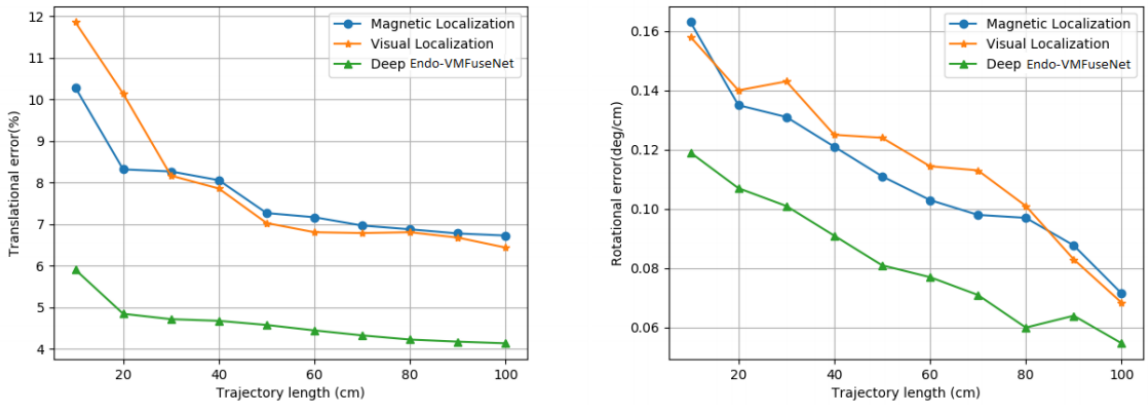


(a) Trajectory length vs translation error

(b) Trajectory length vs rotation error

شکل ۲-۱۱ مقایسه نتایج روش موقعیت‌یابی ترکیبی یادگیری عمیق با روش های موقعیت‌یابی براساس حسگر مغناطیسی و تصویر[۱۱۱].

عملکرد رویکرد عمیق Endo-VMFuseNet با استفاده از برآورد میانگین خطای میانگین مربعات ریشه (RMSE[1]) برای حرکات انتقالی و چرخشی مورد تجزیه و تحلیل قرار گرفت. برای مسیری با پیچیدگی‌های مختلف، مانند مسیرهای بدون مانع با انتقال و چرخش‌های آهسته و اسکن‌های جامع و مسیرهای پیچیده با حرکات چرخشی و انتقالی سریع، آزمایشات Endo-VMFuseNet عمیق در مقایسه با سیستم موقعیت‌یابی مبتنی بر تصویر و موقعیت‌یابی مغناطیسی انجام شده است. میانگین RMSEهای انتقالی و چرخشی برای Endo-VMFuseNet عمیق، موقعیت‌یابی مبتنی بر تصویر به تنهایی و موقعیت‌یابی مغناطیسی به تنهایی در طول مسیرهای مختلف به ترتیب در شکل ۲-۱۱ نشان داده شده است. نتایج نشان می‌دهد که Endo-VMFuseNet عمیق به وضوح بهتر از موقعیت‌یابی مبتنی بر تصویر به تنهایی و موقعیت‌یابی مغناطیسی به تنهایی عمل می‌کند. استفاده موثر از معماری LSTM در معماری Endo-VMFuseNet امکان یادگیری از آرایه سنسور ناهمزمان و غیر کالیبره شده را فراهم کرده است. نتایج همچنین نشان می‌دهد که Endo-VMFuseNet با تفسیر اطلاعات موقعیت‌یابی از داده‌های مغناطیسی فعلی و اطلاعات موقعیت‌یابی مبتنی بر تصویر قبلی ذخیره شده توسط حافظه پنهان داخلی واحدهای LSTM، قادر به مدیریت داده‌های ناهمزمان (داده‌های مغناطیسی ۵۰ هرتز و داده‌های تصویری ۳۰ هرتز) است.

۲-۲. محیط شبیه سازی

ابزار ML-Agents از سال ۲۰۱۷ آغاز به کار کرده است و امکانات زیادی در زمینه یادگیری تقویتی عمیق در اختیار محققان، توسعه دهندگان و برنامه نویسان بازی قرار داده است. این ابزار از ابتدا با هدف کمک به محققان و توسعه دهندگان در تبدیل بازی ها و شبیه سازی‌های ایجاد شده با محیط توسعه Unity به محیط‌های یادگیری عمیق ایجاد شد که در آن عامل ها می‌توانند با استفاده از پیشرفته ترین الگوریتم‌های یادگیری تقویتی عمیق،

[1] Root-mean-square deviation

استراتژی‌های تکاملی و ژنتیکی، و سایر روش‌های یادگیری عمیق (شامل بینایی کامپیوتر، تولید داده‌های مصنوعی) صرفا از طریق یک API ساده شده پایتون آموزش داده شوند.

۲-۲-۱. پلتفرم یونیتی (Unity)

یونیتی (Unity) یک پلتفرم توسعه سه بعدی بلادرنگ است که از یک موتور رندر[1] تصاویر و یک موتور فیزیک و همچنین یک رابط کاربری گرافیکی[2] به نام Unity Editor تشکیل شده است. یونیتی به طور گسترده در صنایع بازی، AEC[3] (معماری، مهندسی، ساخت و ساز)، خودرو و صنایع فیلم پذیرفته شده است و توسط جامعه بزرگی از توسعه دهندگان بازی برای ساخت انواع شبیه سازی‌های تعاملی، از بازی‌های کوچک موبایل و مبتنی بر مرورگر گرفته تا بازی‌های کنسولی و بازی‌های AR/VR[4] استفاده می‌شود. استفاده از موتور یونیتی می‌تواند پیشرفت تحقیقاتی را در زمینه‌های عقب مانده و حیاتی تسریع کند.

پیشینه تمرکز یونیتی بر توسعه یک موتور همه منظوره برای پشتیبانی از انواع پلتفرم‌ها، انواع توسعه‌دهندگان و انواع بازی ها، موتور یونیتی را به یک کاندیدای ایده‌آل پلتفرم شبیه‌سازی برای تحقیقات هوش مصنوعی تبدیل می‌کند. انعطاف‌پذیری زیرساخت موتور، امکان ایجاد برنامه‌هایی از دنیای شبکه دوبعدی ساده تا بازی‌های استراتژیک پیچیده سه بعدی، پازل‌های مبتنی بر فیزیک و بازی‌های رقابتی چند عامله را ممکن می‌سازد. برخلاف بسیاری از پلتفرم‌های تحقیقاتی، زیرساخت موتور به هیچ سبک خاصی از محیط بازی یا شبیه‌سازی محدود نمی‌شود و یونیتی را به یک پلتفرم همه منظوره تبدیل می‌کند. علاوه بر این، محیط ویرایشگر یونیتی، نمونه سازی سریع، توسعه بازی و محیط‌های شبیه سازی شده را امکان پذیر می‌کند.

یک پروژه یونیتی از مجموعه ای از دارایی ها[5] تشکیل شده است. صحنه ها[6] نوع خاصی از دارایی هستند که محیط یک پروژه را مشخص می‌کنند. صحنه ها ترکیبی سلسله مراتبی از اشیاء بازی[7] هستند که با اشیاء واقعی (چه فیزیکی و چه کاملاً منطقی) در محیط مطابقت دارد. رفتار و عملکرد هر کدام از اشیاء بازی توسط اجزای متصل به آن تعیین می‌شود. این اجزا[8] میتواند انواع مختلفی داشته باشند همچون دوربین‌ها، مش‌ها، رندرها، ساختارهای غیر منعطف و بسیاری اجزای دیگر که توسط ویرایشگر یونیتی ارائه می‌شوند. همچنین می‌توان اجزای سفارشی[9] را با استفاده از اسکریپت‌های C# یا افزونه‌های خارجی تعریف کرد [۱۱۶].

[1] Render
[2] GUI (Graphical user interface)
[3] Architecture, Engineering & Construction
[4] Augmented reality (AR) and Virtual Reality (VR)
[5] Asset
[6] Scene
[7] GameObjects
[8] Components
[9] Custom components

۲-۲-۲. ویژگی‌های موتور Unity

موتور یونیتی پیچیدگی لازم را جهت ایجاد محیط‌های یادگیری چالش برانگیز امکان‌پذیر می‌کند.

۲-۲-۲-۱. ویژگی‌های محیط

پیچیدگی سنسور - موتور یونیتی رندر گرافیکی با کیفیت بالا را امکان پذیر می‌کند. این موتور از نورپردازی پیش ساخته و نورپردازی زمان واقعی پشتیبانی می‌کند. همچنین توانایی تعریف سایه بان‌های سفارشی به صورت برنامه نویسی یا از طریق یک زبان برنامه نویسی بصری را دارد. بنابراین، می‌توان به سرعت تصاویری از محیط رندر کرد که کاملا شبیه عکس برداری واقعی از محیط باشد و این تصاویر را به عنوان داده‌های آموزشی[1] در مدل یادگیری ماشین استفاده کرد. بعلاوه با استفاده از سایه‌بان‌های سفارشی میتوان اطلاعات عمق، پوشش اشیا، مادون قرمز و تصاویر با نویز تزریق شده رندر کرد. موتور یونیتی وسیله‌ای برای تعریف سیگنال‌های صوتی ارائه می‌کند که می‌تواند به عنوان اطلاعات مشاهده شده درعوامل یادگیری به کار گرفته شود. پس قابلیت شبیه سازی سیستم‌های تشخیصی مبتنی بر پرتو[2] مانند Lidar را نیز دارد.

پیچیدگی فیزیکی - پدیده‌های فیزیکی در محیط‌های یونیتی را می‌توان با موتورهایNvidia PhysX یا Havok Physics شبیه سازی کرد. که اینکار، امکان تحقیق در محیط‌هایی با بدنه صلب، بدنه نرم، دینامیک ذرات، دینامیک سیالات و همچنین فیزیک ragdoll را فراهم می‌کند. در صورت تمایل، ماهیت توسعه پذیر بودن این پلتفرم امکان استفاده از موتورهای فیزیکی دیگر[3] را فراهم می‌کند. برای مثال، پلاگین‌هایی برای یونیتی وجود دارد که موتورهای فیزیک Bullet و MuJoCo را به عنوان جایگزین PhysX به کار می‌گیرد.

پیچیدگی منطق کار - موتور یونیتی یک سیستم اسکریپت نویسی قدرتمند و انعطاف پذیر از طریق C# ارائه می‌دهد. این سیستم پیاده سازی هر شکلی از گیم پلی، شبیه سازی و کنترل دینامیکی را امکان پذیر می سازد. علاوه بر زبان اسکریپت نویسی، اشیاء بازی و سیستم اجزا[4] امکان مدیریت همزمان چندین نمونه از عوامل، سیاست ها و محیط ها را فراهم می‌کند و امکان تعریف وظایف پیچیده سلسله مراتبی و همچنین وظایفی که برای حل آنها به یادگیری نیاز است را ممکن می سازد.

پیچیدگی اجتماعی - ماهیت زبان اسکریپت نویسی یونیتی و سیستم اجزا، نمایش سناریوهای چند عامله را بسیار ساده و راحت می‌کند. در واقع، از آنجایی که این پلتفرم برای پشتیبانی از توسعه بازی‌های ویدیویی چند نفره طراحی شده است، چندین اجزای انتزاعی مفید مانند سیستم شبکه چند نفره از قبل طراحی شده است.

[1] Training Data
[2] ray-cast
[3] 3rd party
[4] component

۲-۲-۲-۲. ویژگی‌های شبیه سازی

شبیه سازی سریع و توزیع شده - در موتور یونیتی شبیه سازی فیزیک و رندر کردن فریم ها به صورت ناهمزمان انجام می‌شود. لذا بدون نیاز به افزایش نرخ فریم در فرآیند رندر، می‌توان سرعت شبیه‌سازی را تا حد زیادی افزایش داد. همچنین اگر شبیه سازی نیاز به رندر کردن نیازی نداشته باشد می‌توان شبیه سازی‌های یونیتی را بدون رندر نیز اجرا کرد که در این حالت یونیتی منابع پردازشی بسیار کمتری استفاده می‌کند. در سناریوهایی که رندر کردن لازم است، مانند یادگیری از پیکسل ها، می‌توان نرخ فریم و سرعت منطق بازی را کنترل کرد. همچنین کنترل زیاد روی کیفیت رندر این امکان را فراهم می‌کند که هر زمان نیاز باشد نرخ فریم تا حد زیادی افزایش یابد. اما قابلیت‌های اضافه‌شده موتور یونیتی در شبیه‌سازی‌های مقیاس بزرگ، سربار اضافی به همراه دارد. به طوری که میزان حافظه رم نیاز برای شبیه سازی یونیتی از محیط‌های دیگر پلتفرم ها مانند ALE بیشتر است.

کنترل انعطاف پذیر - کنترل بیشتر جنبه‌های شبیه سازی به صورت برنامه نویسی امکان پذیر است. این ویژگی محققان را قادر می سازد تا برنامه‌های آموزشی، سناریوهای متخاصم یا سایر روش‌های پیچیده ای که محیط را در طول فرآیند آموزش تغییر می‌دهد، تعریف کنند. به عنوان مثال، اشیاء بازی را می‌توان به صورت مشروط ایجاد و به صورت بلادرنگ از بین برد. درنهایت موتور یونیتی از طریق پارامترهای بیرونی شبیه‌سازی و یک رابط برنامه نویسی کاربردی (API[1]) پایتون[2]، کنترل بیشتری در شبیه‌سازی را امکان‌پذیر می‌کند.

۲-۲-۳. نرم افزار ویرایشگر Unity

ویرایشگر یونیتی (شکل ۱) یک رابط کاربری گرافیکی است که برای ایجاد محتوای ۲ بعدی و ۳ بعدی استفاده می‌شود. نرم افزار یونیتی در سیستم عامل‌های ویندوز، مک و لینوکس موجود است و مزایای زیادی برای تحقیقات هوش مصنوعی فراهم می‌کند. از جمله آنها می‌توان به مواردی نظیر اشاره کرد:

- ایجاد صحنه‌های سفارشی: این امکان جهت آموزش هوش مصنوعی در محیط‌های مختلف با تجربه‌های گوناگون به کار می رود.
- نمایش متخصص: به این وسیله آموزش هوش مصنوعی به عهده یک متخصص گذاشته می‌شود. متخصص می‌تواند شبیه سازی را شروع کند و عامل‌های مختلف در صحنه را کنترل کند. بنابراین داده‌هایی تولید می‌شود که بر طبق آن آموزش انجام می‌شود. در حقیقت این ویژگی ما را قادر می سازد تا از الگوریتم‌های یادگیری تقلیدی[3] استفاده کنیم. این الگوریتم ها سرعت یادگیری ماشین را در محیط‌های پیچیده بسیار بالا می برد.

[1] Application Programming Interface
[2] Python
[3] Imitation Learning

- نمایش در مقیاس بزرگ: یکی از قدرتمند ترین ویژگی‌های یونیتی توانایی ساخت محیط شبیه سازی در بیش از ۲۰ پلتفرم شامل انواع رایانه، انواع موبایل، کنسولها و ... است. این ویژگی توانایی اندازه گیری کارایی از کاربران متنوع را فراهم می‌کند. همچنین در ویرایشگر یونیتی می‌توان همزمان چندین محیط را به صورت موازی ایجاد کرد که این کار منجر به افزایش نمایی سرعت یادگیری می‌شود.

۲-۲-۴. ابزار ML-Agent در Unity

ابزار ML-Agents یک پروژه منبع باز است که محققان و توسعه دهندگان را قادر می سازد محیط‌های شبیه سازی شده را با استفاده از ویرایشگر یونیتی ایجاد کرده و از طریق یک API پایتون با آنها تعامل داشته باشند. ML-Agents یک جعبه ابزار (SDK) ارائه می‌دهد که شامل تمام توابع لازم برای تعریف محیط ها در ویرایشگر یونیتی است. این توابع شامل اسکریپت‌های اصلی C# برای ساخت یک روند یادگیری است.

ویژگی‌های این جعبه ابزار شامل مجموعه‌ای از محیط‌های نمونه، الگوریتم‌های پیشرفته یادگیری تقویتی (SAC[1]) و (PPO[2])، الگوریتم‌های یادگیری تقلیدی (GAIL[3]) و شبیه‌سازی رفتاری (BC) است. پشتیبانی از Self-Play در بازی‌های متقارن و نامتقارن، و همچنین امکان گسترش الگوریتم‌ها و سیاست‌ها با روش‌های ICM و LSTM از دیگر ویژگی‌های این جعبه ابزار است.

در ابزار ML-Agents یک مفهوم به نام عامل ها[4] وجود دارد. مؤلفه عامل نشان دهنده اشیا بازی موجود در یک صحنه هستند که می‌تواند مشاهدات را جمع آوری کند، اقداماتی انجام دهد و پاداش دریافت کند. عامل می‌تواند مشاهدات را با استفاده از انواع حسگرها مبتنی بر اشکال مختلف داده مانند تصاویر رندر شده، پخش پرتو و ... در ابعاد مختلف برداری، جمع آوری کند. هر جزء عامل حاوی یک سیاست است که با یک نام رفتاری برچسب گذاری شده است.

هر تعداد عامل می‌تواند سیاستی با نام رفتاری مشابه داشته باشد که این عوامل همگی یک سیاست را اجرا می‌کنند و داده‌های تجربی را در طول آموزش به اشتراک می‌گذارند. به‌علاوه، می‌تواند هر تعداد نام رفتاری برای سیاست‌ها در یک صحنه وجود داشته باشد که ساخت سناریوهای چند عاملی را با گروه‌ها یا عامل‌های فردی، ممکن می‌سازد. یک سیاست می‌تواند برآمده از مکانیسم‌های تصمیم‌گیری مختلف از جمله ورودی بازیکن، اسکریپت‌های از پیش کدگذاری شده، مدل‌های شبکه عصبی تعبیه‌شده داخلی باشد یا اینکه از طریق تعامل با API پایتون مشخص شود. این امکان برای عامل ها وجود دارد تصمیم ها را از سیاست‌های ثابت یا پویا (همانطور که توسط توسعه دهنده محیط تعریف شده است)، درخواست کنند.

[1] Actor-Critic
[2] Proximal Policy Optimization
[3] Generative Adversarial Imitation Learning
[4] Agents

تابع پاداش، که برای ارائه سیگنال یادگیری به عامل استفاده می‌شود، می‌تواند در هر زمانی در طول شبیه سازی با استفاده از سیستم اسکریپت یونیتی تعریف یا تغییر یابد. به همین ترتیب، شبیه سازی را می‌توان در یک حالت انجام شده در سطح یک عامل فردی یا محیط قرار داد. این حالت یا از طریق فراخوانی اسکریپت یونیتی یا با رسیدن به حداکثر تعداد گام از پیش تعریف شده اتفاق می‌افتد.

در هر شبیه سازی یک آکادمی[1] وجود دارد که برای پیگیری مراحل شبیه سازی و مدیریت عامل ها استفاده می‌شود. آکادمی همچنین دارای قابلیت تعریف پارامترهای محیطی است که می‌توان از آن برای تغییر پیکربندی محیط در زمان اجرا استفاده کرد. به طور خاص، جنبه‌های فیزیک محیطی و بافت‌ها، اندازه‌ها و وجود اشیاء از طریق پارامترهای قابل دسترسی کنترل می‌شوند. پس می‌توانند مجدداً نمونه‌برداری شده و در طول آموزش تغییر کنند. به عنوان مثال، گرانش در محیط می‌تواند در هر بازه ثابتی نوسان کند یا موانع اضافی می‌تواند زمانی که یک عامل به مهارت خاصی برسد، ایجاد شود. این قابلیت، ارزیابی یک عامل را در محیط‌های مختلف و در شرایط یادگیری و تست فراهم می‌کند و ایجاد سناریوهای یادگیری برنامه آموزشی را تسهیل می‌کند.

۲-۲-۵. معیارهای عملکرد

برای یک شبیه سازی ضروری است که محیط قادر به ارائه سرعت شبیه سازی بیشتری نسبت به زمان واقعی باشد. در ابزار شبیه سازی ML-Agents امکان افزایش سرعت تا صد برابر زمان واقعی وجود دارد. با این حال در عمل افزایش سرعت، ممکن مطابق با منابع پردازشی موجود و همچنین پیچیدگی‌های محیط متفاوت باشد. در موتور یونیتی، منطق بازی می‌تواند مستقل از رندر کردن فریم ها اجرا شود. به این ترتیب، محیط‌هایی که به مشاهدات بصری متکی نیستند، مانند محیط‌هایی که از پرتوها استفاده نمی‌کنند، می‌توانند از شبیه‌سازی با سرعت بیشتری نسبت به محیط‌هایی که استفاده می‌کنند بهره ببرند.

۲-۲-۶. اجزای اصلی ابزار ML-Agents

ابزار ML-Agents از ۴ جزء اصلی سطح بالا تشکیل شده است:

- ***محیط یادگیری*** - شامل صحنه یونیتی و تمامی شخصیت‌های بازی می باشد. صحنه یونیتی محیطی را فراهم می‌کند که در آن عامل ها محیط را مشاهده می‌کنند، در آن عمل‌های متفاوت انجام می‌دهند و همچنین از آن محیط یاد می‌گیرند. نحوه تنظیم صحنه یونیتی برای استفاده به عنوان یک محیط یادگیری به هدف آموزش بستگی دارد. صحنه ها هم جهت آموزش و هم جهت آزمایش استفاده می شوند. پس برای یک مسئله یادگیری تقویتی خاص با دامنه محدود می‌توان از یک صحنه همزمان برای آموزش و آزمایش عوامل آموزش دیده استفاده کرد. همچنین به منظور آموزش عامل ها در یک بازی پیچیده یا شبیه سازی، می‌توان از صحنه‌های مجزا بهره برد که ممکن است کارآمدتر و کاربردی‌تر

[1] Academy

باشد. جعبه ابزار ML-Agents به وسیله بسته پایتون این امکان را می‌دهد که با تعریف عامل ها و رفتارهای آنها، هر صحنه یونیتی را به یک محیط یادگیری تبدیل کنید.

- ***رابط برنامه نویسی کاربردی پایتون*** - شامل یک رابط پایتون سطح پایین برای تعامل و دستکاری یک محیط یادگیری است. برخلاف محیط یادگیری، API پایتون بخشی از یونیتی نیست، بلکه یک ماژول خارجی است و از طریق ارتباط دهنده[1] با یونیتی ارتباط برقرار می‌کند. این API در یک بسته اختصاصی پایتون موجود است و توسط فرآیند آموزش پایتون برای برقراری ارتباط و کنترل آکادمی در طول آموزش استفاده می‌شود. با این حال، می‌توان آن را برای اهداف دیگری نیز مانند استفاده از یونیتی به عنوان موتور شبیه سازی الگوریتم‌های یادگیری ماشین، بکار برد.
- ***ارتباط دهنده خارجی*** - محیط یادگیری را به API پایتون متصل می‌کند.
- ***موتور آموزش دهنده پایتون*** - که شامل تمام الگوریتم‌های یادگیری ماشینی است که عامل ها را آموزش می‌دهد. این الگوریتم‌ها در پایتون پیاده‌سازی شده‌اند و فقط با API سطح پایین پایتون رابط کاربری دارد.

۲-۲-۶-۱. محیط یادگیری

محیط یادگیری شامل دو مؤلفه است که به سازماندهی صحنه یونیتی کمک می‌کند:

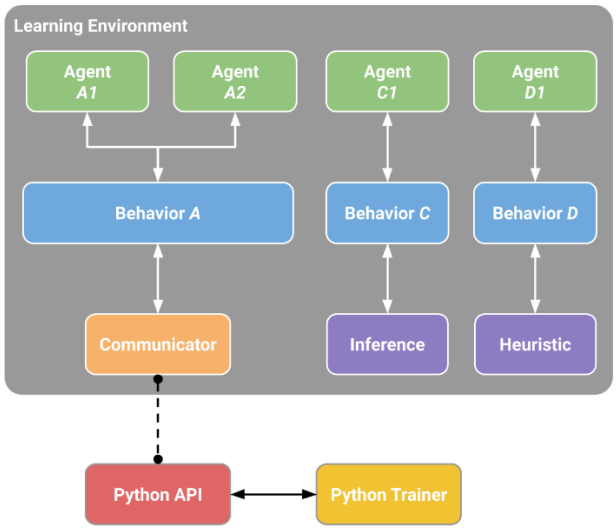
- ***عامل‌ها*** - که به یک شی از یونیتی (هر شی در صحنه) متصل است و مشاهدات آن را ایجاد می‌کند، اقداماتی را که دریافت می‌کند انجام می‌دهد و در صورت لزوم پاداش (مثبت / منفی) اختصاص می‌دهد. هر عامل به یک رفتار پیوند خورده است.
- ***رفتار***[2] - ویژگی‌های خاص عامل را تعریف می‌کند، مانند تعداد اقداماتی که عامل می‌تواند انجام دهد. یک رفتار را می‌توان به عنوان عملکردی در نظر گرفت که مشاهدات و پاداش ها را از عامل دریافت می‌کند و اعمال را بر می‌گرداند. یک رفتار می‌تواند یکی از سه نوع باشد: یادگیری، اکتشافی یا استنتاج.

هر محیط آموزشی همیشه یک عامل برای هر شخصیت در صحنه خواهد داشت. در حالی که هر عامل باید به یک رفتار پیوند خورده باشد، این امکان وجود دارد که عامل‌هایی که مشاهدات و اقدامات مشابهی دارند، رفتار مشابهی داشته باشند و این بدان معنا نیست که در هر کدام، ارزش‌های مشاهده و عمل یکسانی خواهند داشت. در یک محیط، می‌تواند همزمان چندین عامل و چندین رفتار وجود داشته باشد. محیط یادگیری از طریق آکادمی

[1] Communicator
[2] Behavior

(که در نمودار نشان داده نشده است) تضمین می‌کند که همه عامل ها علاوه بر کنترل تنظیمات محیطی، همگام هستند. به طور کلی آکادمی موارد زیر را تضمین می‌کند:

- جمع آوری مشاهدات
- انتخاب عمل با استفاده از سیاست
- انجام عمل
- ریست کردن یادگیری اگر یادگیری انجام شد یا به حداکثر مراحل خود رسید.



شکل ۲-۱۲ محیط یادگیری.

۲-۲-۶-۲. معماری رفتار -آکادمی

داشتن عامل‌هایی که به وسیله ی شبکه عصبی کنترل می شوند، مفهوم اساسی در ساخت محیط بازی یا شبیه سازی به وسیله ML-Agents به شمار می آید. هر کدام از این عالم ها به یک رفتار پیوند خورده‌اند که می‌توان به‌طور زیر طبقه‌بندی کرد:

- ***رفتار استنتاجی*** - این نوع معماری شامل مدل‌های از پیش آموزش دیده است که با استفاده از الگوریتم‌های مختلف یادگیری عمیق آموزش داده شده‌اند. رفتار استنتاجی برای آموزش، نیازی به ارتباط با API پایتون در زمان اجرا ندارد، زیرا قبلاً روی یک مجموعه از پیش موجود فضای مشاهده-عمل آموزش داده شده است.
- ***رفتار ابتکاری*** - این نوع معماری معمولاً دارای هیچ مدل شبکه عصبی نیست. بلکه شامل یک راه حل ابتکاری است که در این رفتار گنجانده شده است. به جای استفاده از یادگیری عمیق، رفتار ابتکاری بیشتر بر پایه برنامه نویسی پویا و یادگیری مبتنی بر حالت، توسعه می یابند.

- ***رفتار یادگیری*** - این رفتار نوع دیگری است که از الگوریتم‌های یادگیری عمیق استفاده می‌کند و در هنگام آموزش عامل در یونیتی استفاده می‌شود. رفتار یادگیری برای آموزش عامل ها در زمان اجرای آموزش از طریق ارتباط دهنده و API پایتون به یونیتی متصل می‌شود. ارتباط دهنده، نقش مهمی در برقراری ارتباط روی پورت ۵۰۰۴ ایفا می‌کند. به وسیله ارتباط دهنده می‌توان برای آموزش از مدل‌های Tensorflow بهره برد. همچنین مدل‌های آموزش دیده از رفتار یادگیری، به‌عنوان مدل‌های تولید شده و از پیش آموزش دیده شده برای رفتار استنتاجی نیز استفاده می شوند.

۲-۲-۶-۳. موتور استنتاج یونیتی: Barracuda

جعبه ابزار ML Agents برای اجرای مدل‌های شبکه عصبی از پیش آموزش دیده با استفاده از رفتار یادگیری، از موتور استنتاج یونیتی یا Barracuda بهره می برد. این یک کتابخانه سبک وزن به تبدیل مدل‌های شبکه عصبی از کتابخانه Tensorflow به فایل‌های سریالی با پسوند .nn کمک می‌کند. موتور استنتاج یونیتی از دو فرمت Barracuda و ONNX پشتیبانی می‌کند. تنظیمات موتور استنتاج یونیتی را می‌توان در داخل اسکریپت پارامترهای رفتاری مشاهده کرد. یک ویژگی مهم موتور استنتاج در بخش مدل وجود دارد که می‌توان به وسیله آن دستگاه پردازشی CPU یا GPU[1] را انتخاب کرد. این قابلیت موتور استنتاج را قادر می سازد تا متناسب با پروژه، بهترین منابع پردازشی را جهت آموزش استفاده کند.

۲-۲-۷. سناریوهای آموزشی انعطاف پذیر

با استفاده از ابزار ML-Agent تعریف کردن چندین سناریو جهت آموزش عامل ها امکان پذیر است:

- تک عامل: یک عامل واحد، با سیگنال پاداش خاص خود. این سناریو روش سنتی آموزش یک عامل مانند بازی‌های تک نفره است.
- تک عامل همزمان: چندین عامل مستقل با سیگنال‌های پاداش مستقل با پارامترهای رفتاری یکسان. یک نسخه موازی از سناریوی آموزشی سنتی است که می‌تواند روند تمرین را تسریع کند. این سناریو زمانی مفید است که چندین نسخه از یک شخصیت در محیطی دارید که باید رفتارهای مشابهی را بیاموزد. به هنوان مثال آموزش چند ربات بازو برای باز کردن یک در به صورت همزمان باشد.
- تقابل دو عامل: دو عامل در تعامل باهم اما با سیگنال‌های پاداش معکوس. در بازی‌های دو نفره، تقابل دو عامل می‌تواند به یک عامل اجازه دهد تا به طور فزاینده‌ای ماهرتر شود، در حالی که همیشه حریف کاملاً همسانی داشته باشد.

[1] Graphics processing unit

- چند عاملی تعاونی: چندین عامل تعاملی با یک سیگنال پاداش مشترک با پارامترهای رفتاری یکسان یا متفاوت. در این سناریو، همه عامل ها باید با هم همکار کنند تا کاری را انجام دهند که به تنهایی قابل انجام نیست. به عنوان مثال می‌توان به محیط‌هایی اشاره کرد که در آن هر عامل تنها به قسمتی از اطلاعات دسترسی دارد، که برای انجام کار یا حل مشترک یک مسئله باید به اشتراک گذاشته شود.
- چندعاملی رقابتی: چندین عامل تعاملی با سیگنال‌های پاداش معکوس با پارامترهای رفتاری یکسان یا متفاوت. در این سناریو، عامل ها باید با یکدیگر رقابت کنند تا یا در یک مسابقه برنده شوند یا مجموعه محدودی از منابع را به دست آورند. به عنوان مثال تمامی ورزش‌های تیمی در این سناریو قرار می‌گیرند.
- زیست بوم: چندین عامل تعاملی با سیگنال‌های پاداش مستقل با پارامترهای رفتاری یکسان یا متفاوت. این سناریو را می‌توان به‌عنوان ایجاد دنیای کوچکی در نظر گرفت که در آن عامل ها با اهداف متفاوت همه با هم تعامل دارند، مانند طبیعت که ممکن است در آن حیوانات مختلف وجود داشته باشد، یا یک شبیه‌سازی رانندگی مستقل در یک محیط شهری.

۲-۲-۷-۱. سیگنال‌های پاداش

در یادگیری تقویتی، هدف نهایی عامل، کشف رفتاری (سیاست) است که پاداش را به حداکثر می رساند. شما باید یک یا چند سیگنال پاداش را در اختیار عامل قرار دهید تا در طول آموزش از آنها استفاده کند. به طور معمول، یک پاداش توسط محیط برای رسیدن به هدف تعریف می‌شود. این نوع از پاداش، بیرونی است زیرا آنها خارج از الگوریتم یادگیری تعریف می شوند. با این حال، پاداش‌ها را می‌توان خارج از محیط نیز تعریف کرد، تا عامل را تشویق کند تا به روش‌های خاصی رفتار کند، یا به یادگیری پاداش بیرونی واقعی کمک کند. این پاداش ها به عنوان سیگنال‌های پاداش ذاتی (مانند GAIL، Curiosity و RND) بشمار می روند. کل پاداشی که عامل یاد می‌گیرد تا به حداکثر برسد، می‌تواند ترکیبی از سیگنال‌های پاداش بیرونی و درونی باشد. جعبه ابزار ML-Agents اجازه می‌دهد تا سیگنال‌های پاداش به صورت ماژولار تعریف شود.

۲-۳. یادگیری عمیق

۲-۳-۱. مقدمه ای بر یادگیری تقویتی

کنترل سیستمهای مکانیکی به سطح بالایی از درک سیستم متکی است [۱۱۷]، که عمدتا از طریق مدلهای دینامیک بیان می‌شود [۱۱۸]،[۱۱۹]. با این حال، این روش برای سیستم‌های دارای درجه پیچیدگی یا عدم اطمینان بالا محدودیت‌های زیادی دارد [۱۲۰]. به طور خاص، با در نظر گرفتن الگوریتم‌های کنترل مبتنی بر مدل، مانند

تکنیک‌های تنظیم کننده خطی-درجه دوم (LQR[1]) و پیشگام (BS[2])، استفاده از معادلات دیفرانسیل-جبری حرکت چالش برانگیز است. اینها به شدت به دقت مدل متکی هستند، که مانع کاربرد آنها در سیستم‌های واقعی می‌شود [۱۲۱]. بنابراین، توسعه کنترل کننده ها برای سیستم‌های غیرخطی بسیار چالش برانگیز، وقت گیر و در بسیاری از موارد کاری غیرقابل اجرا است. روش مهندسی سنتی شامل استخراج تحلیلی سیستم حاکم بر معادلات و تنظیم دستی پارامترهای سیستم کنترل متناسب با برخی پارامترهای فیزیکی اندازه گیری شده است. بنابراین تخصص پیشرفته در ریاضیات کاربردی، نظریه سیستم‌های دینامیکی، روش‌های محاسباتی، مدل سازی ریاضی، چارچوب‌های بهینه سازی و تنظیم الگوریتمی پارامترهای کنترل به کمک اپراتور مورد نیاز است.

۲-۳-۲. توضیح روش یادگیری تقویتی

یک روش جایگزین استفاده از روشهای داده-محور یادگیری ماشین است. با توجه به حجم عظیم داده، می‌توان با بهره گیری از تکنیک‌های مناسب ML مدل‌هایی را برای سیستم‌های پیچیده ایجاد کرد [۱۲۲]. علاوه بر این، شبیه سازی ها و داده‌های مصنوعی برای آموزش امکان استفاده از شبکه‌های عصبی در برنامه‌های متنوع با درجات مختلف پیچیدگی را فراهم می‌کند [۱۲۳]. به عنوان مثال، نویسندگان زیادی سعی در یادگیری تقلیدی برای به دست آوردن مهارت‌های حرکتی از تظاهرات انسانی دارند [۱۲۴]. با این حال، نتایج این روش ها در رباتیک نزدیک به بهینه است. راه حل استفاده از روش یادگیری تقویتی (RL) است. این یک روش ML است که بر روی بهینه سازی فرایندهای تصمیم گیری مارکوف تمرکز دارد. هدف RL یادگیری نحوه کنترل سیستم برای به حداکثر رساندن پاداش‌های تجمعی است. برخلاف یادگیری تحت نظارت یا بدون نظارت، این روش نمونه‌هایی را تولید می‌کند که بدون برچسب هستند، بنابراین فقط سیگنال‌های پاداش تنک را در نظر می‌گیرند [۱۲۵]. به طور کلی، بهینه سازی سیاست RL به سیگنال پاداش بستگی دارد. پاداش هنگامی می‌تواند موثر واقع شود که نشان دهد کدام سیاست و عملکرد منفرد بهتر از دیگران است [۱۲۶]. RL برای حل مشکلات کنترل بهینه، با اطلاعات محدود در مورد حالت دینامیکی به دانش قبلی کمی نیاز دارد [۱۲۷]. به طور کلی، روش‌های RL از نظر محاسباتی ساده بوده و نمایانگر روشی است که مستقیماً برای کنترل سیستم‌های غیرخطی [۱۲۸]، با ترکیب ویژگی‌های کنترل کننده‌های بهینه و تطبیقی، قابل استفاده است [۱۲۹]. در دهه گذشته، RL موفقیت زیادی در کنترل فضای کوچک گسسته ی عمل-حالت کسب کرده است. اخیراً، از طریق تقریب تابع، برای حل مشکلات مربوط به فضای بزرگ و پیوسته استفاده شده است [۱۳۰]، به این ترتیب پتانسیل قابل استفاده در رشته‌های متنوعی مانند مدیریت انرژی [۱۳۱]، وسایل نقلیه خودران [۱۳۲]، و تحلیل روند مالی [۱۳۳] را دارا است. بنابراین، یادگیری عمیق به روش اصلی برای دستیابی به RL موثر، تبدیل شده است [۱۳۴]. ترکیب این دو روش اصطلاحا (DRL[3]) نامیده می‌شود که در حال حاضر یک رویکرد امیدوار کننده برای هوش مصنوعی در حالت عمومی است [۱۳۵].

[1] Linear Quadratic Regulator
[2] Backstepping
[3] Deep RL

اخیرا، DRL در رباتیک مورد توجه بسیاری قرار گرفته است، جایی که عامل به کنترل کننده ربات تبدیل می‌شود و خود ربات و محیط اطراف آن محیط هستند [۱۳۶]. DRL در رباتیک یک مسئله ی کنترلی است که در آن یک ربات به عنوان عامل، با انتخاب متوالی اقدامات از طریق دنباله ای از مراحل زمانی، در یک محیط تصادفی عمل می‌کند [۱۳۷]. این چارچوب به یک عامل رباتیک این امکان را می‌دهد تا مهارت‌هایی را برای حل وظایف از طریق آزمون و خطا به طور مستقل بیاموزد. با این حال، ابعاد مسئله کنترلی رباتیک در فضاهای حالت پیوسته اغلب دارای ابعاد بالا هستند. به همین دلیل، متأسفانه راه حل گسسته سازی فضای عمل محدودیت‌های زیادی ایجاد می‌کند.

قوام در برابر عدم اطمینان در مدل سازی از بیست سال گذشته یکی از مهمترین موضوعات تحقیقاتی در سیستم‌های کنترل بوده است. این امر عمدتا ناشی از دشواری‌هایی است که هنگام انتقال یک مدل به دنیای واقعی پیدا می‌شود. به طور دقیق تر دلیل این امر ساده سازی بیش از حد در معادلات دینامیکی، پیش فرض‌های ساده کننده مسئله، دینامیک‌های مدل نشده و مقادیر تقریبی عددی خصوصیات فیزیکی است. از این منظر، مزیت اصلی RL توانایی یادگیری آن از تعامل با محیط براساس یافتن یک سیاست کنترل بهینه است، نه اینکه توسط یک طراح انسانی برنامه ریزی شود. با این حال، مشکلی که در اینجا ایجاد می‌شود همگرایی ضعیف RL بر اساس شبکه‌های عصبی است، که تاکنون یک چالش بزرگ برای ربات‌های خودمختار است. علاوه بر این، فعل و انفعالات فیزیکی متعددی معمولاً برای این سیستمها غیرقابل انجام است که شاید منجر به فرسودگی و آسیب دیدن سخت افزار شود. بنابراین، هرچه سیستم آسیب پذیرتر باشد، میزان کارایی داده در یادگیری نیز بیشتر است. تکنیک‌های ML و RL عمیق پتانسیل زیادی برای کاربردهای مهندسی کنترل دارند. اساساً دو نکته اساسی وجود دارد. اول، روش‌های RL امکان ایجاد کنترل کننده‌های غیرخطی موثر سیستم‌های مکانیکی را فراهم می‌کنند که تا حدی رفتار فیزیکی سیستم دینامیکی را که باید کنترل شود نادیده می‌گیرند. در حقیقت، با سیستم مکانیکی که باید کنترل شود، به عنوان یک جعبه سیاه که از آن داده‌های ورودی و خروجی جمع می‌شود، رفتار می‌شود. پس این روش امکان استفاده از استراتژی کنترلی طراحی شده را در طیف گسترده ای از سیستم‌های حالت فراهم می‌کند [۱۳۸].

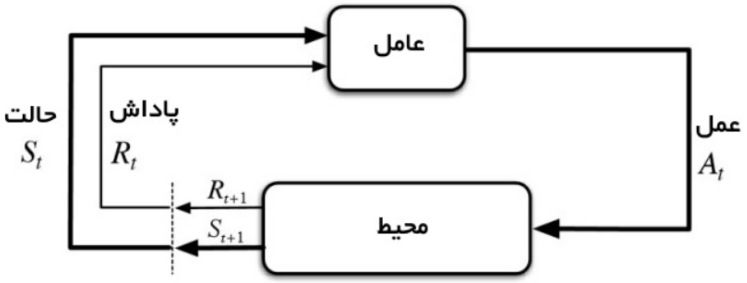
۲-۳-۳. الگوریتم‌های یادگیری تقویتی

برخی از اصطلاحاتی که عناصر یک مساله یادگیری تقویتی را تشریح می‌کنند در زیر بیان شده است:

- محیط[1]: جهان فیزیکی که عامل در آن عمل می‌کند.
- حالت[2]: موقعیت کنونی عامل
- پاداش[3]: بازخورد از محیط

[1] Environment
[2] State
[3] Reward

- سیاست[1]: روشی برای نگاشت حالت عامل به عمل
- ارزش[2]: پاداش آینده که یک عامل با اقدام به یک عمل در یک حالت خاص به آن دست می‌یابد.



شکل ۲-۱۳ الگوریتم‌های یادگیری تقویتی.

به منظور ساخت یک سیاست بهینه، عامل با مساله جستجوی حالت‌های جدید در حالیکه پاداش را نیز به طور هم‌زمان بیشینه می‌کند مواجه است. به این کار موازنه جستجو و استخراج[3] گفته می‌شود. فرآیندهای تصمیم‌گیری مارکوف[4] چارچوب‌های ریاضی هستند که برای تشریح محیط در یادگیری تقویتی استفاده می‌شوند و تقریبا همه مسائل این حوزه قابل رسمی شدن باMDP ها هستند. یک MDP شامل مجموعه‌ای متناهی از حالت‌های محیط S، مجموعه‌ای از اعمال ممکن $A(s)$ در هر حالت، یک تابع پاداش دارای مقدار حقیقی $R(s)$ و مدل انتقال $P(s', s|a)$ است. اگرچه، برای محیط‌های جهان واقعی فقدان هرگونه دانش اولیه‌ای از محیط وجود دارد. روش‌های مستقل از مدل یادگیری تقویتی در چنین شرایطی مفید واقع می‌شوند. Q-learning معمولا به عنوان یک رویکرد مستقل از مدل مورد استفاده قرار می‌گیرد و برای ساخت عامل خودبازی‌کن قابل استفاده خواهد بود. این روش حول محور به‌روز رسانی ارزش‌های Q است که ارزش انجام عمل a را در حالت s نشان می‌دهد. قاعده به‌روز رسانی ارزش، هسته الگوریتم Q-learning محسوب می‌شود.

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{مقدار قدیمی}} + \underbrace{\alpha}_{\text{نرخ یادگیری}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{پاداش}} + \underbrace{\gamma}_{\text{فاکتور تنزیل}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{تخمین ارزش آینده مطلوب}} \right)}^{\text{ارزش یاد گرفته شده}}$$

در این بخش مروری بر الگوریتم‌های اصلی یادگیری تقویتی استفاده از چارچوب‌های یادگیری عمیق گسترش صورت می‌گیرد.

- روش‌های مبتنی بر ارزش: این روش ها احتمال قرار گرفتن در یک حالت معین را تخمین می زنند، با استفاده از آن سیاست کنترل تعیین می‌شود. تخمین حالت متوالی با استفاده از معادلات بلمن (معادله امید بلمن و معادله بهینه سازی بلمن) انجام می‌شود. الگوریتم‌های RL مبتنی بر ارزش شامل

[1] Policy
[2] Value
[3] Exploration vs Exploitation trade-off
[4] Markov Decision Processes (MDP)

(SARSA[1]) و Q-Learning هستند، که در اهدافشان متفاوت هستند، این همان مقدار هدف است که Q به صورت بازگشتی و به‌اندازه یک پله در هر مرحله آپدیت می شوند. SARSA یک روش برمبنای سیاست است که در آن تخمین ارزش نسبت به یک سیاست به روز می‌شود در حالی که Q-Learning، یک روش بدون سیاست است که تخمین ارزش را به سمت یک سیاست بهینه هدف به روز می‌کند. این الگوریتم یک الگوریتم پیچیده است که برای بیان مشکلات مختلف ظاهر چندگانه استفاده می‌شود اما محدودیت‌های محاسباتی به مانع بهره بردن از آن می‌شود.

- روشهای مبتنی بر سیاست: برخلاف روشهای مبتنی بر ارزش، روشهای مبتنی بر سیاست مستقیماً سیاست را به روز می‌کنند بدون اینکه به تخمین ارزش نگاه کنند و از نظر همگرایی، حل مسائل با داده‌های پیوسته در ابعاد بالا و حل سیاستهای غیرتصادفی به طور اثربخش، کمی بهتر از روشهای مبتنی بر ارزش هستند. آنها به دو روش مبتنی بر گرادیان و بدون گرادیان[۳۱،۳۲] پارامترها را تخمین می زنند. اینجا بر روی روشهای مبتنی بر گرادیان تمرکز می‌شود که الگوریتم بهینه سازی گرادیان نزولی انتخاب شده است. در اینجا، ما تابع هدف را بهینه می‌کنیم:

$$J(\pi_\theta) = E_{\pi_\theta}[f_{\pi_\theta}(.)] \quad (۱-۲)$$

که در آن تابع امتیاز[۳۳] برای سیاست π_θ توسط $f_{\pi_\theta}(.)$ داده می‌شود. با استفاده از معادله بالا، می‌توانیم در مورد عملکرد مدل با توجه به وظیفه انجام شده نظر دهیم. الگوریتم RL به سادگی بازده نمونه را برابر با تابع امتیاز قرار می‌دهد:

$$f_{\pi_\theta}(.) = G_t \quad (۲-۲)$$

اصطلاح $b(s)$ از بازده نمونه کم می‌شود تا واریانس تخمین که معادله زیر را به روز می‌کند، کاهش دهد:

$$f_{\pi_\theta}(.) = G_t - b_t(s_t) \quad (۳-۲)$$

در حالی که از تابع ارزشQ استفاده می‌کنید، تابع امتیاز می‌تواند از گرادیان سیاست تصادفی یا گرادیان سیاست قطعی استفاده شده کند که در زیر داده شده است:

$$\nabla_\theta(\pi_\theta) = E_{s,a}[\nabla_\theta \log \pi_\theta(a|s).Q^\pi(s,a)] \quad (۴-۲)$$

و

$$\nabla_\theta(\mu_\theta) = E_s[\nabla_\theta \mu_\theta(s).Q^\mu(s,\mu_\theta(s))] \quad (۵-۲)$$

مشاهده شده است که این روش مطمئناً از نظر محدودیت‌های زمانی و مکانی محاسباتی بر روش قبلی غلبه دارد. ولی هنوز هم نمی‌توان آن را به کارهایی شامل تعامل با محیط‌هایی که به طور پیوسته در حال تکامل هستند و

[1] State-Action-Reward-State-Action

نیاز به عامل تطبیق دارند، تعمیم داد. توجه کنید که به دلیل مسائل ایمنی و محدودیت‌های سخت افزاری، استفاده از سیاست گرادیان در عمل مناسب نیست. بنابراین، بهینه سازی با استفاده از گرادیان سیاست در سیاست‌های تصادفی انجام می‌شود.

روش بازیگر منتقد: روش بازیگر منتقد: این الگوریتم ها نمایشی واضح از تخمین سیاست ها و حالت ها را نگه می دارند. تابع امتیاز برای این کار با جایگزینی بازده G_t از معادله روشهای مبتنی بر سیاست با $Q^{\pi_\theta}(s_t, a_t)$ و $b(s)$ پایه با $V^{\pi_\theta}(s_t)$ بدست می آید که منجر به معادله زیر می‌شود:

$$f_{\pi_\theta}(.) = Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t) \quad (۶-۲)$$

تابع مزیت $A(s,a)$ توسط رابطه زیر بدست می آید:

$$A(s,a) = Q(s,a) - V(s) \quad (۷-۲)$$

روش‌های بازیگر منتقد می‌تواند به عنوان تلاقی روش‌های مبتنی بر سیاست و روش‌های مبتنی بر ارزش توصیف شود، که روش‌های یادگیری تکراری هر دو روش را با هم ترکیب می‌کند.

یکپارچه سازی برنامه ریزی و یادگیری: روشهایی وجود دارد که در آن آنها عامل از تجربیات خود می آموزد و می‌تواند روال‌های خیالی را جمع آوری کند. چنین روشهایی با استفاده از روشهای DRL ارتقا یافته‌اند. این روش ها در گسترش استفاده از تکنیک‌های RL در سیستم‌های رباتیک نرم به کار رفته‌اند، زیرا وجود درجات آزادی زیاد در فضای حالت، منجر به تعامل هزینه بر با محیط و در نتیجه مصالحه در مورد داده‌های آموزش موجود می‌شود.

۲-۳-۴. یادگیری تقویتی عمیق و کاربرد آن در کپسول آندوسکوپی

این ایده که انسانها با تعامل با محیط یاد می‌گیرند، احتمالاً اولین ایده ای است که وقتی به ماهیت یادگیری می اندیشیم، به ذهن ما خطور می‌کند. با استفاده از این ارتباط، اطلاعات زیادی در مورد علت و معلول، در مورد عواقب اعمال و اینکه در جهت دستیابی به اهداف باید چه کاری انجام شود، تولید می‌شود [۱۳۹]. مانند انسانهایی که در حل طیف وسیعی از مشکلات چالش برانگیز از کنترل حرکتی سطح پایین تا کارهای شناختی سطح بالا با تعامل با محیط، سرآمد هستند، اخیراً عوامل مصنوعی می‌توانند در الگوی یادگیری تقویتی، دانش خود را مستقیماً از ورودی‌های خام با دریافت پاداش یا مجازات بسازند [۱۴۰]،[۱۴۱].

با انگیزه موفقیت تکنیک‌های یادگیری تقویتی عمیق در کنترل رباتیک، به ویژه روش‌های Q-Learning عمیق بازیگر منتقد [۱۴۱]، یک راه حل برای کنترل تجهیزات پزشکی پیچیده، به ویژه WCE ها استفاده از یادگیری تقویتی عمیق است. سیستم کنترل مبتنی بر یادگیری و مبتنی بر داده، به مدل سازی پیچیده مبتنی بر فیزیک سیستم نیاز ندارد. این کار، نیاز به دانش تخصصی و کار مهندسی جهت مدل سازی پیچیده حرکت در محیط‌های چالش

برانگیز مانند دستگاه GI با حرکات پریستالتیک را از بین می برد. همچنین سیستم کنترل پیشنهادی قادر به سازگاری با بیماران مختلف و اندامهای موجود در دستگاه گوارش است. شواهد تجربی از چنین سازگاری از طریق آزمایش‌های گسترده در چندین مورد معده گوشت خوک انجام شده است. این دو مورد مهم از بین مزایای کنترل کننده یادگیری تقویتی عمیق است.

ایده اصلی یادگیری تقویتی این است که یک عامل مصنوعی یاد می‌گیرد در تعامل با محیط و با بررسی کیفیت اقدام انجام شده از طریق اخذ نمرات پاداش دریافتی، چگونه رفتار خود را بهینه کند. این رویکرد اصولاً در مورد هر نوع مسئله تصمیم گیری متوالی با تکیه بر تجربه گذشته اعمال می‌شود. محیط ممکن است تصادفی باشد، نماینده فقط ممکن است اطلاعات جزئی در مورد وضعیت فعلی را مشاهده کند، مشاهدات ممکن است در ابعاد بالا باشد (به عنوان مثال، فریم ها و سری‌های زمانی)، عامل ممکن است آزادانه تجربیات خود را در محیط جمع کند یا برعکس، داده ها ممکن است محدود باشد (به عنوان مثال، عدم دسترسی به شبیه ساز دقیق یا داده‌های محدود به دلیل هزینه‌های زیاد یا حریم خصوصی داده ها و غیره). یادگیری تقویت عمیق (DRL) که توانایی استدلال یادگیری تقویت (RL) را با قدرت بازنمایی یادگیری عمیق ترکیب می‌کند، در سال‌های اخیر به عوامل بسیار موفقی منجر شده است که می‌توانند از پس مشکلات تصمیم گیری متوالی غیرخطی برآید. DRL به دلیل توانایی یادگیری سطوح مختلف انتزاع از داده ها، در مسائل فضای حالت با ابعاد بالا و کارهای پیچیده با دانش قبلی پایین، کاربرد دارد.

۲-۴. کنترل ربات به‌وسیله یادگیری تقویتی عمیق

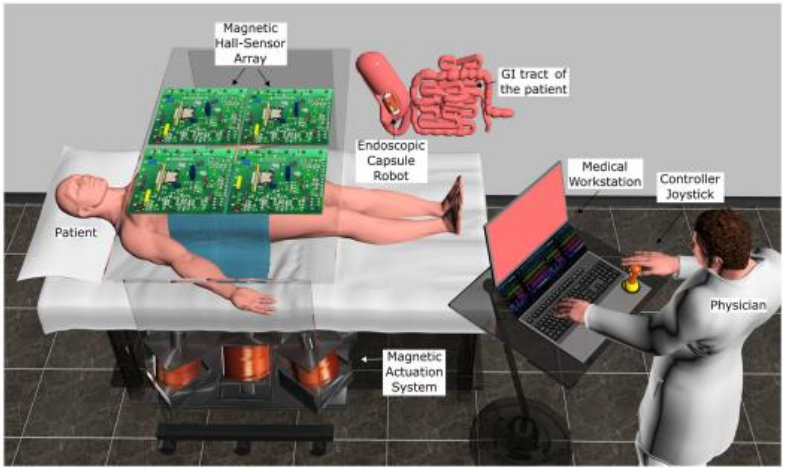
یک موقعیت صحیح و دقیق و کنترل جهت گیری WCE، پزشک را قادر می سازد ربات را به منطقه مربوطه با دقت میلی متر ببرد، داروها به مکان‌های خاص تحویل داده شود ویا داده‌های فیزیولوژیکی با وضوح مکانی بالا را اندازه گیری کند. علاوه بر این، بهره گیری از کنترل بهبود یافته، تصویربرداری تشخیصی را نیز ممکن می سازد. به عنوان مثال، WCEهای نسل دوم تقریباً در ۱۴٪ از بیماران قادر به تشخیص پولیپ[1] نیستند [۱۴۲]. به دست آوردن کنترل کامل بر وضعیت ربات کپسول ممکن است میزان خطا را بهبود بخشد، و مهمتر از همه، ربات‌های کپسول فعال می‌توانند چندین زاویه دید را تسهیل کنند، در صورت کشف مکان‌های مورد آسیب، میزان اطلاعات پزشک را بهبود می بخشند [۱۴۳].

۲-۴-۱. تحریک مغناطیسی

تحریک مغناطیسی از راه دور که نوعی از عملگر کپسول بی سیم خارجی است، یک مزیت منحصر به فرد دارد که نیازی به محرک‌های روی ربات ندارد. این امر پیچیدگی الکترومکانیکی دستگاه را بسیار کاهش می‌دهد

[1] polyp

و نیاز به انرژی و فضای اضافی را برطرف می‌کند. سناریوی کاربردی احتمالی کپسول آندوسکوپی قابل کنترل در بیمارستان ها را نشان می‌دهد، جایی که پزشک ربات را به مناطق مورد نظر جهت نظارت، تحویل دارو ویا عملیات شبیه بیوپسی می برد. آهنرباهای دائمی یا الکترومغناطیسی ممکن است در خارج از بدن بیمار قرار داشته باشد تا نیروی و گشتاور مطلوبی را به کپسولی که در داخل بدن قرار دارد تحمیل کند [۳۰].



شکل ۲-۱۴ سیستم تحریک مغناطیسی.

بسته به محیط، حرکت کپسول ممکن است فرم‌های مختلفی داشته باشد، از جمله کشیدن روی سطوح، حرکت طولی در مایعات یا چرخش به دور محور roll [۸۱]، [۳۲]. تکنیک‌های کنترلی که برای بستن حلقه به پزشک متکی هستند در بعضی موارد به دلیل جاذبه پایدار بین آهنربا خارجی و کپسول امکان پذیر است [۳۲]، اما برای دستیابی به ناوبری با دقت بالا یک کنترل کننده کاملا اتوماتیک لازم است. بستن حلقه نیاز به‌اندازه گیری دقیق وضعیت کپسول ها دارد و چندین روش غیر دیدی برای این منظور وجود دارد [۱۴۴]، [۷۸]. با این حال، دستکاری مغناطیسی حلقه بسته WCE ها همچنان چالش برانگیز است زیرا به طور معمول به مدل‌های دقیق میدان‌های مغناطیسی تولید شده توسط محرک ها نیاز دارد تا وابستگی موقعیت رباتها و جهت گیری آنها در این زمینه به طور مناسب پیش بینی شود [۳۲]. بعلاوه، این تکنیکها اغلب به کالیبراسیون مدلهای میدانی [۱۴۵] و مدلهای برهم کنش ربات و بافت نیاز دارند.

۲-۴-۲. انواع حرکت کپسول (پیچشی)

از نظر عملگر مغناطیسی برای ربات‌های کپسولی، در سال‌های اخیر روش‌های مختلفی ارائه شده است. محققان در مقاله [۳۶] از کویلهای مغناطیسی برای تحریک WCE ها در داخل معده پر آب برای اهداف غربالگری استفاده کردند که ۱۰ نوع حرکت اساسی را فراهم می‌کند. کلر و همکارانش کنترل حلقه بسته را ارائه نمی‌کند و رانش‌های بزرگی در قسمت‌های بالای معده دارد. علاوه بر این، از این روش کنترل نمی‌توان برای بیوپسی و تحویل دارو استفاده کرد زیرا به نظر نمی رسد بر اساس نتایج ارائه شده دقت کافی را ارائه دهد. جهت پیشران کپسول،

دو قطبی مغناطیسی چرخشی که می‌تواند حرکتی مانند پیچ برای WCE ایجاد کند، توسط پژوهش [۸۲] پیشنهاد شده است. اگرچه نشان داده شده است که چنین دستکاری مغناطیسی در سطوح روده مانند عملی است، اما عملیاتی بودن این روش در یک فضای بیولوژیکی متورم، مانند معده، نشان داده نشده است. روش مقاله [۱۴۳] علاوه بر کنترل موقعیت و جهت گیری در رزولوشن بالا از جهت محدودیت نداشتن ارجح است زیرا که روش پیشرانه ی پیچی به حرکت ۵ درجه آزادی محدود می‌شود.

۲-۴-۳. دلیل استفاده از یادگیری تقویتی

در صورت کنترل نیروهای مغناطیسی به صورت حلقه بسته، حرکت از طریق کنترل جهت می‌تواند به حرکت در سطوح جامد منجر شود. تکنیک‌های کنترل حلقه بسته که ممکن است مبتنی بر کنترل کننده‌های استاندارد مانند تناسبی-انتگرالی و تناسبی-انتگرالی-مشتقی باشد [۳۲]، [۱۴۶] بیش از حد به دقت مدل مورد استفاده برای توصیف تعامل بین ربات و میدان مغناطیسی خارجی بستگی دارد . ماهیت پیچیده و غیرخطی میدان‌های مغناطیسی خارجی تحمیل شده، بسته شدن حلقه را با چنین کنترل کننده‌های وابسته به مدل سنتی دشوار می‌کند. علاوه بر عدم قطعیت در میدان مغناطیسی، عدم قطعیت‌هایی در محیط کار وجود دارد که وظیفه کنترل را بیشتر پیچیده می‌کند.د خصوصیات مکانیکی و هندسه متفاوت از یک مکان به مکان دیگر، اختلالات ناشناخته ای نظیر پریستالسیس، فعل و انفعالات ربات و بافت، ناهنجاری‌های روی بافت (مانند تومورهای موجود) یا مواد مغناطیسی اطراف ممکن است تأثیر قابل توجهی بر دینامیک کلی سیستم داشته باشند.