심층강화학습 라이브러리 기술동향

A Survey on Deep Reinforcement Learning Libraries

신승재 (S.J. Shin, sjshin0505@etri.re.kr) 지능네트워크연구실 연구원
조충래 (C.L. Cho, clcho@etri.re.kr) 지능네트워크연구실 책임연구원
전흥석 (H.S. Jeon, jeonhs@etri.re.kr) 지능네트워크연구실 선임연구원
윤승현 (S.H. Yoon, shpyoon@etri.re.kr) 지능네트워크연구실 책임연구원
김태연 (T.Y. Kim, tykim@etri.re.kr) 지능네트워크연구실 책임연구원/실장

ABSTRACT

Reinforcement learning is a type of machine learning paradigm that forces agents to repeat the observation-action-reward process to assess and predict the values of possible future action sequences. This allows the agents to incrementally reinforce the desired behavior for a given observation. Thanks to the recent advancements of deep learning, reinforcement learning has evolved into deep reinforcement learning that introduces promising results in various control and optimization domains, such as games, robotics, autonomous vehicles, computing, industrial control, and so on. In addition to this trend, a number of programming libraries have been developed for importing deep reinforcement learning into a variety of applications. In this article, we briefly review and summarize 10 representative deep reinforcement learning libraries and compare them from a development project perspective.

KEYWORDS 심층강화학습, 심층강화학습 라이브러리

Ⅰ. 서론

강화학습(Reinforcement learning)이란 제어 및 의사결정의 대상이 되는 환경 또는 시스템에서 에이전트가 다양한 행동을 반복하며 누적한 경험을 통해 제어 및 의사결정의 전략을 개선해가는 기계학습 기법을 말한다. 강화학습은 1950년대 처음 제안된 이래[1], 지속적인 연구가 이루어져 왔으며[2], 각종 제어 및 최적화(Optimization) 문제를 해결하는 대표적인 패러다임 중의 하나로 인식되고 있다.

2010년대 이후 심층기계학습(Deep learning)이

* DOI: https://doi.org/10.22648/ETRI.2019.J.340608
* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 정보통신 · 방송 연구개발 사업의 일환으로 수행하였음[2017-0-00045, 초연결 지능 인프라 원천기술 연구개발].

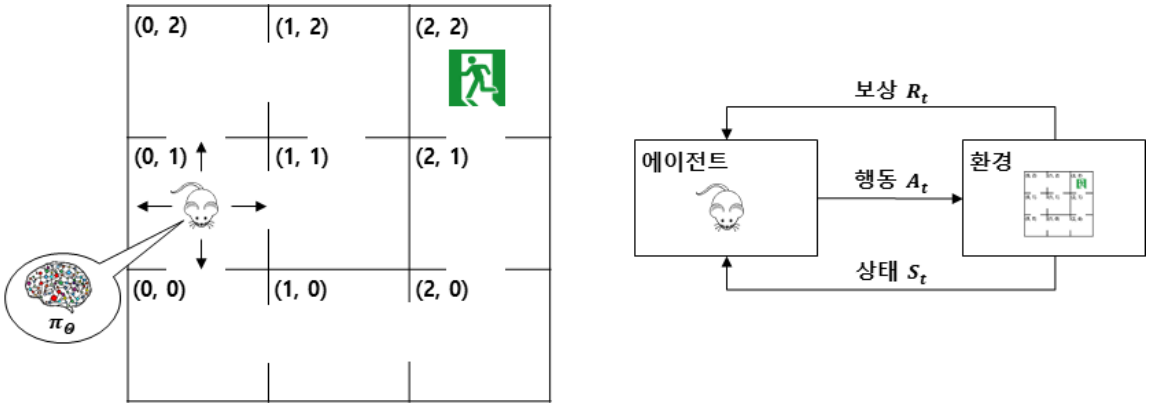

그림 1 강화학습 예시: 미로탈출 에이전트

기계학습의 새로운 혁신적 대안(Break-through)으로 등장하면서[3], 심층신경망(DNN: Deep Neural Network)을 강화학습에 결합한 심층강화학습(Deep reinforcement learning)[1] 역시 다양한 제어 및 최적화 문제에 관한 새로운 해법으로 주목받게 되었다. 심층강화학습은 게임(Game)[4], 로보틱스(Robotics)[5], 무인비행(UAV: Unmanned Aerial Vehicle)[6], 자율주행(Autonomous driving)[7], 컴퓨팅 및 통신 시스템 제어[8-11], 경영 및 금융 의사결정[12] 등 다양한 도메인에서 유망한 가능성을 보여주었으며, 특히 DeepMind의 AlphaGo는 바둑에서 세계 최고 수준의 기사들을 상대로 연이어 승리함으로써 상당한 반향을 일으킨 바 있다[13].

상기 언급한 근래의 기술적 경향에 따라, 심층강화학습을 다양한 시스템 및 솔루션에 활용하기 위한 프로그래밍 라이브러리(Library)들이 다수 제안되었다. 본 고에서는 근래 제안된 대표적인 공개형(Open source) 심층강화학습 라이브러리 10종을 요약 및 분석하고, 이들의 특징을 비교함으로써 각종 개발 프로젝트에 심층강화학습 활용 시 라이브러리 선택에 참고할 만한 유용한 정보를 제시하고자 한다.

Ⅱ. 심층강화학습 개관

강화학습은 제어이론의 MDP(Markov Decision Process) 최적화 및 동물심리학의 trial-and-error 개념을 결합한 기계학습 방법론으로써 각종 제어 및 최적화 문제를 풀기 위한 근사화기법 중 하나로 알려져 있다. 기존 강화학습은 다른 최적화기법[예: DP(Dynamic Programming), LP(Linear Programming), NLP(Non-Linear Programming) 등]들을 압도하는 결과를 보여주지 못하였으나, 2010년 이후 DNN이 결합된 심층강화학습으로 진화함으로써 복잡한 제어 및 최적화 문제를 해결하는 혁신적 해법으로 주목받게 되었다.

강화학습에서 에이전트는 단계(또는 시간) t마다 환경의 상태 S_t를 관찰하고, 내부의 파라미터화된 근사화함수(Approximator)인 정책함수(Policy function) π_θ[1]를 통해 행동 A_t를 결정한다. 이후 에이전트는 행동 A_t에 대한 보상 R_t를 부여받고, 다음 단계 $t+1$로 진행한다.

그림 1은 간단한 미로탈출 문제를 강화학습 모

1) 학습 알고리즘 또는 구현 상세에 따라 상태가치함수 $V_\theta(S_t)=v$ 또는 행동가치함수 $Q_\theta(S_t, A_t)=q$ 등과 같은 다른 근사화함수를 함께 사용한다. 자세한 사항은 참고문헌 [1,2]의 참조를 권한다.

델로 예시한 것이다. 그림의 예에서 쥐는 에이전트, 미로는 환경에 해당되며, 상태 S_t는 미로 상에서 쥐의 위치[예: (0, 1)], 행동 A_t는 쥐의 이동(↑, ↓, ←, →)을 의미한다. 보상 R_t는 미로를 탈출하지 못한 경우 −1, 미로를 탈출한 경우 +1이 주어진다. 즉 미로를 탈출한 경우 포상적(Incentive) 성격, 탈출하지 못한 경우 징벌적(Penalty) 성격의 보상이 주어짐을 알 수 있다.

정책함수 π_Θ는 학습 알고리즘 또는 구현 상세에 따라, $\pi_\Theta(A_t|S_t)=p$ 또는 $\pi_\Theta(S_t)=a$ 형태를 지닌다. 전자의 경우, 상태 S_t가 관찰되었을 때, 행동 A_t를 선택할 확률 p를 출력하며, 후자의 경우, 상태 S_t에 대응되는 행동 a를 직접 출력한다. Θ는 π_Θ의 입출력을 결정하는 파라미터로 일반적으로 다음과 같은 벡터 형태로 표기한다.

$$\Theta=[\theta_0,\theta_1,\theta_2,...,\theta_N] \tag{1}$$

에이전트의 학습이란 상태 S_t의 관찰, 행동 A_t의 결정, 보상 R_t의 지급, 다음 상태 S_{t+1}로 진행하는 것을 반복하며, 기대누적보상(Sum of expected rewards)을 최대화하는 방향으로 π_Θ를 점진적으로 개선하는 것을 지칭한다. 이를 정책개선(Policy improvement)이라고 하고, 이를 위해 사용하는 데이터 튜플(S_t, A_t, R_t, S_{t+1})을 전이(Transition)라고 하며, 시간에 따른 전이의 시퀀스(Sequence)를 사건(Episode)이라고 한다. 그리고 다수 개의 사건들의 전이들을 모은 집합을 경험(Experience)이라고 부른다. 즉, 강화학습은 "$\Theta_0\rightarrow(S_0,\ A_0,\ R_0,\ S_1)\rightarrow\Theta_1\rightarrow(S_1,\ A_1,\ R_1,\ S_2)\rightarrow\Theta_2\rightarrow(S_2,\ A_2,\ R_2,\ S_3)\rightarrow\Theta_3\rightarrow\cdots\rightarrow\Theta_T$"와 같이 전이와 정책개선이 반복되는 과정으로 표현될 수 있다.[2] 해당과정을 통해 파라미터가 Θ_0에서 Θ_T로 개선되면서, π_Θ는 기대누적보상을 증가시키는 행동 패턴들을 강화(Reinforce)한다. 이를 그림 1로 예시하면, 에이전트(쥐)는 좌표의 관찰과 이동 결정을 반복하며 다양한 전이 데이터[예: ((0, 1), ↑, (0, 2), −1)]를 수집하여, π_Θ의 개선을 반복하는 것으로 설명할 수 있다. 에이전트가 출구가 위치한 (2, 2)에 도착하면, 해당 사건이 종결되고, 에이전트 위치가 다시 초기화된다. 전이 및 사건이 여러 차례 반복되면서 탈출시간을 단축시키는 방향으로 π_Θ가 개선된다. 이후 지정된 조건(예, 기대누적보상이 정해진 목표에 도달하거나 정책개선이 반복된 횟수가 임계량을 초과했을 경우)을 만족하면, 정책개선을 멈추고, 학습을 완료한다.

심층강화학습에서 π_Θ는 DNN으로 구현되며,[3] Θ는 DNN의 파라미터, 즉 가중치 벡터 또는 행렬이 된다. 이것을 통해 심층강화학습의 정책개선은 SGD(Stochastic Gradient Descent) 또는 오류반송(Error back-propagation) 알고리즘을 통해 DNN을 훈련(Train)시키는 과정으로도 이해할 수 있다. 이를 수학적 기호로 표현하면 다음과 같다.

$$\Theta^{new}=\Theta-\alpha\ \cdot\ \nabla_\Theta L(\Theta) \tag{2}$$

상기 식에서 $\alpha(\in[0,1])$는 학습률, $L(\Theta)$는 손실(Loss), 오류(Error) 또는 비용(Cost) 함수라고 부른다. $\nabla_\Theta L(\Theta)$는 손실함수의 Θ에 대한 미분값으로 변화(Gradient) 함수라고 부르며, SGD 또는 오류반송 알고리즘에 의해 반송 또는 정정될 손실량(또는 오류량) 벡터를 의미한다.

심층강화학습 에이전트의 설계는 다음의 항목들을 정의함으로써 이루어진다.

2) 실제 많은 구현에서는 여러 차례의 전이당 한 차례의 정책개선이 이루어지는 일괄처리(Batch update) 형태로 학습이 이루어진다. 이는 DNN 훈련 시 다수 개의 데이터를 한 번에 일괄적으로 입력하는 특성에 따른 것이다.

3) 일부 강화학습 알고리즘은 가치함수를 DNN으로 구현하고, π_θ는 가치함수에 종속된 알고리즘으로 구현하기도 한다. 대표적인 예로 DQN[4]이 있다.

표 1 심층강화학습 알고리즘별 특징

종류	특징
DQN [4]	• 근사화함수는 행동가치함수 $Q_\Theta(S_t, A_t) = q$로만 구성. 정책함수 $\pi_\Theta(S_t) = a$는 탐색률 ϵ의 확률로 임의(Random) 행동을 출력하다가, 1-ϵ의 확률로 $a = \arg\max_a Q_\Theta(S_t, \alpha)$를 출력. 즉, π_Θ는 행동가치함수에 종속되도록 프로그램된 함수 • 손실함수는 $L(\Theta) = E[(R_t + \gamma \max_a Q_\Theta(S_{t+1}, a) - Q_\Theta(S_t, A_t))^2]$ • 정책개선 시 N-step back-up[2], PER(Prioritize Experience Replay)[18], network dueling[19] 및 doubling[20] 등의 각종 성능향상 기법 결합 가능
A3C [14]	• 정책개선 시 다수 개의 (에이전트, 환경)-쌍이 동시에 여러 개의 사건(Episode)을 수행하는 비동기(Asynchronous) 병렬학습 형태로 동작하여 성능 및 학습속도 향상 • 근사화함수는 각 (에이전트, 환경)-쌍별로 정책함수 $\pi_\Theta(A_t \mid S_t) = p$ 및 상태가치함수 $V_{\Theta_V}(S_t) = v$를 사용 • 손실함수는 $L(\Theta) = E\left[\log \pi_\Theta(A_t \mid S_t) A_{\Theta_V}(S_t)\right]$ 및 $L(\Theta_V) = E\left[(A_{\Theta_V}(S_t))^2\right]$
DDPG [15]	• 행동집합을 연속공간으로 정의할 수 있으므로, 로보틱스 및 자율주행과 같이 제어변수가 연속적인 도메인에 적합 • 근사화함수는 정책함수 $\pi_\Theta(S_t) = a$ 및 행동가치함수 $Q_{\Theta_Q}(S_t, A_t) = q$와 정책개선 누적을 위한 target 함수 $\pi_\Theta^{target}(S_t) = a^{target}$ 및 $Q_{\Theta_Q}^{target}(S_t, A_t) = q^{target}$으로 구성. 매 정책개선 완료 시 $\Theta \leftarrow \Theta^{target}$ 및 $\Theta_Q \leftarrow \Theta_Q^{target}$를 수행하여 파라미터를 동기화 • 손실함수는 $\nabla_\Theta L(\Theta) = E\left[\nabla_a Q_{\Theta_Q}(S_t, a)\right]_{a=\pi_\Theta(S_t)} \nabla_\Theta \pi_\Theta(S_t)$ 및 $L(\Theta_Q) = E\left[(R_t + \gamma Q_{\Theta_Q}^{target}(S_{t+1}, \pi_\Theta^{target}(S_{t+1})) - Q_{\Theta_Q}(S_t, A_t))^2\right]$
TRPO [16]	• 정책개선 과정에서 현재 입력된 전이 데이터의 과도한 학습으로 인해, 이전에 학습된 내용에 왜곡을 일으키는 현상을 억제함으로써 성능 및 학습속도 향상 • 근사화함수는 정책함수 $\pi_\Theta(A_t \mid S_t) = p$ 및 행동가치함수 $V_{\Theta_V}(S_t) = v$와 전 단계 정책의 복사본 $\pi_\Theta^{prev}(A_t \mid S_t) = p^{prev}$ 및 $V_{\Theta_V}^{prev}(S_t) = v^{prev}$으로 구성. 매 정책개선 완료 시 $\Theta^{prev} \leftarrow \Theta$ 및 $\Theta_V^{prev} \leftarrow \Theta_V$를 수행하여 전 단계 정책을 보존 • 손실함수는 $L(\Theta) = E\left[\frac{\log \pi_\Theta(A_t \mid S_t)}{\log \pi_\Theta^{prev}(A_t \mid S_t)} A_{\Theta_V^{prev}}(S_t) - P(\gamma, \epsilon, \Theta, \Theta^{prev})\right]$, $L(\Theta_V)$는 A3C와 동일. $L(\Theta)$에서 $P(\Theta, \Theta^{prev}, \gamma, \epsilon) = \frac{2\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\Theta \| \Theta^{prev})$는 KL 발산(Kullback-Leibler divergence)[21] 기반의 규제(Penalty) 함수. ϵ은 규제강도를 결정하는 계수
PPO [17]	• 정책개선 과정에서 근사화함수 파라미터들의 과도한 변화를 억제한다는 점에서 TRPO와 동일한 개념을 지향함. TRPO보다 구현 복잡도 대비 성능 우수 • 근사화함수 구성은 TRPO와 동일 • 손실함수 $L(\Theta)$는 TRPO와 유사하나, 규제함수 P 대신 $\frac{\log \pi_\Theta(A_t \mid S_t)}{\log \pi_\Theta^{prev}(A_t \mid S_t)}$ 성분이 특정 임계값 이상으로 커지지 않도록 정류(Clip)하는 방식을 사용. $L(\Theta_V)$는 TRPO와 동일

* 표에 기술한 기호 중 $\gamma(\in [0,1])$는 가치감쇄율이라고 한다. $A_{\Theta_V}(S_t)$는 상대가치(Advantage)라고 하며, $A_{\Theta_V}(S_t) = \sum_{i=0}^{N} \gamma^i R_{t+i} + V_{\Theta_V}(S_{t+N}) - V_{\Theta_V}(S_t)$로 정의된다.

- 상태집합
- 행동집합
- 보상함수
- 정책개선절차
- 근사화함수 모델링

상기 항목에서 상태 및 행동집합은 환경 및 문제의 특성을 고려하여 결정한다. 그림 1의 예시에서 상태집합은 {(0, 0), (0, 1), (0, 2), …, (2, 2)}, 행동집합은 {↑, ↓, ←, →}로 정의되었다. 보상함수는 최적화 목표를 고려하여 결정된다. 그림 1의 예에서는 에이전트가 아직 미로에 있을 때 −1, 출구에 도착했을 때 +1로 정의하였다. 이는 탈출 소요시간을 최소화하는 것을 최적화 목표로 반영한 것이다. 정책개선절차는 각 근사화함수의 정책개선 과정 및 손실함수[4]를 정의하는 부분으로, 적용하는 강화학습 알고리즘의 종류에 따라 달라진다. 표 1은 대표적인 강화학습 알고리즘 5종의 정책개선 절차상 특징을 약술한 것이다. 알고리즘 각각의 상세는 참고문헌 [1,2,4,14−17]의 참조를 권한다. 근사화함수 모델링은 각각의 근사화함수에 사용될 DNN 모델을 정의하는 것을 지칭한다.[5] DNN 모델링의 상세는 참고문헌 [22,23]의 참조를 권한다.

최근 심층강화학습 기술은 상기 언급한 수준을 넘어 훈련 시 다수 개의 CPU(Central Processing Unit), GPU(Graphical Processing Unit) 또는 노드(Node)상에서(에이전트, 환경)−쌍을 분할 배치하고, 이들이 생성하는 경험을 정책개선 과정에서 융합함으로써 학습의 속도와 품질을 높이는 분산(Distributed) 강화학습[24−26], 다수 개의 에이전트들이 협업 또는 경쟁하는 환경에서 총 누적보상의 향상을 꾀하는 다중 에이전트(Multi−agent) 강화학습[27−29] 등의 형태로 진화하고 있다.

III. 심층강화학습 라이브러리

1. 분석기준

본 절에서는 심층강화학습 라이브러리들을 분석하기 위한 기준들을 제시한다. 해당 기준들은 주로 라이브러리와 개발 프로젝트 간 적합성에 중점을 두었다.

- **목표 및 지향점:** 제작자들이 명시한 라이브러리의 목표 및 지향점은 라이브러리와 프로젝트 간 적합성을 가늠하는 데 필요한 기초적인 정보가 된다.
- **개발 지속성:** 공개 저장소(Repository)의 별점(Star), 기여자(Contributor) 및 변경(Commit) 내역 등은 해당 라이브러리가 지속적으로 유지, 보수되고 발전되는가를 판단하는 기준으로 사용할 수 있다.
- **알고리즘 다양성:** 라이브러리가 제공하는 정책개선 알고리즘이 다양할수록 에이전트 구현 시 선택의 폭이 넓어지고, 융통성의 향상을 기대할 수 있다.
- **환경 지원성:** 강화학습 에이전트 구현 시 환경은 에이전트의 학습, 시험 및 적용 대상인 응용(Application)을 지칭한다. 심층강화학습 라이브러리가 프로젝트에서 가정하는 응용환경을 지원하지 않으면, 이를 보완하기 위한 추가 개발이 불가피하다. 즉, 환경 지원성은 라이브러리와 개발 프로젝트 간의 적합성을 가늠하는 중요한 기준으로 볼 수 있다.
- **분산학습 및 다중 에이전트 지원:** 분산학습과 다중 에이전트는 최근 심층강화학습 기술의

4) 손실함수 대신 변화함수로 정의하기도 한다.

5) DNN 기반의 회귀(Regression) 예측기(Predictor)를 설계하는 것으로, 일반적인 지도학습(Supervised learning)과 개념상으로 유사한 부분이다.

발전 및 진화를 주도하는 개념으로 볼 수 있다. 높은 학습속도 및 확장성을 필요로 하는 프로젝트의 경우 분산학습 및 다중 에이전트 지원 여부가 중요한 사항으로 고려될 것이다.

- **개발 편의성:** 소스코드(Source Code)의 복잡도가 높거나 충분한 문서화가 수반되지 않는 경우 라이브러리 습득을 위한 시간적 비용이 증가한다. 따라서 개발 편의성은 프로젝트의 시간 및 인적 제약성이 높은 경우 중요한 기준으로 사용될 것이다.

2. 심층강화학습 라이브러리 일람

본 절에서는 근래 제안된 공개형 심층강화학습 라이브러리 중 10종을 선택하여 상기 제시한 기준으로 요약 및 분석한다.

가. Coach

Coach[30]는 스타트업 기업인 Nervana Systems에서 처음 개발을 시작하였으며, 2016년 Intel에 인수되어, 2017년 공개형 SW(Software)로 전환하였다. 이후 현재[6]까지 Intel의 비공식 프로젝트로 진행되고 있다. 개발진들은 다양한 강화학습 알고리즘을 쉽게 구현 가능한 모듈화된 빌딩 블록(Building block)을 API(Application Programming Interface) 형태로 제공하는 것을 목표로 언급하였다.

참고문헌 [31]에 따르면, 9회의 개정을 거쳐 배포판 1.0.0이 발표되었고, 28인의 기여자가 현재까지 총 491회의 변경을 수행하였으며, 최근까지 활발히 업데이트되고 있다. 현재 GitHub 별점은 1,470점이다. 동작 OS 및 프로그래밍 환경으로 Ubuntu 16.04 LTS(Long-Term Support) 및 Python 3.6이 테스트되었으며, 기반 DNN 라이브러리로 TensorFlow[32] 및 MXNet[33]을 지원한다. 또한, Intel Optimized TensorFlow[34]를 활용하여 CPU-only 환경에서의 성능 향상을 위한 옵션도 제공하고 있다. 제공하는 알고리즘은 24종(A3C, DQN, PPO, DDPG, DFP, NAF, Rainbow, SAC 등)이며, 8종의 환경(OpenAI Gym[35], Roboschool[36], Gym Extensions[37], PyBullet[38], VizDoom[39], CARLA[40], PySC2[41], DeepMind Control Suite[42])을 지원한다. 또한, 다중 CPU 및 다중노드(Multi-node) 분산학습과 계층적 다중 에이전트를 지원한다. 다중 GPU 분산학습은 현재까지는 지원되지 않는 것으로 추정된다.

Coach는 자체적인 시각화 도구(Utility)를 제공하고 있으며, 개발 개념, API에 관한 준수한 문서화 및 튜토리얼(Tutorial)을 포함한다는 점에서 우수한 수준의 개발 편의성을 지닌 것으로 생각된다.

나. Dopamine

Dopamine[43,44]은 Google의 비공식 연구 프로젝트로써 2018년 공개되었다. 개발진들은 심층강화학습 활용 연구 시 쉽고 빠른 프로토타입을 제작할 수 있는 일반적인 라이브러리를 제작하는 것을 목표로 언급하였다.

참고문헌 [45]에 따르면, 2회의 개정을 거쳐 배포판 2.0이 발표되었고, 5인의 기여자가 총 114회의 변경을 수행하였으며, 최근까지 업데이트가 이어지고 있다. 현재 GitHub 별점은 8,388점이다. 동작 OS로 Ubuntu 및 Mac OS X, 프로그래밍 환경으로 Python 2.7 및 3, 기반 DNN 라이브러리로 TensorFlow 및 Keras[46]를 명시하고 있다. 제공하는 알고리즘은 4종(DQN, C51, IQN, Rainbow) 내외로 추정되며, OpenAI Gym 환경 중 이산(Discrete) 도메

6) 2019년 10월 1일 기준. 본 절의 다른 정보들도 동일한 일자를 기준으로 한다.

인 부분을 지원한다. 분산학습 및 다중 에이전트 개념은 현재까지는 지원되지 않는 것으로 추정된다.

자체 시각화 도구, 문서화 및 코드 주석을 제공하며, 프로그래밍 없이 설정(Configuration) 구문(Script)으로만 구현 가능하다는 점에서 개발 편의성 면에서도 장점을 지닌 것으로 생각된다. 반면, 프로토타입핑을 넘어서는 복잡한 수준의 프로젝트에 적용하는 경우 개발 난이도가 높아질 것으로 판단된다.

다. Keras-RL

Keras-RL[47]은 2016년 공개되었으며, 고계층(High-level) 심층기계학습 API인 Keras 사용자들에게 익숙한 형태의 자연스러운(Seamless) 심층강화학습 API를 제공하는 것을 목표로 제작되었다.

참고문헌 [47]에 따르면, 9회의 개정을 거쳐 배포판 0.4.2가 발표되었고, 36인의 기여자가 현재까지 총 307회의 변경을 수행하였으나, 최근에는 다른 라이브러리에 비해 업데이트가 둔화된 편이다. 현재 GitHub 별점은 4,165점이다. 동작 프로그래밍 환경 및 라이브러리는 Python 및 Keras를 명시하고 있다. 제공하는 알고리즘은 7종(DQN, Double DQN, Duel DQN, DDPG, NAF, CEM, Deep SARSA)이며, OpenAI Gym 및 PySC2 환경을 지원한다. 분산학습 및 다중 에이전트는 개념은 현재까지는 지원되지 않는 것으로 추정된다.

Keras-RL은 Keras의 확장판(Add-on)으로써 이해될 수 있으며, 고계층 API를 지향하기 때문에 습득이 용이하고, 개발 편의성이 높을 것으로 생각된다. 반면, API 확장 본연의 목표에 충실하며, 시각화와 같은 추가 도구는 제공하지 않는다.

라. OpenAI Baselines

OpenAI Baselines[48]는 OpenAI Gym 환경으로 유명한 OpenAI에서 제작한 심층강화학습 라이브러리로 2017년 공개되었다. 개발의 목표는 다양한 도메인에서 심층강화학습 활용의 기초가 되는 정책개선 알고리즘의 집합을 형성하는 것이다.

참고문헌 [48]에 따르면, 108인의 기여자가 현재까지 총 339회의 변경을 수행하였고, 최근까지 꾸준히 업데이트되고 있다. 현재 GitHub 별점은 8,513점이다. 동작 OS로 Ubuntu 및 Mac OS X, 프로그래밍 환경 및 라이브러리로 Python 3, OpenMPI[49], TensorFlow를 사용한다. 제공하는 알고리즘은 10종(A2C, ACER, ACKTR, DDPG, DQN, GAIL, HER, PPO, TRPO)이며, OpenAI Gym 환경을 지원한다. 분산학습 및 다중 에이전트 개념은 현재까지는 지원되지 않는 것으로 추정된다.

OpenAI Baselines는 문서화, 주석 및 예제를 풍부한 수준으로 제공하지 않는다는 점에서, 개발 프로젝트에 해당 라이브러리를 적용 시 코드의 수정 및 이식(Porting)을 위한 부담이 있을 것으로 판단된다. 또한, 시각화와 같은 추가 도구를 제공하지 않는다는 점에서 개발 편의성 면에서 개선의 여지가 있는 것으로 생각된다.

마. OpenAI Spinning Up

2018년에 발표된 OpenAI Spinning Up[50]은 OpenAI에서 제작하였으며, 초심자들의 교육 및 학습에 중점을 둔 심층강화학습 라이브러리이다. 개발진들은 사용자들에게 “심층강화학습의 이론이 어떻게 실제적 구현으로 이어지는가?”에 관한 통찰을 제공하는 것을 목표로 지향하고 있다. 이에 따라 심층강화학습을 이론적으로 설명한 문서들과 해당 문서들의 의사코드(Pseudo code)에 부합하는 라이브러리 및 예제를 함께 제공한다고 언급하였다.

참고문헌 [51]에 따르면, 2회의 개정을 거쳐 배

포판 0.1.1이 발표되었고, 21인의 기여자가 현재까지 총 82회의 변경을 수행하였으나, 최근에는 업데이트가 다소 둔화된 편이다. 현재 GitHub 별점은 3,431점이다. 동작 OS로 Ubuntu 및 Mac OS X, 프로그래밍 환경 및 라이브러리로 Python 3, OpenMPI, TensorFlow를 사용한다. 제공하는 알고리즘은 6종(PG, TRPO, PPO, DDPG, TD3, SAC)이며, OpenAI Gym 환경의 일부인 MuJoCo[52]를 지원한다. 분산학습 및 다중 에이전트 개념은 현재까지는 지원하지 않는 것으로 추정된다.

OpenAI Spinning Up은 교육 및 학습에 중점을 둔 라이브러리답게, 풍부한 문서와 예제, 복잡성을 제거한 직관적 소스코드에 의해 우수한 습득 난이도를 제공할 것으로 생각된다. 반면, 복잡한 개발 프로젝트에 적용하는 경우 라이브러리 내부의 수정 및 확장이 불가피할 것으로 예상된다.

바. RLlib

RLlib[53,54]은 UC Berkeley의 RISE Lab에서 제작한 심층강화학습 라이브러리로 2017년 발표되었다. RLlib은 RISE Lab 연구진들이 개발한 분산 심층기계학습 라이브러리인 Ray[55]를 기반으로 동작하며, 다수의 GPU 또는 노드 상에서 분산심층강화학습을 제공하는 것을 목표로 제작되었다. 개발진들은 복잡한 분산병렬 프로그래밍의 상세는 라이브러리 내부에 추상화(Abstraction)하고, *N*차 병렬화를 *N*회 반복문으로 표현하는 형태의 직관적 API를 제공함으로써 개발 난이도는 낮추되, 모듈화 및 재사용성은 높이는 것을 목표로 언급하였다.

참고문헌 [56]에 따르면, RLlib이 속한 Ray는 27회의 개정을 거쳐 배포판 0.7.5가 발표되었고, 현재 200인의 기여자가 3,304회의 변경을 수행하였으며, 최근까지 활발히 업데이트되고 있다. 현재 GitHub 별점은 8,801점이다. 동작 OS 및 프로그래밍 환경으로 Linux 및 Mac OS와 Python 2.7, 3.5, 3.6, 3.7이 테스트되었으며, 기반 DNN 라이브러리로 TensorFlow[32] 및 PyTorch[57]를 지원한다. 제공하는 알고리즘은 17종(A3C, PPO, IMPALA, DQN, Rainbow, APEX-DQN, APEX-DDPG, ES, QMIX, MARWIL 등)이며, OpenAI Gym 환경을 기본으로 제공하되, 다른 환경의 지원을 위한 기반 라이브러리를 제공하고 있다. 또한, CARLA 환경의 지원을 예제 형태로 포함하고 있다.

RLlib은 Ray를 기반으로 동작한다는 점에서, 태생적으로 분산심층강화학습을 주요한 특징으로 강조하고 있다. 참고문헌 [54]에 따르면, 512개 CPU 및 64개 GPU로 이루어진 클러스터상에서 준수한 수준의 병렬화 및 학습 성능을 보여준 바 있다. 또한, 계층화 및 다중 에이전트 환경을 위한 기능 역시 지원한다. 현재 RLlib은 현존하는 심층강화학습 라이브러리 중 가장 강력한 분산학습 기능을 제공하는 것으로 판단된다.

RLlib은 준수한 수준의 문서화와 풍부한 예제 및 자체 유틸리티를 제공하지만, Ray에 대한 사전경험이 필요하고, 소스코드 복잡도가 높다. 즉 습득 난이도가 높은 편으로, 개발 프로젝트 적용 시 다른 라이브러리에 비해 상대적으로 긴 개발 기간이 소요될 것으로 생각된다.

사. Stable Baselines

2018년 발표된 Stable Baselines[58]는 프랑스 INRIA와 ENSTA ParisTech에서 개발하였으며, OpenAI Baselines에서 분화된(Forked) 라이브러리이다.

개발진들은 기존 OpenAI Baselines의 자료구조, 소스코드, 문서화, 테스트 및 정책개선 알고리즘 전반에 이르는 품질 개선을 제작 목적으로 언

급하였다. 참고문헌 [59]에 따르면, 21회의 개정을 거쳐 배포판 2.8.0이 발표되었고, 77인의 기여자가 760회의 변경을 수행하였으며, 최근까지 활발히 업데이트되고 있다. 현재 GitHub 별점은 1,244점이다. 동작 OS로 Ubuntu, Mac OS X 및 Windows 10을 사용하고, 프로그래밍 환경 및 라이브러리로 Python 3.5 및 OpenMPI, TensorFlow를 사용한다. 제공하는 알고리즘은 12종(OpenAI Baselines 제공 알고리즘에 SAC 및 TD3 추가)이며, OpenAI Gym 환경을 지원한다. 분산학습 및 다중 에이전트 개념은 현재까지는 지원하지 않는 것으로 추정된다.

Stable Baselines는 OpenAI Baselines의 전반적 품질 개선을 목적으로 만들어졌기 때문에 준수한 형태의 문서화와 소스코드를 제공하고 있다. 또한, Stable Baselines로 훈련된 100여 개 이상의 에이전트들의 모음인 RL Baselines Zoo[60]를 제공함으로써 라이브러리 사용자의 편의성 증대를 도모하였다.

아. TensorForce

2017년 발표된 TensorForce[61]는 Google의 비공식 연구 프로젝트로써 심층강화학습 구현에 보편적으로 필요한 부분을 모듈화된 컴포넌트(Component)로 구현하여, 개발 시 정책개선 알고리즘 및 응용 종속성에 의해 발생하는 복잡도를 제거하는 것을 목적으로 개발되었다. 즉, 앞서 기술한 Coach와 유사한 지향성을 지닌 것으로 생각된다.

참고문헌 [62]에 따르면, 16회의 개정을 거쳐 배포판 0.5.1이 발표되었고, 현재 48인의 기여자가 1,739회의 변경을 수행하였으며, 최근까지 활발히 업데이트되고 있다. 현재 GitHub 별점은 2,447점이다. 동작 환경 및 기반 DNN 라이브러리로는 Python 3 및 TensorFlow 1.13을 명시하였다. 제공하는 알고리즘은 11종(DQN, Dueling DQN, Double DQN, DQfD, VPG, REINFORCE, PPO, A3C, TRPO, NAF, GAE) 이상이며, 7종의 환경(OpenAI Gym, Arcade Learning Environment[63], MazeExplorer[64], OpenAI Retro[65], OpenSim[66], PyGame[67], VizDoom[39])을 지원한다. 분산학습 개념 및 다중 에이전트는 현재까지는 지원하지 않는 것으로 추정된다.

TensorForce는 기본적으로 TensorFlow의 확장 개념으로 제작되어, 태생적으로 TensorFlow에서 제공하는 각종 유틸리티(예, Tensor-board)를 지원하며, 준수한 형태의 문서화를 제공하고 있다. 현재 소스코드 구조 및 각종 오류에 관한 개선작업(Major revision)이 활발히 진행 중으로, 향후 개발 편의성 증대를 기대할 수 있을 것으로 생각된다.

자. TF-Agents

TF-Agents[68]는 Google의 비공식 연구 프로젝트로써 2018년 발표되었으며, TensorForce와 유사한 목적의 라이브러리로 개발되었다.

참고문헌 [68]에 따르면, 현재 공식 배포판이 발표되지 않은 상태로 33인의 기여자가 702회의 변경을 수행하였으며, 최근까지 꾸준히 업데이트되고 있다. 현재 GitHub 별점은 899점이다. 동작 프로그래밍 환경 및 기반 DNN 라이브러리로 Python 및 TensorFlow를 명시하고 있다. 제공하는 알고리즘은 7종(DQN, DDQN, DDPG, TD3, REINFORCE, PPO, SAC)이며, OpenAI Gym 환경을 지원한다. 분산학습 및 다중 에이전트 개념은 지원하지 않는 것으로 추정된다.

TF-Agents는 초심자를 위한 튜토리얼을 제공하며, 준수한 모듈화 및 주석을 제공하는 것으로 판단된다. 현재 초기 개발단계임을 감안하면, 향후 향상된 개발 편의성을 제공할 수 있을 것으로 생각된다.

표 2 심층강화학습 라이브러리 비교

라이브러리	개발정보 †	알고리즘 다양성	환경 지원성	개발 편의성 및 분산학습 지원성
Coach*	배포판 1.0.0, 기여자 28인 변경 491회, 별점 1,470점	24종	8종	–안내서, API 문서, 튜토리얼 및 시각화 도구 제공 –다중 CPU 및 다중노드 지원
Dopamine	배포판 2.0, 기여자 5인 변경 114회, 별점 8,388점	4종	1종	–안내서, API 문서, 예제 및 시각화도구 제공
Keras–RL	배포판 0.4.2, 기여자 36인 변경 307회, 별점 4,165점	7종	2종	–안내서, API 문서 및 예제 제공
OpenAI Baselines	기여자 108인, 변경 339회 별점 8,513점	10종	1종	–안내서 제공
OpenAI Spinning Up	배포판 0.1.1, 기여자 21인 변경 82회, 별점 3,431점	6종	1종	–안내서, 학습자료, API 문서, 예제 및 연습문제 제공
RLlib*	배포판 0.7.5, 기여자 200인 변경 3,304회, 별점 8,801점	17종	2종	–안내서, API 문서, 튜토리얼, 예제 및 튜닝 (Tuning) 도구 제공 –다중 CPU, 다중 GPU 및 다중노드 지원
Stable Baselines	배포판 2.8.0, 기여자 77인 변경 760회, 별점 1,244점	12종	1종	–안내서, API 문서, 예제 및 훈련된 에이전트 모음 제공
TensorForce	배포판 0.5.1, 기여자 48인 변경 1,739회, 별점 2,447점	11종	7종	–안내서, API 문서, 예제 제공
TF–Agents	기여자 33인, 변경 702회 별점 899점	7종	1종	–안내서, 튜토리얼 및 예제 제공
TRFL	기여자 7인, 변경 62회 별점 2,740점	23종	1종	–안내서 및 API 문서 제공

† 2019년 10월 1일 기준

* 다중 에이전트 지원

차. TRFL

2018년 발표된 TRFL(Truffle)[69]은 심층강화학습 분야의 선도기업 중 하나로 알려진 DeepMind에서 제작된 라이브러리로, 심층강화학습을 위한 TensorFlow 기반의 모듈화된 빌딩 블록 제공을 위해 개발되었다. 즉 앞서 기술한 Coach, TensorForce 및 TF–Agents와 유사한 지향성을 지닌 것으로 생각된다.

참고문헌 [69]에 따르면, 7인의 기여자가 62회의 변경을 수행하였으며, 현재 기준으로 2019년 4월 이후로 추가 업데이트는 없는 상태이다. 현재 GitHub 별점은 2,740점이다. 동작 프로그래밍 환경 및 기반 DNN 라이브러리로 Python 및 TensorFlow를 명시하고 있다. 제공하는 알고리즘은 23종(DQN, Double DQN, SARSA, DDPG, A2C, A3C 등)이며, OpenAI Gym 환경을 지원한다. 분산학습 및 다중 에이전트 개념은 지원하지 않는 것으로 추정된다.

TRFL은 DeepMind 내부에서 사용되던 라이브러리를 공개한 것으로, Keras–RL과 같이 심층강화학습 API 제공이라는 본연의 목적에 집중하였다. 따라서, 심층강화학습 API 외에 다른 도구를 제공하지 않는다. 그러나 Keras–RL과 반대로 저계층 API의 형태를 지향하고 있다. 이는 라이브러리의 정밀한 사용이 가능하다는 장점이 있지만, 습득 난이도가 높아지는 원인이 된다. 이는 TRFL에서 제공하는 튜토리얼 및 풍부한 소스코드의 주석을 통해 보충할 수 있을 것이라 생각된다.

3. 라이브러리 간 비교분석

표 2는 이전 절에서 제시한 10종의 심층강화학습 라이브러리를 개발정보, 알고리즘 다양성, 환경 지원성, 개발 편의성, 분산학습 및 다중 에이전트 지원 측면에서 비교한 표이다. 프로젝트 기획 시 적합한 심층강화학습 라이브러리의 선택은 해당 프로젝트의 성격에 따라 달라진다. 이를 예시하면 다음과 같다.

- 자율주행 시뮬레이션 환경인 CARLA상에서의 심층강화학습 기반 주행 알고리즘 개발이 프로젝트의 목적이라면 해당 환경을 지원하는 Coach가 적합한 선택이 될 수 있다.
- 프로젝트 수행 인원들의 심층강화학습 활용 경험이 전무하며, 단기간 내 간단한 형태의 PoC 프로토타입을 제작 및 검증하는 것이 목적이라면, 습득이 용이한 OpenAI Spinning Up이나 Dopamine이 적합한 선택이 될 수 있다.
- 프로젝트 수행 기간이 길고, 수행 인원들이 심층강화학습 활용 경험이 있으며, 대규모 분산학습이 필요하다면 RLlib이 적합한 선택이 될 수 있다.

상기 제시한 10종의 라이브러리 외에도 심층강화학습 활용을 위한 다양한 라이브러리들이 개발되고 있다. 해당 내용은 참고문헌 [70–72]의 참조를 권한다.

Ⅳ. 결론

최근 심층강화학습은 게임, 로보틱스, 자율주행, 무인비행, 컴퓨팅 및 금융에 이르는 응용에 대해 혁신적 해법으로 등장하였다. 또한, 분산학습 및 다중 에이전트 등의 발전적 개념이 더해지면서 다양한 도메인에 심층강화학습을 활용하는 시도는 꾸준히 증가할 것으로 생각된다.

본 고에서는 심층강화학습 개념을 개괄적으로 소개하고, 근래 제안된 심층강화학습 라이브러리들을 개발 프로젝트와의 적합성 관점에서 요약 및 분석하였다. 주어진 프로젝트에 대한 심층강화학습 라이브러리와의 적합성은 목표 및 지향점, 개발 지속성, 알고리즘 다양성, 환경 지원성, 분산학습 지원, 다중 에이전트 지원 및 개발 편의성 등의 기준으로 추정해볼 수 있으며, 이를 바탕으로 가장 적합한 라이브러리를 선택하는 것은 프로젝트의 성공적 수행을 위해 꼭 필요한 과정이라고 생각된다. 본 고에서 소개한 라이브러리 및 분석기준이 독자들의 심층강화학습 활용 프로젝트의 기획 및 시작 단계에서 소정의 참고자료로 활용될 것을 기대한다.

용어해설

강화학습 제어 및 의사결정의 대상이 되는 환경 또는 시스템에서 에이전트가 관찰-행동-보상 프로세스(Process)를 반복하며, 누적한 경험을 통해 제어 및 의사결정의 전략을 점진적으로 발전시키는 기계학습 기법

심층신경망 입력층과 출력층 사이의 여러 개의 은닉층을 포함함으로써 깊이를 늘린 신경망

심층기계학습 심층신경망을 학습모델로 활용하는 기계학습 기법

심층강화학습 강화학습의 정책 또는 가치함수를 심층신경망으로 구성하는 기법

라이브러리 컴퓨터 프로그램 작성 시 미리 작성되어 부품 역할을 하는 프로그램 모듈(Module)들을 모아놓은 집합. 일반적으로 목적에 따라 그룹화된 API 및 안내서를 제공

약어 정리

ACER	Actor–Critic with Experience Replay
ACKTR	Actor–Critic with Kronecker–factored Trust Region
APEX	Advanced Prioritized EXperience replay
API	Application Programming Interface

A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
CEM	Cross-Entropy Method
CPU	Central Processing Unit
C51	Categorical 51
DDPG	Deep Deterministic Policy Gradient
DFP	Direct Future Prediction
DNN	Deep Neural Network
DP	Dynamic Programming
DQfD	Deep Q-learning from Demonstration
DQN	Deep Q-Network
ES	Evolution Strategies
GAE	Generalized Advantage Estimation
GAIL	Generative Adversarial Imitation Learning
GPU	Graphical Processing Unit
HAC	Hierarchical Actor Critic
HER	Hindsight Experience Replay
IMPALA	IMPortance weighted Actor-Learner Architecture
IQN	Implicit Quantile Network
LP	Linear Programming
LTS	Long-Term Support
MARWIL	Monotonic Advantage Re-Weighted Imitation Learning
MDP	Markov Decision Process
NAF	Normalized Advantage Functions
NLP	Non-Linear Programming
PG	Policy Gradient
PoC	Proof-of-Concept
PPO	Proximal Policy Optimization
QMIX	Q-MIXing network
SAC	Soft Actor-Critic
SARSA	State-Action-Reward-State-Action
SGD	Stochastic Gradient Descent
SW	SoftWare
TD3	Twin Delayed DDPG
TRPO	Trust Region Policy Optimization
UAV	Unmanned Aerial Vehicle
VPG	Vanilla Policy Gradient

참고문헌

[1] 장수영 외, "심층 강화학습 기술 동향," 전자통신동향분석 34권 제4호, 2019. 8, pp. 1-14.

[2] R.S. Sutton et al., *Reinforcement Learning: An Introduction, 2nd edition*, Cambridge, MA, USA: MIT Press, 2018.

[3] Y. LeCun et al., "Deep Learning," *Nature*, vol. 521, May 2015. pp. 436-444.

[4] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," arXiv:1312.5602, Dec. 2013.

[5] A.S. Polydoros et al., "Survey of Model-based Reinforcement Learning: Applications on Robotics," *J. Intell. Robotic Syst.*, vol. 86, no. 2, Mar. 2017, pp. 153-173.

[6] J. Hwangbo et al., "Control of a Quadrotor with Reinforcement Learning," arXiv:1707.5110, July 2017.

[7] J. Zhang et al., "Query-Efficient Imitation Learning for End-to-End Autonomous Driving," arXiv:1605.06450, May 2016.

[8] H. Mao et al., "Resource Management with Deep Reinforcement Learning," in *Proc. HotNets'16*, Atlanta, CA, USA, Nov. 2016, pp. 50-56.

[9] H. Mao et al., "Neural Adaptive Video Streaming with Pensieve," in *Proc. Conf. SIGCOMM'17*, Los Angeles, CA, USA, Aug. 2017, pp. 197-210.

[10] H. Mao et al., "Learning Scheduling Algorithms for Data Processing Clusters," arXiv:1810.01963, Oct. 2018.

[11] 김근영 외, "기계학습을 활용한 5G 통신 동향," 전자통신동향분석 31권 제5호, 2016.10, pp. 1-10.

[12] Y. Deng et al., "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 28, no. 3, March 2017, pp. 653-664.

[13] https://www.yna.co.kr/view/AKR20171018151400017?input=1179m

[14] V. Mnih et al., "Asynchronous Methods for Deep Reinforcement Learning," in *Proc. Int Conf. Machine Learning*, New York, USA, June 2016, pp. 1928-1937.

[15] T.P. Lillicrap et al., "Continuous Control with Deep Reinforcement Learning," arXiv:1509:02971, Sept. 2015.

[16] J. Schulman et al., "Trust Region Policy Optimization," in *Proc.*

Int. Conf. Machin Learning, Lille, France, July 2015, pp. 1889-1897.

[17] J. Schulman et al., "Proximal Policy Optimization Algorithms," arXiv:1707.06347, Jul. 2017.

[18] T. Schaul et al., "Prioritized Experience Replay," arXiv:1511.05952, Nov. 2015.

[19] Z. Wang et al., "Dueling Network Architectures for Deep Reinforcement Learning," in *Proc. Int Conf. Machine Learning*, New York, USA, June 2016, pp. 1995-2003.

[20] H. Hasselt et al., "Deep Reinforcement Learning with Double Q-Learning," in *Proc. AAAI Conf. Artif. Intell.*, Fhoenix, AZ, USA, Feb. 2016, pp. 2094-2100.

[21] 오일석, 패턴인식, 교보문고, 2008년.

[22] https://hunkim.github.io/ml/

[23] I. Goodfellow et al., *Deep Learning*, MIT Press, 2016.

[24] L. Espeholt et al., "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures," in *Proc. Int. Conf. Machine Learning*, Stockholm, Sweden, July 2018, pp. 1407-1416.

[25] D. Horgan et al., "Distributed Prioritized Experienced Replay," arXiv:1803.00933, March 2018.

[26] S. Kapturowski et al., "Recurrent Experience Replay in Distributed Reinforcement Learning," in *Proc. Int. Conf. Machine Learning*, Long Beach, CA, USA, May 2019.

[27] R. Lowe et al., "Multi-Agent Actor Critic for Mixed Cooperative-Competitive Environments," arXiv:1706.02275, July 2017.

[28] T. Rashid et al., "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," in *Proc. Int. Conf. Machine Learning*, Stockholm, Sweden, July 2018, pp. 4295-4304.

[29] S. Li et al., "Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient," in *Proc. AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, Jan. 2019.

[30] https://nervanasystems.github.io/coach/

[31] https://github.com/NervanaSystems/coach

[32] https://www.tensorflow.org/?hl=ko

[33] https://mxnet.incubator.apache.org/

[34] https://software.intel.com/en-us/frameworks/tensorflow

[35] https://gym.openai.com/

[36] https://github.com/openai/roboschool

[37] https://github.com/Breakend/gym-extensions

[38] https://github.com/bulletphysics/bullet3

[39] http://vizdoom.cs.put.edu.pl/

[40] http://carla.org/

[41] https://github.com/deepmind/pysc2

[42] https://github.com/deepmind/dm_control

[43] https://opensource.google/projects/dopamine

[44] P.S. Castro et al., "Dopamine: A Research Framework for Deep Reinforcement Learning," arXiv:1812.06110, Dec. 2018.

[45] https://github.com/google/dopamine

[46] https://keras.io/

[47] https://github.com/keras-rl/keras-rl

[48] https://github.com/openai/baselines

[49] tps://www.open-mpi.org

[50] https://spinningup.openai.com/en/latest/

[51] https://github.com/openai/spinningup

[52] https://gym.openai.com/envs/#mujoco

[53] https://ray.readthedocs.io/en/latest/rllib.html

[54] E. Liang et al., "RLlib: Abstractions for Distributed Reinforcement Learning," in *Proc. Int. Conf. Machine Learning*, Stockholm, Sweden, July 2018, pp. 3053-3062.

[55] https://ray.readthedocs.io/en/latest/index.html#

[56] https://github.com/ray-project/ray

[57] https://pytorch.org/

[58] https://stable-baselines.readthedocs.io/en/master/

[59] https://github.com/hill-a/stable-baselines

[60] https://github.com/araffin/rl-baselines-zoo

[61] https://tensorforce.readthedocs.io/en/latest/

[62] https://github.com/tensorforce/tensorforce

[63] https://github.com/mgbellemare/Arcade-Learning-Environment

[64] https://github.com/microsoft/MazeExplorer

[65] https://github.com/openai/retro

[66] https://opensim.stanford.edu

[67] https://github.com/ntasfi/PyGame-Learning-Environment

[68] https://github.com/tensorflow/agents

[69] https://github.com/deepmind/trfl

[70] https://winderresearch.com/a-comparison-of-reinforcement-learning-frameworks-dopamine-rllib-keras-rl-coach-trfl-tensorforce-coach-and-more/

[71] https://medium.com/@vermashresth/a-primer-on-deep-reinforcement-learning-frameworks-part-1-6c9ab6a0f555

[72] https://mc.ai/choosing-a-deep-reinforcement-learning-library/