ML 라이브러리와 언어 선택

여러 가지 선택 사항이 있는데 이를 잘 결정해야 한다.

- DNN 라이브러리
  - tensorflow + keras vs. pytorch
- 언어
  - c# vs. c++ vs. python

위 두 가지 선택이 가장 크다. 이를 자세히 살펴서 정리하고 결정한다.

DNN 라이브러리

tensorflow + keras 조합에서 tensorflow는 c++, .net 포팅이 있다. keras는 근본적으로 python 라이브러리로 python 바인딩을 통한 포팅만 .net에 제공되고 있다.

pytorch는 torchsharp이 libtorch c++ 라이브러리의 wrapper로 구현되어 있다. pytorch는 keras와 같은 상위 레벨 API를 자체적으로 포함하고 있어 c++, .net 환경에서 거의 동일한 기능을 제공하고 있다.

기능 상의 차이는 거의 없어 보이며 tensorflow 2.0이 pytorch의 영향을 크게 받은 형태로 맞췄기 때문에 사용성에서 두 가지 라이브러리는 동일하다고 본다. (세부에서는 차이가 있을 것이다)

https://data-newbie.tistory.com/425

여러 측면에서 잘 비교해 놓았다.

keras가 있어야만 tensorflow를 사용하기 쉬운데 python 밖으로는 포팅이 잘 되어 있지 않다는 점이 가장 큰 걸림돌이 된다.

언어

그간 C#과 .net에 투자하고 경험을 축적한 부분이 크다. 따라서, C#으로 잘 사용할 수 있는 DRL 라이브러리 구축이 가능하다면 그 편이 제일 좋다.

DRL은 다른 ML처럼 python이 가장 편하다. 하지만 파이썬은 봇의 다른 기능 처리는 느리다. 특히, 통신이나 recastdetour나 bullet physics와 같은 외부 라이브러리 연동이 안 되어 있거나 너무 pythonic 한 경우가 많다.

위 두 가지를 고려하면 C#으로 직접 쉽게 사용하고 활용 가능한 라이브러리가 있다면 좋은데 torchsharp이 그 기능을 제공하는 것으로 보인다. 또한 c++에서도 동일한 방식으로 사용 가능하다.

https://reagent.ai/ 와 같은 pytorch 기반의 RL 프레임워크도 당연히 있고 raylib에서도 pytorch 구현을 제공한다. 근본적으로 tensorflow+keras와 pytorch가 매우 유사한 인터페이스를 제공한다는 점은 편리한 쪽을 선택할 수 있는 근거가 된다.

바인딩 구현

pytorch의 c# 바인딩은 scisharp의 torch.net과 torchsharp 두 가지가 보인다. scisharp은 python 바인딩을 통해 pytorch도 지원한다 (keras처럼). 따라서, torchsharp을 사용해야 한다. 그 이유는 python 바인딩을 사용할 경우 python interpreter가 하나만 생성되어 멀티쓰레드 환경에서 사용할 경우 여러 문제가 발생할 수 있고 파이썬을 그대로 사용하는 것과 동일하므로 .net과 c#의 장점을 최대한 활용하기 어렵

기 때문이다.

따라서, torchsharp을 fork 하여 참여하고 torchsharp으로 진행한다.

파이썬에서 벗어남

언어를 바꾸게 되면 raylib나 다른 파이썬 라이브러리 구현을 해당 언어로 직접 구현해야 하는 부담이 생긴다. 이런 부담은 궁극적으로는 장점이 된다. 왜냐하면 직접 구현하는 과정에서 이해가 깊어지고 활용 능력도 극대화되기 때문이다. 처음에는 힘들겠지만 시간이 지나면 훨씬 강력한 도구를 얻게 된다.

이런 방향은 언리얼 엔진에 DRL이 가능한 C++로 구현한 플러그인을 개발하고, 블루프린트와 연동하여 학습의 세부를 조절할 수 있는 능력을 확보하게 된다. 다른 많은 장점이 있고 그런 방향의 진행을 pytorch가 가능하게 한다.

점진적이고 효율적으로 진행하기 위해 분산 학습에서 learner는 python으로 처리할 수 있다. 따라서, 분산 학습의 환경은 여러 언어를 수용 가능하도록 아키텍처를 유지해야 한다.

평가

써봐야 알 수 있다. keras는 c#에서 python 바인딩 외에 찾을 수 없으므로 pytorch의 c# 버전으로 MNIST 분류 문제를 구현한 코드를 실행해 보고 어려운 점은 없는 지 확인한다.

torchsharp의 개발자들이 원래 pytorch의 wrapper로만 유지하도록 최대한 노력하고 있어 아마도 코드는 python, c++, c#이 거의 같을 것으로 보이고, 이 점이 pytorch의 가장 큰 장점이다.