مجله کنترل
ISSN (print) 2008-8345
ISSN (online) 2538-3752
جلد۱۴، شماره ۴، زمستان ۱۳۹۹، صفحه ۲۳-۱۳

افزایش سرعت فرایند یادگیری DQN با مکانیزم آثار شایستگی

سید علی خوشرو[۱]، سید حسین خواسته[۲]

[۱] فارغ‌التحصیل کارشناسی ارشد مهندسی کامپیوتر، گروه هوش مصنوعی، دانشگاه صنعتی خواجه نصیرالدین طوسی، Alikhoshroo@email.kntu.ac.ir

[۲] استادیار، دانشکدهٔ مهندسی کامپیوتر، گروه هوش مصنوعی، دانشگاه صنعتی خواجه نصیرالدین طوسی، Khasteh@kntu.ac.ir

دریافت: ۱۳۹۸/۰۲/۲۳ پذیرش: ۱۳۹۸/۱۰/۱۹

چکیده: برای سرعت بخشیدن به فرآیند یادگیری در مسائل یادگیری تقویتی با ابعاد بالا، معمولا از ترکیب روش‌های TD، مانند یادگیری Q یا سارسا، با مکانیزم آثار شایستگی، استفاده می‌شود. در الگوریتم شبکه عمیق Q (DQN)، که به تازگی معرفی شده، تلاش شده است که با استفاده از شبکه‌های عصبی عمیق در یادگیری Q، الگوریتم‌های یادگیری تقویتی را قادر سازد که به درک بالاتری از دنیای بصری رسیده و به مسائلی گسترش یابند که در گذشته رام‌نشدنی تلقی می‌شدند. DQN که یک الگوریتم یادگیری تقویتی عمیق خوانده می‌شود، از سرعت یادگیری پایینی برخوردار است. در این مقاله سعی می‌شود که از مکانیزم آثار شایستگی که یکی از روش‌های پایه‌ای در یادگیری تقویتی به حساب می‌آید، در یادگیری تقویتی در ترکیب با شبکه‌های عصبی عمیق استفاده شود تا سرعت فرایند یادگیری بهبود بخشیده شود. همچنین برای مقایسه کارایی با الگوریتم DQN، روی تعدادی از بازی‌های آتاری ۲۶۰۰، آزمایش انجام شد و نتایج تجربی به دست آمده در آنها نشان می‌دهند که روش ارائه شده، زمان یادگیری را در مقایسه با الگوریتم DQN، به طرز قابل توجهی کاهش داده و سریعتر به مدل مطلوب همگرا می‌شود.

کلمات کلیدی: شبکه‌های عصبی عمیق، Deep Q Network (DQN)، آثار شایستگی، یادگیری تقویتی عمیق.

Increase the speed of the DQN learning process with the Eligibility Traces

Seyed Ali Khoshroo, Seyed Hossein Khasteh

Abstract: To accelerate the learning process in high-dimensional learning problems, the combination of TD techniques, such as Q-learning or SARSA, is usually used with the mechanism of Eligibility Traces. In the newly introduced DQN algorithm, it has been attempted to using deep neural networks in Q learning, to enable reinforcement learning algorithms to reach a greater understanding of the visual world and to address issues Spread in the past that was considered unbreakable. DQN, which is called a deep reinforcement learning algorithm, has a low learning speed. In this paper, we try to use the mechanism of Eligibility Traces, which is one of the basic methods in reinforcement learning, in combination with deep neural networks to improve the learning process speed. Also, for comparing the efficiency with the DQN algorithm, a number of Atari 2600 games were tested and the experimental results obtained showed that the proposed method significantly reduced learning time compared to the DQN algorithm and converges faster to the optimal model.

Keywords: Deep Neural Networks, Deep Q Networks (DQN), Eligibility Traces, Deep Reinforcement Learning.

۱- مقدمه

یادگیری تقویتی [۱، ۲] چهارچوبی مناسب برای مسائلی است که نیاز به تصمیم‌گیری ترتیبی دارند، مانند مقاطعی که عامل ترتیبی از محیطش مشاهداتی را دریافت کرده و بر اساس آن تصمیم می‌گیرد. تا به امروز، روش‌های یادگیری تقویتی بسیاری توسعه پیدا کردند [۱] که از معروفترین و موفق‌ترین آنها در الگوریتم‌های تفاضل موقتی یادگیری تقویتی [۱]، یادگیری Q [۳] و سارسا [۴] می‌باشند. این روش‌ها در مسائل متنوعی از کنترل گرفته تا رباتیک [۵]، تخصیص منابع [۶] و پردازش ابری اعمال شده‌اند. اما بسیاری از مسائل دنیای واقعی، فضای حالت بزرگ با پاداش تاخیری دارند، مانند مسائل با ابعاد بالا و پاداش‌های پراکنده. برای این مسائل، ساختار ابتدایی این روش‌ها خیلی بهینه نبوده و حتی در صورت مفید بودن این الگوریتم‌ها، فرآیند یادگیری بسیار کندی داشته و نیازمند تعداد زیادی گام زمانی می‌باشند.

برای کار با مسائل یادگیری تقویتی که با ابعاد بالا سروکار دارند و برای تسریع فرایند یادگیری در آنها، راه‌حل‌های زیادی مانند یادگیری تقویتی سلسله‌مراتبی (HRL) [۷] و مکانیزم آثار شایستگی [۸] ارائه شدند. آثار شایستگی یکی از رایج‌ترین مکانیزم‌هایی است که در یادگیری تقویتی استفاده می‌شود. استفاده از این مکانیزم می‌تواند به شکل قابل ملاحظه‌ای سرعت یادگیری را افزایش دهد. برای رسیدن به این سرعت، یک روش TD پایه‌ای با آثار شایستگی ترکیب می‌شود که آن را TD(λ) [۸] می‌نامند. ترکیب این روش‌ها موجب ایجاد پلی بین الگوریتم‌های یادگیری TD و مونته‌کارلو شده و امکان بهره بردن از قدرت هر دو روش را فراهم می‌کند. پارامتر λ کنترل می‌کند که پشتیبان پس از چند گام گرفته شود. در واقع، مقدار λ برای آثار شایستگی، تعیین کننده‌ی تعادل بین TD و مونته‌کارلو می‌باشد.

اما چیزی که در سال های اخیر مشاهده شده است، ظهور یادگیری عمیق بوده که وابسته به ویژگی‌های قدرتمند تخمین‌گر تابع و یادگیری ویژگی شبکه‌های عصبی عمیق بوده و ابزاری جدید برای برخورد با این مسائل فراهم می‌آورد. ظهور یادگیری عمیق، تاثیر بسزایی در بسیاری از بخش‌های مختلف یادگیری ماشین داشته و آن‌ها را در کارهایی مانند تشخیص اشیاء، درک گفتار و ترجمه زبان-های مختلف به طور چشم‌گیری بهبود بخشیده است [۹]. شبکه‌های عصبی عمیق می توانند به طور خودکار یک ویژگی کم بعد از داده-های با ابعاد بالا (مانند تصویر، صدا و متن) بیابند که مهمترین ویژگی یادگیری عمیق است. با القا کردن بایاس درون ساختار شبکه عصبی، مخصوصا شبکه هایی که ویژگی سلسله مراتبی دارند، متخصصان یادگیری ماشین، پیشرفت موثری در برخورد با مشکل ابعاد بالا داشتند [۱۰]. یادگیری عمیق نیز به طور مشابه با استفاده از الگوریتم‌های خود در داخل RL، موجب تسریع در پیشرفت آن و ایجاد حوزه جدید یادگیری تقویتی عمیق (DRL) شده است.

تحقیقات اخیر روی یادگیری عمیق و یادگیری تقویتی، موجب معرفی روشی جدید به نام شبکه‌ی Q عمیق (DQN) [۱۱، ۱۲] شد که ترکیبی از الگوریتم‌های یادگیری Q [۱۳] و شبکه‌ی عصبی کانولوشنال [۱۴] (که نوعی از شبکه‌های عصبی عمیق می‌باشد) است. DQN در محیط بازی‌های آتاری ۲۶۰۰ آزمایش شد و در بسیاری از بازی‌ها، استراتژی آن از بازیکن انسانی بهتر عمل کرده و در چندین بازی با یک ساختار شبکه (فراپارامترها) به عالی‌ترین کارایی دست پیدا کرد. اما اعمال این روش به مسائل دنیای واقعی مانند رباتیک، بسیار چالش برانگیز است. چراکه اجرای تعداد زیادی قسمت‌های یادگیری برای جمع‌آوری نمونه، منابع زیادی مصرف کرده و در بسیاری موارد حتی غیر ممکن است. به همین علت، نیاز به ترکیبات دیگری از یادگیری تقویتی و شبکه‌های عصبی عمیق بود تا این مشکل را حل کنند.

یکی از مهمترین تعمیم‌های الگوریتم یادگیری Q ساده (تک‌گام)، یادگیری Q(λ) می‌باشد که ترکیبی از یادگیری Q و TD(λ) است [۳، ۱۵]. این الگوریتم در تعدادی از فعالیت‌ها، بسیار بهتر از یادگیری Q ابتدایی عمل می‌کند [۱، ۸]. علت آن است که مکانیزم آثار شایستگی، کارایی آن را با در نظر گرفتن تاریخچه تعاملات، حالت‌های مشاهده شده و عمل‌های انجام شده، تقویت می‌کند.

در این مقاله، روی همین ایده‌ی استفاده از مکانیزم آثار شایستگی کار می‌شود، به خصوص الگوریتم یادگیری Q(λ). این روش با استفاده از شبکه‌های عصبی عمیق به عنوان تخمین‌گر تابع (همانند DQN)، به ساختار جامع‌تری توسعه داده می‌شود. این شبکه‌ی عصبی عمیق، برای تخمین مقادیر Q استفاده می‌شود تا سرعت یادگیری افزایش یابد. الگوریتمی که در نهایت ارائه خواهد شد در تعدادی از بازی‌های آتاری ۲۶۰۰، مورد ارزیابی قرار می‌گیرد.

در ادامه مقاله به معرفی ساختار مسئله و پیش‌زمینه فنی یادگیری تقویتی و یادگیری Q عمیق پرداخته خواهد شد. سپس، الگوریتم پیشنهادی معرفی شده و نحوه عملکرد آن شرح داده خواهد شد. پس از آن نشان داده خواهد شد که روش ارائه شده در تعدادی از بازی‌های آتاری ۲۶۰۰ از DQN بهتر عمل می‌کند. بر اساس نتایج به دست آمده نیز نتیجه‌گیری نهایی انجام می‌شود.

۲- پیش زمینه

هدف عامل یادگیری تقویتی تخمین سیاست بهینه یا تابع ارزش بهینه یک فرایند تصمیم‌گیری مارکوف (MDP) در یک محیط ناشناس است. در صورت محدود بودن فضای حالت و عمل، مسئله یک MDP محدود خوانده می‌شود و در اینجا نیز همانند بسیاری از کارهای مشابه که محیط یک MDP محدود فرض می‌شود، مسئله در یک MDP محدود مورد بررسی قرار می‌گیرد.

مسئله RLی که به شکل یک فرایند تصمیم‌گیری مارکوف مدل می‌شود، به این صورت تعریف می‌شود: عامل یادگیرنده با محیط توسط حسگرهای خود تعامل داشته و با انجام عمل، مشاهدات و پاداش دریافت می‌کند. تعاملات تا زمانی که به حالت پایانی برسد یا یک شرط خاتمه مشاهده شود،

ادامه خواهد داشت. یک MDP یک پنج‌تایی (S,A,γ,T,R) است که پارامترهای آن به شرح زیر خواهند بود:

- S مجموعه‌ای از حالات در فضای حالت
- A مجموعه‌ای از عمل‌ها در فضای عمل
- $\gamma \leq 1 \geq 0$ نرخ نزول
- T تابع گذار (به این صورت که T(s,a,s') بیانگر احتمال رسیدن به حالت s' از حالت s با انجام عمل a است)
- R تابع پاداش (R(s,a) بیانگر مجموع پاداش مورد انتظار با انجام عمل a در حالت s در زمان t است)

هدف عامل یادگیرنده، یادگیری سیاست بهینه π است که احتمال انتخاب عمل a را در حالت s را تعیین کرده و با دنبال کردن آن، مجموعه پاداش نزولی دریافت شده در طول زمان بیشینه می‌شود. امید ریاضی پاداش بازگشتی نزولی R_t، در زمان t، به شکل معادله ۱ که در زیر آمده است، تعریف می‌شود.

$$R_t = E\{r_t, \gamma r_{t+1}, \gamma^2 r_{t+2} + \cdots\} = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k}] \qquad (۱)$$

که در آن $E[.]$ امید ریاضی با توجه به توزیع پاداش بوده و $r_t \in R$ پاداش اسکالر دریافت شده در گام t است. با در نظر داشتن تابع گذار و امید ریاضی نزولی پاداش لحظه‌ای، که عناصری ضروری برای معین کردن دینامیک یک MDP محدود هستند، تابع عمل-ارزش $Q_\pi(s,a)$ به این صورت (معادله ۲) تعریف می‌شود.

$$Q^\pi(s,a) = E_\pi[R_t | s_t = s, a_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s, a_t = a\right] \qquad (۲)$$

تابع ارزش-عمل $Q_\pi(s,a)$ برای یک عامل، امید ریاضی پاداش قابل دریافت از حالت s، $s \in S$، و اجرای عمل a، $a \in A$، و سپس دنبال کردن سیاست π است، که π نگاشتی از حالت‌ها به عمل‌ها یا توزیع روی عمل-هاست. به خاطر ویژگی بازگشتی معادله ۲، می‌تواند به شکل معادله ۳ بازنویسی شود:

$$Q_{i+1}^\pi(s,a) = E_\pi\left[r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right] = E_\pi[r_t + \gamma Q_i^\pi(s_{t+1} = s', a_{t+1} = a') | s_t = s, a_t = a] \qquad (۳)$$

که به عنوان قانون بروز رسانی تخمین تابع ارزش در تکرار iام استفاده می-شود.

سیاست بهینه π^*، سیاستی است که $Q_\pi(s,a)$ را بیشینه می‌کند و نتیجه‌ی آن تابع ارزش بهینه $Q^*(s,a)$ خواهد بود. معادله بروز رسانی تکراری برای تخمین تابع ارزش بهینه به شکل معادله ۴ تعریف می‌شود:

$$Q_{i+1}(s,a) = E_\pi[r_t + \gamma max_{a'} Q_i(s',a') | s,a] \qquad (۴)$$

واضح است که $s, s' \in S$ و $a, a' \in A$ می‌باشند. این تکرار در صورتی که $I \to \infty$ به تابع ارزش بهینه Q^* همگرا شده و الگوریتم تکرار ارزش خوانده می‌شود.

یک شکل معروف یادگیری تفاضل موقتی [۸] برای تخمین Q_π یک سیاست داده شده، الگوریتم یادگیری Q است که توسط واتکینس [۳] معرفی شد. در بسیاری از فعالیت‌ها در دنیای واقعی، فضاهای حالت و عمل بسیار بزرگ بوده و استفاده از یک جدول برای کل مقادیر $Q(s,a)$ ناکارآمد است. برای حل این مشکل از تکنیک تخمین تابع استفاده می‌شود که تابع ارزش را تخمین می‌زند [۱۶]. بنابراین تابع ارزش به شکل $Q(s,a;w)$ با بردار پارامتر w پارامتری سازی می‌شود. معمولا روش-های گرادیان نزولی، برای یادگیری پارامترها با تلاش برای کمینه کردن تابع زیان خطای میانگین مربعات که در معادله ۵ آمده است، در مقادیر Q استفاده می‌شوند:

$$L(w) = E[(r + \gamma max_{a'} Q(s',a';w) - Q(s,a;w))^2] \qquad (۵)$$

که $r + \gamma max_{a'} Q(s',a';w)$ مقدار هدف می‌باشد. معمولا برای بهینه‌سازی تابع زیان بالا از روش گرادیان نزولی تصادفی استفاده می‌شود. به همین خاطر در الگوریتم یادگیری Q، پارامترها به شکل معادله ۶، بروز رسانی می‌شوند:

$$w_i = w_{i-1} + \alpha(y_i - Q(s,a;w_i)) \frac{\partial Q(s,a;w_i)}{\partial w_i} \qquad (۶)$$

مشخص است که $y_i = r + \gamma max_{a'} Q(s',a';w_{i-1})$ مقدار هدف برای تکرار i بوده و α نرخ یادگیری اسکالر است.

برای افزایش سرعت فرآیند یادگیری در یادگیری تقویتی، روش‌های TD(λ) (یادگیری TD با مکانیزم آثار شایستگی) [۸] با الگوریتم یادگیری Q ترکیب می‌شود که نتیجه‌ی آن یادگیری Q(λ) خواهد بود. آثار شایستگی، یک تاریخچه‌ی موقت از مجموعه‌ی گذارها مانند حالت‌های مشاهده شده و اعمال انتخاب شده را در نظر می‌گیرد. در TD(λ) ساخت پشتیبان پس از هر گام انجام نشده و پس از مقدار مشخص n گام تهیه می‌شود. مقدار n توسط پارامتر $\lambda \in [0,1]$ کنترل می‌شود (برای مثال در TD(0) تهیه پشتیبان بعد از هر گام انجام می‌شود). مقدار اثر شایستگی، برای هر جفت حالت-عمل در فرآیند یادگیری عمل-ارزش، پس از بر خورد با جفت حالت-عمل مربوطه بزرگتر شده و در صورت برخورد نداشتن با آن کوچکتر می‌شود. هنگامی که از تخمین‌گر تابع برای تخمین مقادیر Q به جای جدول Q استفاده می‌شود، یک اثر برای هر جزء بردار پارامتر w در نظر گرفته می‌شود و برای یک حالت، تنها یک اثر نیست که به آن مربوط می‌شود. بنابراین TD(λ) بردار w را به صورت معادله ۷، بروزرسانی می‌کند:

$$w_i = w_{i-1} + \alpha \delta_i e_i \qquad (۷)$$

که در آن $\delta_i = y_i - Q(s,a;w_i)$ خطای TD بوده و $e_i = \gamma\lambda e_{i-1} + \frac{\partial Q(s,a;w_i)}{\partial w_i}$ مقدار اثر شایستگی آن می‌باشد. باید توجه داشت که در صورتی که $\lambda = 0$ باشد، بروز رسانی TD(λ) به بروز رسانی TD(0) تبدیل می‌شود.

دو رویکرد اصلی در ترکیب آثار شایستگی، با یادگیری Q وجود دارند که تفاوت آن‌ها در برخورد با بحث اکتشاف در اعمال (غیر حریصانه) است: نخست Q(λ) واتکینس [۳] است که بیان می‌کند هرجا که عملی غیر حریصانه انجام شود، مقدار اثر شایستگی به صفر تغییر می‌یابد (یادگیری، پس از هر انتخاب عمل غیر حریصانه، متوقف می‌شود)، و دوم Q(λ) پنگ [۱۵] است که تفاوتی بین اعمال حریصانه و غیر حریصانه قایل نمی‌شود.

شبکه‌ی یادگیری Q عمیق [۱۱، ۱۲] از مزایای ساخت ویژگی مجرد یادگیری عمیق، در یادگیری سیاست بهینه بهره‌مند است. الگوریتم DQN یک شبکه‌ی عصبی عمیق به عنوان تخمین‌گر تابع در خود جای داده و برای یک حالت داده شده، مقدار ارزش عمل آن را خروجی می‌دهد. استفاده از الگوریتم‌های یادگیری تقویتی بدون مدل، مثل الگوریتم یادگیری Q با یک تخمین‌گر تابع غیر خطی، ناپایدار بوده و ممکن است موجب واگرایی [۱۷] شود. دلایلی که باعث بوجود آمدن این مشکلات می‌شوند به این قرار می‌باشند:

۱) حالت‌های متوالی در مسائل یادگیری تقویتی عمیق به هم مرتبط هستند.
۲) سیاست عامل به علت تغییرات کوچک در مقادیر Q همواره در حال تغییر است.

DQN برای مقابله با این مشکلات، راه‌حل‌هایی ارائه می‌دهد که باعث می‌شوند کارایی الگوریتم به طرز قابل توجهی افزایش یابد. برای مشکل حالت‌های مرتبط، DQN از تکرار تجربه [۱۸] استفاده می‌کند. در این روش، در هر گام زمانی، DQN تجربه عامل (s_t, a_t, r_t, s_{t+1}) را در یک پایگاه داده ذخیره می‌کند که در آن s_t، a_t و r_t به ترتیب حالت، عمل انتخاب شده و پاداش دریافت شده می‌باشند و s_{t+1} حالت در گام بعدی است. برای بروزرسانی شبکه، DQN در زمان یادگیری از بروزرسانی‌های دسته‌ای تصادفی استفاده می‌کند که به طور تصادفی یکنواخت، از حافظه تکرار تجارب (گذارهای قبلا مشاهده شده) نمونه‌برداری می‌شوند. این کار موجب کاهش تاثیر ارتباط قوی بین نمونه‌های متوالی می‌شود. مشکل ناپایداری سیاست با شبکه‌ی Q هدف حل می‌شود. شبکه به وسیله‌ی شبکه Q هدف برای به دست آوردن اهداف یادگیری Q مناسب، آموزش می‌بیند به این صورت که پارامترهای وزن w^- استفاده شده در یادگیری Q هدف، ثابت نگه داشته شده و به صورت دوره‌ای در هر N گام با پارامترهای شبکه اصلی w بروزرسانی می‌شوند. مقادیر هدف DQN به صورت معادله ۸ نمایش داده می‌شوند:

$$y_i = r + \gamma max_{a'} Q(s', a'; w_{i-1}^-) \quad (۸)$$

که در آن w^-، پارامتر شبکه هدف است.

۳- کارهای مرتبط

از زمان ارائه DQN، تلاش‌های بسیاری جهت تقویت سرعت و پایداری آن انجام شد. از بین روش‌های ارائه شده برای بهبود این الگوریتم، می‌توان مواردی همچون یادگیری Q دوگانه، تکرار تجربه با اولویت، شبکه‌های رقابتی، یادگیری از هدف‌های چندگام، یادگیری Q توزیعی و DQN نویزی و ... را برشمرد.

یادگیری Q مرسوم، به علت وجود گام بیشینه سازی در معادله‌ی خود با یک بایاس بیش‌برآورد[1] روبروست که می‌تواند برای فرایند یادگیری مشکل ایجاد کند. در مقاله ارائه شده توسط Hasselt و همکاران [۱۹] نشان داده شده است که ایده الگوریتم یادگیری Q دوگانه، که ابتدا در قالب جدولی ارائه شد، می‌تواند به تخمین‌گرهای تابع دلخواه تعمیم یابد که شامل شبکه عصبی عمیق نیز می‌شود. با استفاده از این ایده، الگوریتمی جدید به نام DQN دوگانه[2] ساخته شد که نه تنها تخمین‌های دقیقتری محاسبه می‌کند، بلکه امتیازات بالاتری نیز در چندین بازی به دست آورد. یادگیری Q دوگانه [۱۹]، این بایاس را با جداسازی انتخاب یک عمل از ارزیابی آن از بین می‌برد. این تغییر نشان داده شد که می‌تواند بیش‌برآوردهای زیان‌آوری را که در DQN وجود داشتند، کاهش داده و در نتیجه کارایی را افزایش دهد.

در مقاله ارائه شده توسط Schaul و همکاران [۲۰] به بررسی این ایده پرداخته شده است که اولویت بندی اینکه تکرار چه تجربه‌ای نسبت به تکرار یکنواخت، موجب بهینه‌تر و موثرتر شدن تکرار تجربه می‌شود. ایده کلیدی این بود که یک عامل RL بهتر می‌تواند از برخی گذارها نسبت به دیگر گذارها یادگیری انجام دهد. تکرار تجربه، عامل‌های یادگیرنده آنلاین را از پردازش تجربه‌ها به ترتیبی که روی دادند، آزاد می‌کند. اما تکرار تجربه با اولویت[3] [۲۰]، عامل‌ها را از اینکه تجربه‌ها را با همان نرخی که روی دادند، ببینند، رهایی می‌بخشد.

به طور دقیق‌تر، این ایده ارائه شد که تجربه‌هایی بیشتر تکرار شوند که پیشرفت یادگیری آنها، که توسط اندازه خطای TDشان محاسبه می‌شود، بیشتر باشد. برای انجام این کار، تجربه‌ها با احتمال p_t که با آخرین خطای مطلق TD مشاهده شده نسبت دارند، انتخاب می‌شوند. البته باید توجه شود که گذارهای تصادفی نیز امکان دارد که ترجیح داده شوند، حتی اگر چیز زیادی برای یادگیری نداشته باشند. الگوریتمی که در نهایت به دست آمد، مقیاس‌پذیر بوده و نشان داده شد که در بازی‌های آتاری ۲۶۰۰، یادگیری سریعتری انجام داده و کارایی بهتری دارد.

در کاری که توسط Wang و همکاران [۲۱] ارائه شده است، تلاش شده است بجای استفاده از شبکه‌های عصبی استاندارد مانند شبکه‌های کانولوشنال، پرسپترون‌های چندلایه (MLP)، حافظه‌های طولانی کوتاه‌مدت (LSTM) و اتوانکودرها در RL، رویکردی تکمیلی در طراحی

[1] Overestimation
[2] Double Deep Q Network
[3] Prioritized Experience replay

شبکه‌ای در پیش گرفته شود که برای RL بدون مدل، مناسب‌تر باشد. معماری شبکه ارائه شده که معماری رقابتی[1] نامیده شد، نمایش ارزش حالت را از مزیت عمل [۲۲، ۲۳] جدا می‌کند. این معماری از دو جریان که نماینده توابع ارزش و مزیت هستند، تشکیل می‌شود که یک ماژول یادگیری ویژگی کانولوشنال را به اشتراک می‌گذارند. این دو جریان با یک لایه تجمیع خاص ترکیب می‌شوند تا تخمینی از تابع حالت–عمل Q تولید کنند.

در مقاله ارائه شده توسط Hessel و همکاران [۲۴] مطالعه‌ای انجام شد که سعی می‌کرد با ترکیب تمام بهبودهایی که تا به آن روز روی الگوریتم DQN داده‌شده بود به عاملی پایدارتر و قدرت بیشتر دست پیدا کند. از آنجایی که این بهبودها روی قسمت‌های مختلفی انجام می‌گیرند و روی یک چهارچوب پیاده می‌شوند، نشان داده شد که این امکان وجود دارد که بتوانند با هم ترکیب شوند. در نهایت، الگوریتم مرکب به دست آمده که رنگین‌کمان[2] خوانده شد، بهترین نتایج را هم از نظر بهینگی استفاده از داده و هم کارایی، در محیط آتاری ۲۶۰۰ در بر داشت.

یکی از فراپارامترهایی که در چهارچوب DQN از آن استفاده شد، نرخ پرش فریم[3] است که بیانگر تعداد دفعاتی است که عمل انتخاب شده توسط عامل، باید تکرار شود. اگر این نرخ پایین باشد، نرخ انتخاب عمل عامل بیشتر می‌شود، که نتیجه‌ی آن سیاست‌هایی است که نسبت به زمان و فضای حالت با نوسان بالایی تغییر می‌کنند. سیاست‌هایی که با نرخ پرش فریم بالا آموزش دیده‌اند، دنباله‌ی عمل کمتری خواهند داشت که نشان دهنده‌ی عکس‌العمل‌های فرا–انسانی در مقایسه با پرش فریم پایین DQN (که از نرخ پرش ۴ استفاده می‌کند) است.

در مقاله ارائه شده توسط Braylan و همکاران [۲۵] نشان داده شد که عامل‌هایی که با پرش‌های فریم بالا کار می‌کنند، می‌توانند به نسبت تعداد قسمت‌های آموزشی، از آنهایی که هیچ پرش فریمی ندارند، یادگیری سریعتری داشته باشند. همچنین به طور تجربی نشان دادند که پارامتر پرش فریم، فاکتوری مهم در تعیین کارایی عاملشان در حوزه آتاری ۲۶۰۰ است. برای مثال نشان داده شد که استفاده از نرخ ۱۸۰ برای پرش فریم در بازی سیکوئست بهترین نتیجه را به دست خواهد آورد.

در مقاله ارائه شده توسط Lakshminarayanan و همکاران [۲۶] اولین گام در جهت تجریدهای زمانی در فضای سیاست، با بررسی یک مدل پرش فریم پویا که روی معماری DQN پیاده‌سازی می‌شود، برداشته شده است. در این مدل که DQN با پرش فریم پویا (DFDQN) خوانده شد، عامل می‌تواند برای بازی از بین عمل‌های ماکرو که می‌توانند دو مقدار متفاوت (اما ثابت) نرخ پرش فریم که در هر حالت بازی موجود هستند، برای بازی انتخاب کند. آنها نشان دادند که سیاست‌های یاد گرفته شده توسط همچین مدل‌هایی، نسبت به آنهایی که توسط DQN آموخته شدند (با نرخ فریم ثابت)، تاثیرگذارتر می‌باشند. نتایج آزمایشات تجربی آنها به این حقیقت رسید که تکرار عمل به صورت پویا، راهی مهم برای توسعه قابلیت‌های عامل‌های هوش مصنوعی است.

٤- روش پیشنهادی

با وجود انواع مختلف یادگیری Q(λ) مانند پنگ برای ترکیب با یادگیری عمیق، ما در اینجا یک نوع ساده از واتکینس را در نظر می‌گیریم. این نوع ساده، شبیه Q(λ) واتکینس بوده، اما اثر شایستگی در عمل‌های غیر حریصانه صفر نمی‌شوند. با توجه به TD(λ) قانون بروزرسانی برای بردار w در الگوریتم ارائه شده به شکل معادله ۹ می‌باشد:

$$w_i = w_{i-1} + \alpha\delta_t E_t \qquad (۹)$$

که در آن:

$$E_i = \gamma\lambda E_{i-1} + \frac{\partial Q(s,a;\, w_i)}{\partial w_i} \qquad (۱۰)$$

و

$$\delta_i = y_i - Q(s,a;\, w_i) \qquad (۱۱)$$

و در آن $y_i = r + \gamma max_{a'} Q(s',a';w_{i-1}^{-})$ مقدار هدف است که همانند DQN محاسبه می‌شود.

با مقایسه معادله ۹ با معادله ۷، تفاوت کلیدی بین این روش و DQN مشخص می‌شود. این دو رویکرد با استفاده از شبکه‌ی هدف و وزن‌های w_{i-1}^{-} در محاسبه مقدار هدف با هم شباهت دارند. شبکه‌ی هدف به طور دوره‌ای بر پایه‌ی شبکه‌ی اصلی بروزرسانی می‌شود. برای جلوگیری از واگرایی در پارامترها، مکانیزم تکرار تجربه اعمال می‌شود [۱۲].

Algorithm. Deep Q Network with Eligibility Traces

initialize $\boldsymbol{w}$ with random values
initialize replay memory $\boldsymbol{M}$ with capacity N
for each episode repeat:
  initialize $\boldsymbol{E} = \mathbf{0}$
  initialize $\boldsymbol{s}$
  for each step in episode repeat:
    choose action $\boldsymbol{a}$ according to $\boldsymbol{\epsilon}$-greedy policy
    take action $\boldsymbol{a}$, observe reward $\boldsymbol{r}$ and next state $\boldsymbol{s}'$
    store transition $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$ in $\boldsymbol{M}$
    $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
    $\boldsymbol{b} \leftarrow$ sample a sequence of transitions from replay memory, $\boldsymbol{M}$
    if $\boldsymbol{s_b}$ (as the last state in the sequence) == terminal:
      $\boldsymbol{y} \leftarrow \mathbf{0}$
    else:
      $\boldsymbol{y} \leftarrow \boldsymbol{max_a Q(s_b, a;\, w^{-})}$
    for each transition $(\boldsymbol{s_j, a_j, r_j, s'_j})$ in reverse(b) repeat:
      $\boldsymbol{y} \leftarrow \boldsymbol{r_j + \gamma y}$
      $\boldsymbol{E} \leftarrow \boldsymbol{\gamma\lambda E + \frac{\partial Q(s_j,a_j;w)}{\partial w}}$
      $\boldsymbol{\delta} \leftarrow \boldsymbol{y - Q(s_j, a_j; w)}$
      $\boldsymbol{w} \leftarrow \boldsymbol{w + \alpha\delta E}$
  until $\boldsymbol{s}$ is terminal

شکل ۱: شبه کد الگوریتم ارائه شده

[1] Dueling network architecture
[2] Rainbow
[3] Frame skipping

شبه کد الگوریتم روش ارائه شده در شکل ۱ خلاصه می‌کند که در آن بردار e شامل بردار اثر برای هر جزء بردار پارامتر w مربوط به آثار شایستگی می‌شود. برای $\lambda = 0$ الگوریتم مشابه DQN عمل خواهد کرد.

۱-۴- پیاده سازی

کار مستقیما با فریم‌های خام آتاری که تصاویر ۲۱۰ * ۱۶۰ پیکسلی در ۱۲۸ رنگ می‌باشند، از لحاظ محاسباتی و حافظه‌ای، بسیار سنگین خواهند بود، به همین دلیل، یک پیش‌پردازش ساده انجام می‌گیرد تا ابعاد ورودی کاهش یافته و برنامه با شبیه‌ساز آتاری ۲۶۰۰ بهتر کار کند. برای این کار، فریم‌های خام ابتدا از قالب RGB به سیاه و سفید تبدیل شده و سپس به ابعاد ۱۱۰ * ۸۴ نمونه‌برداری کاهشی می‌شوند. در نهایت ورودی با استخراج یک ناحیه ۸۴ * ۸۴ از تصویر که با تقریب خوبی ناحیه بازی را شامل می‌شود، به دست می‌آید. برش نهایی انجام شده به علت پیاده‌سازی کانولوشن 2D برای GPU بوده که نیاز به ورودی مربعی دارد.

روش‌های مختلفی برای پارامترسازی Q در شبکه‌های عصبی وجود دارد. از آنجایی که Q جفت تاریخچه-عمل را به مقادیر Q تخمین زده شده آنها نگاشت می‌کند، رویکردهای گذشته، تاریخچه و عمل را به عنوان ورودی شبکه عصبی استفاده کرده‌اند [۲۷، ۲۸]. اشکال اصلی این نوع معماری، نیاز به یک گذر جداگانه برای محاسبه مقدار Q برای هر عمل می‌باشد، که هزینه‌ای را نتیجه خواهد داد که با تعداد عمل‌ها به صورت خطی رشد می‌کند. به جای این کار از معماری استفاده می‌کنیم که در آن برای هر عمل ممکن، یک واحد خروجی جداگانه خواهد داشت و حالت تنها ورودی شبکه عصبی خواهد بود. خروجی‌ها، مرتبط با مقادیر Q پیش‌بینی شده برای هر عمل در حالت ورودی خواهند بود. مزیت اصلی این معماری، قابلیت محاسبه مقادیر Q برای تمام عمل‌های ممکن در حالت ورودی داده شده، تنها با یک گذر در شبکه می‌باشد.

ورودی شبکه عصبی که برای آزمایش در بازی‌ها استفاده می‌شود، شامل یک تصویر ۸۴ * ۸۴ * ۴ که در مرحله پیش‌پردازش تولید شده، می‌باشد. اولین لایه مخفی شامل ۳۲ فیلتر ۸ * ۸ با گام[1] ۴ بوده که تصویر ورودی را دریافت کرده و تابع فعال‌سازی غیرخطی یکسوساز را به آن اعمال می‌کند. لایه مخفی دوم از ۶۴ فیلتر ۴ * ۴ با گام ۲ تشکیل شده است که مانند لایه قبل، یک تابع فعال‌سازی غیرخطی یکسوساز را به دنبال خواهد داشت. سومین لایه مخفی، ۶۴ فیلتر ۳ * ۳ با گام ۱ را شامل می‌شود که از یک یکسوکننده به عنوان تابع فعال‌ساز بهره خواهد برد. لایه مخفی پایانی که کاملا متصل است، از ۵۱۲ واحد یکسوساز تشکیل می‌شود. لایه خروجی، یک لایه کاملا متصل خطی است که یک یک خروجی برای هر عمل، که بین ۴ تا ۱۸ برای بازی‌های مختلف متغیر می‌باشد، به همراه خواهد داشت.

پیاده‌سازی سیستم‌های بزرگی مانند DQN، معمولا به علت اینکه مقاله اصلی، به جزئیات مهم تنظیمات پارامترها و راه‌حل مهندسی نرم‌افزار اشاره نمی‌کنند، دشوار خواهند بود. در مقاله DQN [۱۲] نیز، برخی جزئیات مهم سطح پایین الگوریتم، به طور واضح و مشخص بیان نشده‌اند که برای به دست آوردن این جزئیات کلیدی، نیاز به بررسی کد اصلی منتشر شده می‌باشد[۲۹].

هر قسمت، با تعدادی تصادفی (بین ۰ تا ۳۰) از عمل سطح پایین no-op آتاری شروع می‌شود (بر خلاف عمل‌های عامل که در چهار فریم تکرار می‌شوند) تا تعادلی بین فریم‌هایی که عامل می‌بینید برقرار شود، چراکه عامل هر چهار فریم آتاری را خواهد دید. به طور مشابه، تاریخچه m فریمی که به عنوان ورودی CNN استفاده می‌شود، آخرین m فریمی است که عامل می‌بیند نه آخرین m فریم آتاری. همچنین قبل از هر گام گرادیان نزولی، یک سیاست تصادفی برای ۵۰۰۰۰ گام اجرا می‌شود تا با به دست آوردن تجاربی کافی، از بیش‌برازش روی تجارب اولیه جلوگیری شود.

پارامتر دیگری که باید مورد توجه قرار بگیرد، فرکانس بروزرسانی شبکه است، پیاده‌سازی اصلی DQN به جای اینکه گرادیان نزولی را در هر گام محیط، همانطور که در الگوریتم بیان شده‌است انجام دهد، هر چهار گام این بروزرسانی را انجام می‌داد. این کار نه تنها سرعت آموزش را بسیار افزایش می‌دهد (به علت اینکه گام‌های یادگیری روی شبکه بسیار هزینه‌برتر از گذرهای رو به جلو هستند)، بلکه موجب می‌شود حافظه تجارب، بیشتر به توزیع حالت‌های سیاست فعلی شباهت داشته باشد (چراکه چهار فریم جدید بین گام‌های یادگیری به جای یکی، به حافظه اضافه می‌شوند) و ممکن است از بیش‌برازش شبکه جلوگیری کند.

در بسیاری از بازی‌های آتاری، مولفه‌ای به نام فرصت[2] برای بازیکن وجود دارد که بیانگر تعداد دفعاتی است که بازیکن می‌تواند، بدون پایان یافتن بازی، شکست بخورد. برای افزایش کارایی، [۱۲] این از دست دادن فرصت‌ها را به عنوان حالت پایانی MDP در طول آموزش در نظر گرفت که برای رسیدن به کارایی آنها لازم است، اما جزئیات آن در مقاله شرح داده نشد. این اطلاعات به ما کمک کرد تا از مزایای آموزش‌های اولیه و پایداری بهره برده و در بازی‌های پیچیده‌تر، کارایی کلی را افزایش دهیم.

حالت پایانی در یک MDP، برای عامل مشخص می‌کند که پاداش بیشتری قابل دریافت نیست. تقریبا تمام بازی‌های آتاری پاداش مثبت برمی‌گردانند و این دانش به عامل اطلاع می‌دهد که از دست دادن یک فرصت باید به هر قیمتی اجتناب شود. دادن این اطلاعات اضافی به عامل منطقی به نظر می‌رسد چراکه بسیاری از بازیکنان انسانی از بد بودن از دست دادن فرصت در همان ابتدای بازی کردن، اطلاع دارند.

اما به هر حال، چند مسئله تئوری در اجبار این محدودیت وجود دارند. اولین آن، مارکوفی نبودن این فرایند به خاطر وابستگی توزیع اولیه حالت به سیاست فعلی است. برای مثال اگر عامل در بازی اسپیس اینویدرز[3]، خوب عمل کرده و چندین دشمن را قبل از کشته شدن، از بین ببرد، حالت ابتدایی جدید برای فرصت بعدی، دشمنان بسیار کمتری نسبت به حالت ابتدایی قبلی خواهد داشت. مسئله بعدی، اطلاعات اضافی مهمی است که این

[1] Stride
[2] Lives
[3] Space invaders

سیگنال به شبکه می‌دهد و تعمیم آن را به دامنه‌هایی که این سیگنال قدرتمند وجود ندارد، بسیار چالش برانگیز می‌کند (مانند رباتیک در دنیای واقعی یا بازی‌های ویدئویی بدون انتها).

یکی از الگوریتمهای استفاده شده در پیاده سازی روش پیشنهادی که در پیاده سازی الگوریتم اصلی DQN نیز استفاده شده است، الگوریتم بهینه سازی RMSProp [۳۰] است، RMSProp یک الگوریتم بهینه‌سازی است که برای بروزرسانی تمامی وزن‌های شبکه عصبی از آن استفاده شده است. یکی از مشکلاتی که در استفاده از فراپارامترهای ارائه شده توسط [۱۲] وجود دارد، استفاده نکردن آنها از تعریف از RMSProp است که کتابخانه‌های مختلف یادگیری عمیق ارائه می‌دهند. الگوریتم بهینه‌سازی گرادیان نزولی RMSProp، توسط جفری هینتون معرفی شد. RMSProp هینتون، یک میانگین از گرادیان برای هر پارامتر نگه می‌دارد. قانون بروزرسانی برای این میانگین به شکل معادله ۱۲ نوشته می‌شود:

$$MeanSquare(w,t) = \gamma MeanSquare(w,t-1) + (1-\gamma)(\frac{\partial E}{\partial w}(t))^2 \quad (۱۲)$$

که در آن w، تک پارامتری از شبکه، γ فاکتور نزولی و E خطا می‌باشد. پارامترها سپس به شکل معادله ۱۳ بروزرسانی می‌شوند:

$$w_t = w_{t-1} - \frac{\alpha}{\sqrt{MeanSquare(w,t)+\varepsilon}}\frac{\partial E}{\partial w}(t) \quad (۱۳)$$

که α، نرخ یادگیری و ε، ثابتی کوچک برای جلوگیری از تقسیم بر صفر هستند.

اگرچه [۱۲] به RMSProp هینتون ارجاع می‌دهد، آن را با یک تغییر کوچک در الگوریتم استفاده می‌کنند. پیاده‌سازی آن که در کد اصلی قابل مشاهده است، یک فاکتور مومنتوم به الگوریتم RMSProp اضافه می‌کند که به صورت معادله ۱۴ بروزرسانی می‌شود.

$$Momentum(w,t) = \eta Momentum(w,t-1) + (1-\eta)\frac{\partial E}{\partial w}(t) \quad (۱۴)$$

در این معادله، η، فاکتور نزولی مومنتوم است. به این ترتیب قانون بروزرسانی پارامتر به شکل معادله ۱۵ تغییر می‌یابند.

$$w_t = w_{t-1} - \frac{\alpha}{\sqrt{MeanSquare(w,t)-(MeanSquare(w,t))^2+\varepsilon}}\frac{\partial E}{\partial w}(t) \quad (۱۵)$$

سخت‌افزاری که تست‌های نهایی روی آن انجام شد، از یک کارت گرافیک GTX 1080 و پردازنده اینتل i7 تشکیل شده است.

در پیاده‌سازی، از کتابخانه دیپ‌لرنینگ۴جی[1] [۳۱] کمک گرفته شده‌است. این کتابخانه تعریف ساختار MDP را آسان کرده و بسیاری از الگوریتم‌های یادگیری و برنامه‌ریزی مهم را فراهم آورده و اجازه ساخت الگوریتم‌های جدیدتر را نیز می‌دهد.

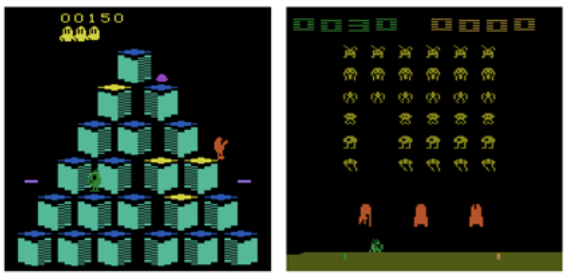
کتابخانه شبکه عصبی عمیق انویدیا کودا (cuDNN) یک کتابخانه اختصاصی انویدیا برای گذرهای رو به جلو و عقب در لایه‌های پرکاربرد شبکه‌های عصبی است که به طور خاص برای کارت‌های گرافیک انویدیا بهینه‌سازی شده است.

cuDNN برای کتابخانه دیپ‌لرنینگ۴جی در دسترس می‌باشد، اما به صورت پیش‌فرض استفاده نمی‌شود. پس از تنظیمات کوچکی در کتابخانه دیپ‌لرنینگ۴جی، می‌توان محاسبات پس‌زمینه جبری را به راحتی با استفاده از cuDNN به کارت گرافیک منتقل کرد.

روش دیگری که موجب افزایش سرعت پیاده‌سازی می‌شد، پیش‌تخصیص حافظه قبل از آموزش بود که در کد اصلی DQN نیز انجام داده شده بود. تخصیص حافظه‌ای بزرگ، عملیات پرهزینه‌ای است، به همین علت، پیش‌تخصیص حافظه برای بردارهای بزرگ، مانند حافظه تجربه و گروه‌های کوچک داده و استفاده مجدد از آنها در هر تکرار، این هزینه زمانی را بسیار کاهش می‌دهد.

۵- ارزیابی

در این قسمت با ارائه نتایج کارایی الگوریتم ارائه شده، نشان داده می‌شود که در سرعت یادگیری از DQN بهتر عمل می‌کند. روش ارائه شده با دو بازی آتاری ۲۶۰۰ در ALE[2] [۳۲] ارزیابی می‌شود. ALE محیط شبیه‌سازی شده‌ی بازی‌های آتاری ۲۶۰۰ را فراهم می‌آورد. این محیط برای یادگیری تقویتی بسیار چالش برانگیز است چراکه ورودی دیداری با ابعاد بالایی داشته و به طور کامل نیز قابل مشاهده نیست. با داشتن یک سری بازی‌های جذاب، محیط مناسبی برای آزمایش روش‌های جدید می‌باشد. برای این آزمایش دو بازی آتاری کیوبرت[3] و اسپیس اینویدرز همانطور که در شکل ۲ قابل مشاهده هستند، انتخاب شدند. هدف الگوریتم RL، یادگیری یک سیاست بهینه خاص، تنها با استفاده از فریم‌های پیکسلی خام به عنوان ورودی، برای بازی در هر یک از این بازی‌هاست.



شکل ۲: دو فریم از بازی‌های آتاری ۲۶۰۰: کیوبرت (سمت چپ) و اسپیس اینویدرز (سمت راست)

معماری شبکه استفاده شده همانند [۱۲] می‌باشد که از سه لایه‌ی مخفی کانولوشنال و یک لایه‌ی مخفی کاملا متصل تشکیل شده‌است. لایه‌ی خروجی، یک لایه‌ی کاملا متصل خطی بوده که در آن برای هر عمل در بازی یک نورون در نظر گرفته شده است. شبکه مقادیر Q هر عمل حالت

[1] Deeplearning4j
[2] Arcade Learning Environment
[3] Q*bert

ورودی را به‌طور جداگانه محاسبه کرده که در آن هر حالت یک پشته از آخرین چهار فریم مشاهده شده می‌باشند.

یک شبکه جداگانه برای هر بازی آموزش می‌بیند که از یک معماری، الگوریتم یادگیری و تنظیمات فراپارامتری یکسان در آنها برای همه بازی‌ها استفاده می‌شود. این امر نشان می‌دهد که این رویکرد با داشتن کمترین دانش ممکن، پایداری کافی برای کار کردن با بازی‌های مختلف را دارد. با اینکه ارزیابی عامل روی بازی‌های بدون تغییر انجام گرفته است، برای برقراری شرایط یکسان بین روش ارائه شده و DQN، ساختار پاداش‌ها تنها در زمان آموزش تغییر یافتند به شکلی که پاداش‌های مثبت به ۱ و پاداش‌های منفی به ۱- تبدیل شده و پاداش صفر بدون تغییر باقی ماند. این نحوه برش پاداش‌ها، مقیاس مشتق خطاها را محدود کرده و استفاده از یک نرخ یادگیری یکسان بین تمامی بازی‌ها را آسان‌تر می‌کند. اما این کار روی کارایی عامل تاثیرگذار خواهد بود، چراکه توانایی تمایز قایل شدن بین اندازه پاداش‌های گرفته شده از آن سلب می‌شود.

سیاست رفتاری استفاده شده در طول آموزش از نوع ε حریصانه بوده که در آن ε به صورت خطی از مقدار ۱ تا ۰.۱ در یک میلیون فریم اول کاهش یافته و پس از آن مقدار ثابت ۰.۱ حفظ شده است. در اینجا یک شکل ساده از تکنیک پرش فریم که در قسمت‌های قبلی به آن اشاره شد، استفاده شده است. به طور دقیق‌تر، عامل به جای تمام فریم‌ها، هر k فریم را دیده و عمل مورد نظر خود را انتخاب می‌کند و عمل انتخاب شده در فریم‌هایی که از روی آنها پرش انجام شده تکرار می‌شود. علت این کار نیز، کاهش مقدار محاسبات مورد نیاز در پیش بردن شبیه‌ساز در یک گام است، زمانی که عملی انتخاب نمی‌شود. همچنین این تکنیک به عامل اجازه می‌دهد تا به طور تقریبی k برابر بیشتر بازی انجام دهد، بدون اینکه زمان اجرا افزایش شدیدی پیدا کند. در اینجا همانند DQN از مقدار ۴ برای پارامتر k استفاده شد.

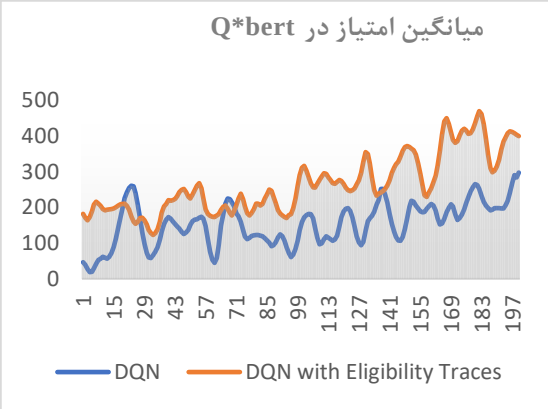
ارزیابی سیاست یاد گرفته شده توسط عامل با اجرای یک سیاست ε حریصانه در هر ۱۰ قسمت، انجام شد. برای این کار مقدار ε برابر ۰.۰۵ در نظر گرفته شده و میانگین‌گیری امتیازات و گام‌های بدست آمده در ۱۰ قسمت محاسبه شده است. آموزش شبکه در ۲۰۰ اپاک انجام شد که هر اپاک برابر ۱۰ قسمت بوده و اندازه حافظه تکرار تجربه ۵۰۰۰۰۰ قرار گرفتند. تمام وزن‌های شبکه‌ها با بهینه‌ساز RMSProp بروزرسانی شدند که در آن نرخ یادگیری α برابر ۰.۰۰۰۲۵ و مومنتوم برابر ۰.۹۵ قرار داده شد. بروزرسانی شبکه هدف پس از هر ۱۰۰۰۰ گام انجام شد. آموزش شبکه برای تمام بازی‌ها، بدون تغییر در معماری آن‌ها و فراپارامترها انجام شدند و باقی تنظیمات همانند [۱۲] در نظر گرفته شدند.

مقادیر فراپارامترها و پارامترهای بهینه‌سازی در DQN با یک جستجوی غیر رسمی روی چند بازی انتخاب شدند، چراکه یک جستجوی سیستماتیک هزینه محاسباتی بالایی را طلب می‌کرد. در نهایت این پارامترها در بین دو بازی که ارزیابی روی آنها انجام گرفت، بدون تغییر نگه داشته شدند.

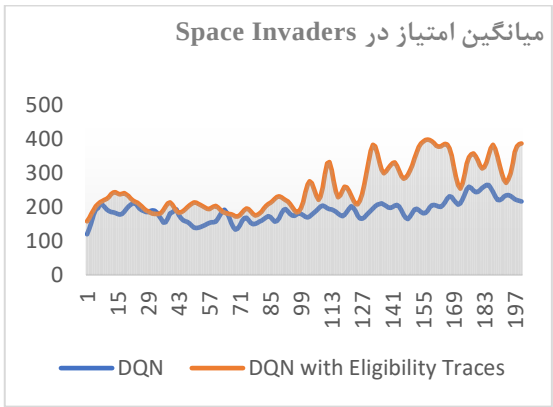
در این مقاله نیز همانند DQN، کمترین دانش اولیه برای محیط آزمایش فراهم شده است: داده‌های ورودی از تصاویر تشکیل شده‌اند (که علت استفاده از شبکه‌های کانولوشنال عمیق را توجیه می‌کند) و تعداد عمل‌ها.

۵-۱ نتایج

برای بررسی معتبر بودن نتایج این رویکرد، آن را با شبکه Q عمیق مقایسه می‌کنیم. نتایج ارائه داده شده در شکل‌های پایین نتایج کارایی الگوریتم ارائه شده و DQN را نشان می‌دهند. شکل ۳شکل ۱و شکل ۴، میانگین کل پاداش‌های جمع‌آوری شده را به ترتیب در ۲ بازی کیوبرت و اسپیس اینویدرز نشان می‌دهند.



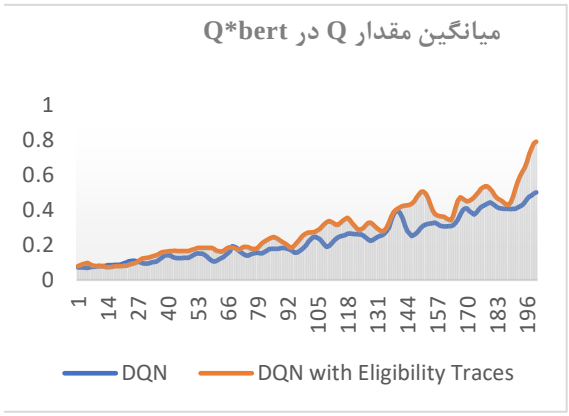
شکل ۳: مقایسه کارایی الگوریتم ارائه شده و DQN، با میانگین امتیاز به دست آمده در هر اپاک در بازی کیوبرت



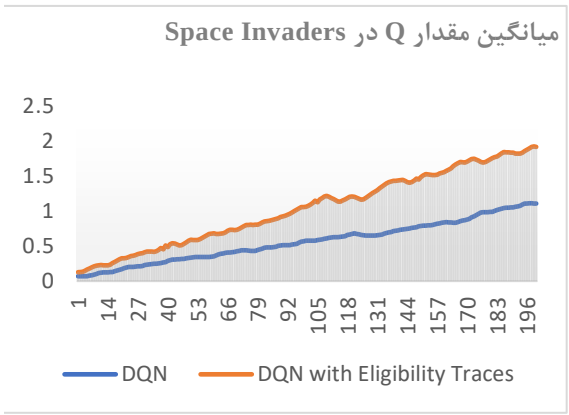
شکل ۴: مقایسه کارایی الگوریتم ارائه شده و DQN، با میانگین امتیاز به دست آمده در هر اپاک در بازی اسپیس اینویدرز

همانطور که توسط [۱۲] شرح داده است، معیاری که برای ارزیابی عامل یادگیری تقویتی استفاده می‌شود، مقدار Q تخمین زده شده سیاست است که پاداش دریافتی را هنگامی که عامل یک سیاست خاص را دنبال می‌کند، محاسبه می‌کند. شکل‌های سمت راست نشان می‌دهند که مقدار میانگین Q تخمین زده شده روش ما در طول زمان با سرعت بیشتری از DQN افزایش می‌یابند. این نشان دهنده این است که مدل به تدریج و به

صورت پایدار در حال یادگیری بوده و بسیار سریعتر از DQN عمل می‌کند. شکل ۵ و شکل ۶، میانگین Q پیش‌بینی شده در طول زمان آموزش را به ترتیب در بازی‌های کیوبرت و اسپیس اینویدرز نشان می‌دهند.



شکل ۵: مقایسه کارایی الگوریتم ارائه شده و DQN، با میانگین مقادیر Q پیش‌بینی شده در هر اپیاک در بازی کیوبرت



شکل ۶: مقایسه کارایی الگوریتم ارائه شده و DQN، با میانگین مقادیر Q پیش‌بینی شده در هر اپیاک در بازی اسپیس اینویدرز

همانطور که انتظار می‌رفت، نتایج ما شتاب گرفتن یادگیری را نشان می‌دهند. اشکال بالا همگرایی سریع‌تر الگوریتم در مقایسه با DQN را نشان می‌دهد. همانطور که از نتایج برمی‌آید روش ارائه شده می‌تواند به طور تقریبی ۱.۵ برابر سریعتر از DQN به امتیاز میانگین بهینه برسد و همچنین در طول زمان یادگیری، میانگین امتیاز بسیار بهتری به دست آمد.

از نظر زمان صرف شده برای یادگیری و آموزش شبکه، زمان موردنیاز برای آموزش در هر اپیاک در روش پیشنهادی حدود ۱۰٪ از زمان موردنیاز برای آموزش هر اپیاک در DQN بیشتر است، ولی با توجه به سرعت بسیار بالاتر یادگیری از نظر تعداد اپیاک موردنیاز برای رسیدن به مقدار Q یکسان یا از منظر کسب امتیاز در روش پیشنهادی نسبت به DQN (حدود ۱.۵ برابر)، در مجموع زمان کمتری برای رسیدن به مقدار Q یکسان یا جمع آوری مقدار مشخصی امتیاز در روش پیشنهادی نسبت به روش DQN موردنیاز است.

برای بررسی بیشتر یک آزمایش t جفت[1] نیز انجام گرفت تا میانگین کل پاداش دریافت شده بین الگوریتم ارائه شده و DQN در بازی‌های کیوبرت و اسپیس اینویدرز مقایسه شود. در جدول ۱ قابل مشاهده است که الگوریتم ارائه شده میانگین پاداش بسیار بالاتری ($p < ۰.۰۵$) برای هر بازی به دست آورده است. این نتایج بیانگر این است که روش ارائه شده می‌تواند امتیازات بیشتری در مراحل ابتدایی یادگیری گرفته و در نتیجه سرعت یادگیری را افزایش دهد.

جدول ۱: نتایج آزمایش t جفت برای مقایسه کل پاداش دریافت شده در الگوریتم پیشنهادی و الگوریتم DQN

بازی	تعداد اپیاک	ثابت t	مقدار p
کیوبرت	۱۰۰	-۲.۶۹۶	۰.۰۰۸
اسپیس اینویدرز	۱۰۰	-۲.۹۶۷	۰.۰۰۳

۶- نتیجه‌گیری و راهکارهای آینده

روش ارائه شده در این مقاله، ترکیبی از شبکه‌های عصبی عمیق و یادگیری Q(λ) است که الگوریتم DQN را با مکانیزم آثار شایستگی توسعه می‌دهد. این ترکیب جدید، به ما اجازه می‌دهد تا با بهره بردن از مزایای الگوریتم DQN و مکانیزم آثار شایستگی، فرایند یادگیری را سرعت ببخشیم. همچنین روش ارائه شده با DQN در دو بازی آتاری ۲۶۰۰ مقایسه شد که تنها از پیکسل‌های خام به عنوان ورودی استفاده می‌کرد. نتایج تجربی به دست آمده در دو بازی که آزمایش روی آنها انجام شد، نشان دادند که روش ترکیبی ارائه شده می‌تواند، سیاست‌های کنترلی مناسب را، در تعداد گام‌های کمتری نسبت به DQN بیاموزد.

همانطور که در بخش روش پیشنهادی اشاره شد، پیش از دادن فریم‌های دریافتی از شبیه‌ساز ALE به شبکه عصبی، پیش‌پردازشی همانند کاری که DQN انجام داد، روی فریم‌ها انجام می‌شد، به طوری که فریم‌ها به قالب سیاه و سفید تبدیل شده و تنها ناحیه‌ای به ابعاد ۸۴ * ۸۴ از آن استخراج می‌شد. علت این امر نیز کاهش پیچیدگی محاسباتی و حافظه‌ای بیان شد که می‌تواند موجب کند شدن کل فرایند ساخت مدل شود. این کار می‌تواند قسمت‌هایی از ورودی را که امکان وجود اطلاعات مفید برای یادگیری در آنها وجود داشته باشد، مانند رنگ اشیاء مختلف در بازی و همچنین نواحی خارج از ابعاد بریده شده که بازی در آن جریان دارد، دور بریزد. اما با توجه به پیشرفت روزانه سخت‌افزارها و افزایش سرعت آنها، می‌توان از چنین پیش‌پردازش‌هایی چشم‌پوشی کرده و به نتایجی بهتر دست یافت.

مکانیزم آثار شایستگی، مکانیزمی پایه‌ای در یادگیری تقویتی است و می‌تواند به تسریع یادگیری کمک شایانی بکند. در بخش کارهای مرتبط، روش‌های دیگری نیز معرفی شدند که هرکدام تلاش می‌کردند به نحوی مشکلات الگوریتم یادگیری Q و DQN را برطرف نمایند. مکانیزم آثار

[1] Paired t-test

شایستگی می‌تواند از لحاظ تئوری با تمامی این الگوریتم‌ها ترکیب شود و پتانسیل بهبود سرعت یادگیری را در آنها دارد. مانند [۲۴] که در آن الگوریتم رنگین‌کمان نشان داد که می‌توان با ترکیب روش‌هایی که ایرادات قسمت‌های مختلفی از DQN را برطرف می‌کنند، به نتایج بسیار بهتری دست یافت. همچنین این الگوریتم نیز می‌تواند با استفاده از این مکانیزم، سرعت یادگیری خود را بهبود ببخشد.

مراجع

[1] A. J. Krener and W. Respondek, “Nonlinear observer with linearizable error dynamics,” SIAM J. Control & Optim., vol. 23, pp. 197-216, 1985.

[2] R. S. Sutton, and A. G. Barto, Introduction to Reinforcement Learning: MIT Press, 1998 .

[3] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” Journal of artificial intelligence research, vol. 4, pp. 237-285, 1996 .

[4] C. J. C. H. Watkins, “Learning from Delayed Rewards,” King's College, Cambridge University, Cambridge, UK, 1989 .

[5] G. A. Rummery, and M. Niranjan, On-Line Q-Learning Using Connectionist Systems, Engineering Department, Cambridge University, Cambridge, UK 1994 .

[6] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238-1274, 2013 .

[7] D. Vengerov, “A reinforcement learning approach to dynamic resource allocation,” Engineering Applications of Artificial Intelligence, vol. 20, no. 3, pp. 383-390, 2007 .

[8] A. G. Barto, and S. Mahadevan, “Recent Advances in Hierarchical Reinforcement Learning,” Discrete Event Dynamic Systems, vol. 13, no. 4, pp. 341-379, 2003 .

[9] R. S. Sutton, “Learning to Predict by the Methods of Temporal Differences,” Machine Learning, vol. 3, no. 1, pp. 9-44, 1988.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. Nature, 521(7553):436–444, 2015.

[11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. IEEE Trans. on Pattern Analysis and Machine Intelligence, 35(8):1798–1828, 2013.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari With Deep Reinforcement Learning,”, In proceedings of NIPS Deep Learning Workshop, 2013 .

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529- 533, 2015 .

[14] Christopher JCH Watkins and Peter Dayan. "Q-Learning" . Machine Learning, vol. 8 No. 3, pp. 279–292, 1992.

[15] Wang X, Deep reinforcement learning: case study with standard RL testing domains, Master’s thesis, Technische Universiteit Eindhoven, Eindhoven, 2016

[16] J. Peng, and R. J. Williams, “Incremental multi-step Q-learning,” Machine Learning, vol. 22, no. 1-3, pp. 283-290, 1996 .

[17] R. S. Sutton, A. M. David, P. S. Satinder, and Y. Mansour, “Policy Gradient Methods for Reinforcement Learning with Function Approximation,” In Advances in Neural Information Processing Systems (NIPS) 12, pp. 1057--1063, 2000 .

[18] J. N. Tsitsiklis, and B. Van Roy, “An analysis of temporal-difference learning with function approximation,” IEEE transactions on automatic control, vol. 42, no. 5, pp. 674-690, 1997 .

[19] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993

[20] van Hasselt, H.; Guez, A.; and Silver, D. Deep reinforcement learning with double Q-learning. In Proc. Of AAAI, pp. 2094–2100, 2016.

[21] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. In proceedings of ICLR, 2016.

[22] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling Network Architectures for Deep Reinforcement Learning. In proceedings of ICLR, 2016.

[23] Leemon C Baird . Advantage Updating. Technical report, DTIC Document, 1993.

[24] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In proceedings of ICLR, 2016.

[25] Hessel M., Modayil J., Van Hasselt H., Schaul T., Ostrovski G., Dabney W., Horgan D., Piot B., Azar M., and Silver D., Rainbow: Combining improvements in deep reinforcement learning. In proceedings of AAAI Conference on Artificial Intelligence (AAAI). 2018.

[26] Alex Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. Frame skip is a powerful

parameter for learning to play atari. In proceedings of AAAI workshop, 2015.

[27] Aravind S Lakshminarayanan, Sahil Sharma, and Balaraman Ravindran. Dynamic frame skip deep q network, In Proceedings of the Workshops at the International Joint Conference on Artificial Intelligence. New York, USA, 2016.

[28] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In proceedings of ECML, pp. 317–328., 2005.

[29] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In proceedings of The 2010 International Joint Conference on Neural Networks (IJCNN), pp. 1–8., 2010.

[30] https://github.com/deepmind/dqn, Accessed at august 2018 .

[31] Hinton Geoffrey, Lecture 6a: Overview of mini-batch gradient descent, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, accessed at august 2018

[32] Kovalev, Vassili & Kalinovsky, Alexander & Kovalev, Sergey. Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy?, In proceedings of The 13th International Conference on Pattern Recognition and Information Processing, 2016

[33] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. In proceeidings of IJCAI, 2015.