**<Playing Atari with Deep Reinforcement Learning 논문 리뷰>**

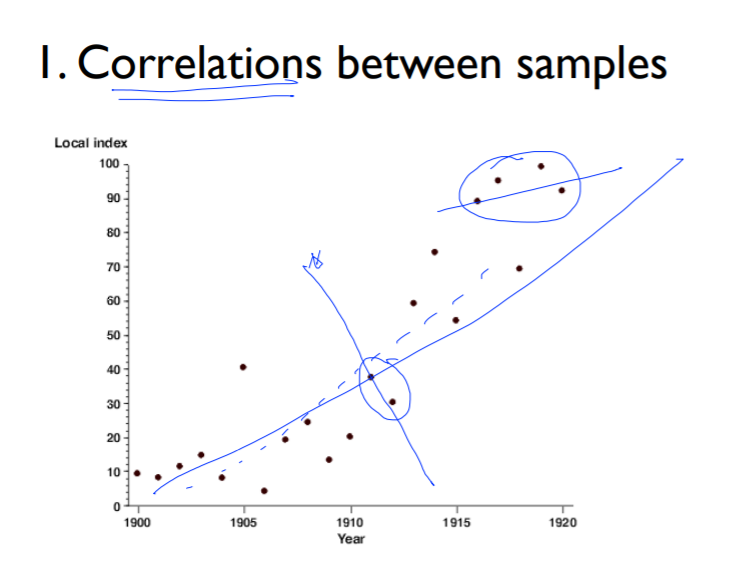
* **Abstract** 
  + 강화학습을 이용한 첫번째 딥러닝 모델이며, high-dimensional sensory input인 pixel을 직접 받아 처리한다. ­­
  + CNN 모델을 사용하며, input은 raw pixel이고, output은 value function이다.
  + 게임마다 network는 바꾸지 않은 채 Atari 2600에서 7개의 게임에 적용해보았는데, 3개의 게임에서는 사람을 능가했다.
* **1. Introduction: 풀고자 하는 문제 (Challenge)**
  + high-dimensional sensory input을 direct로 입력 받고 싶었으나 기존까지는 어려웠다.
  + 대부분의 RL(reinforcement learning)은 hand-crafted features에 의존해야 했다.
* CNN, Multilayer perceptron 등 Neural Network를 통해 문제를 해결하고자 함.

but, 대부분의 DL(deep learning)은 hand-labelled된 데이터를 사용하지만, RL은 그렇지 못하다. 또한, DL은 독립적인 데이터를 가정하는데, RL은 sequence로서 correlated data를 사용한다.

* 해결책:

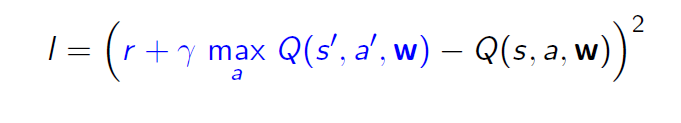
1. hand-labelled data -> Q-learning
2. correlated data -> experience replay

\* correlated data일 경우, 아래 그림과 같이 최적의 선과 다르게 학습될 수 있음.



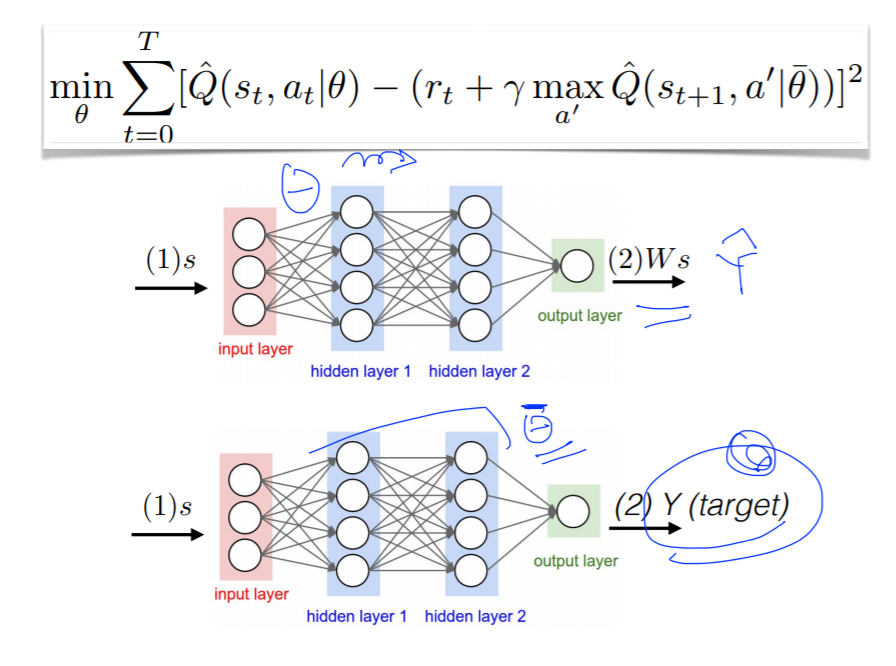
* **2. Background**
  + State: sequences of actions and observations
  + Value function: Bellman equation으로부터 Q(s,a) 정의
  + Q-Network:

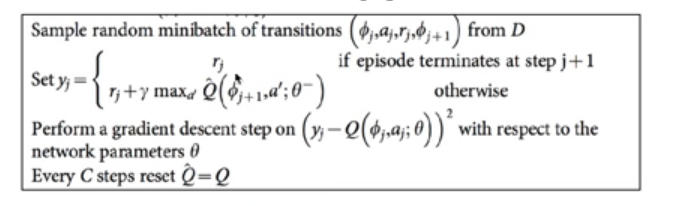
1. 기존의 Q-table로 표현하기에는 상태공간이 크기 때문에 network로 표현한다.
2. Q(s,a,**w(weight)**)가 optimal한 Q(s,a)(=target 또는 y)와 근사하도록 하는 것이 목표이다. Stochastic gradient descent를 통해 loss function을 최소화하는 방향으로 weight를 학습한다.



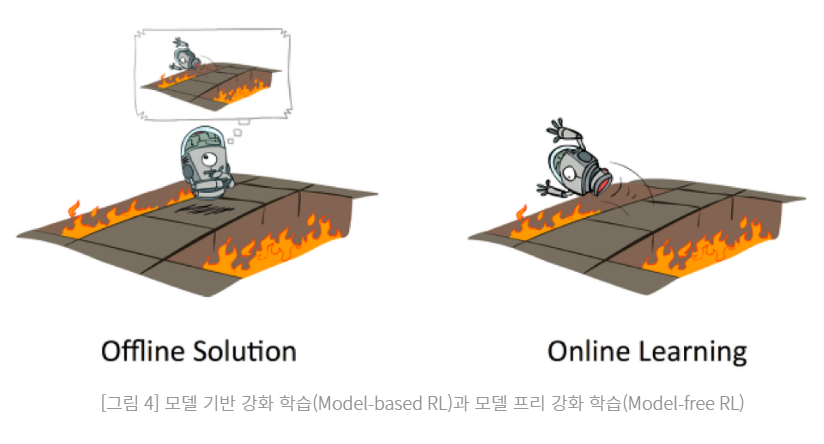
\* 참고로, Q-table에서는 optimal한 Q(s,a)를 찾을 수 있었던 것에 반해, neural net을 사용할 경우에는 수렴이 안되었었는데, Experience Replay를 통해 해결하였다.

\* Non-stationary targets: target과 Q(s,a,w)가 같은 네트워크를 사용할 경우, weight이 변경이 될 때 target도 같이 변화한다는 문제가 발생한다. (활을 쏘는데, 과녁도 같이 움직인다고 생각하면 됨) -> 네트워크 분리 (target network 생성)





* + model-free algorithm: 가치함수 계산 시, 샘플링 기법 (모델 기반 강화학습과 달리 환경이 어떻게 동작되는지 알지 못한다. 즉, 주어진 상태에서 에이전트는 어떤 행동을 하고, ‘수동적으로’ 환경을 알려주는 다음 번 상태의 보상을 얻게 된다.)



* + off-policy: behavior distribution(행동정책)은 e-greedy를 사용하고, 타깃정책은 greedy 로, 다음 상태에서 취할 행동 중 행동가치가 가장 큰 행동을 선택한다.

\* 현재 행동을 선택하는 행동정책과 가치함수를 학습하기 위해 다음 상태에서 행동을 선택하는 타깃정책이 다른 것을 off-policy라고 한다.

\* e-greedy: 행동이 선택될 확률을 epsilon으로 조정한다. 최적 행동 외에 모든 행동이 확률적으로 선택될 가능성이 있다. (-> exploration)

* **3. Related Work**
  + TD-gammon:

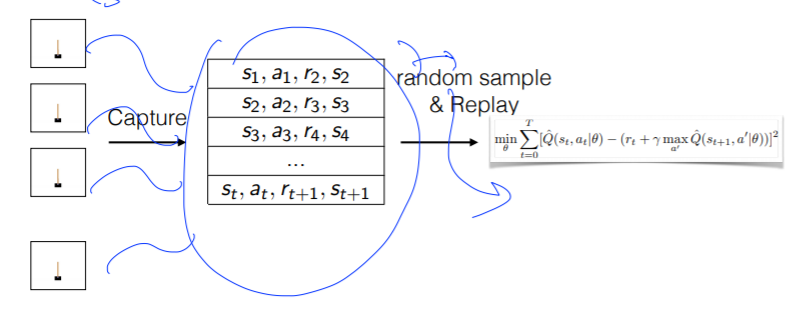
1. Q-learning과 유사한 model-free algorithm
2. value function 계산하는데 multi-layer perceptron 사용한다.
3. backgammon에서만 적용 가능하다
4. Q-learning과 같은 model-free 알고리즘을 non-linear function approximator나 off-policy learning에 적용하면 Q-network는 수렴하지 않는다.
   * Deep neural network와 RL 결합:
5. divergence issue: gradient temporal-difference의 경우 non-linear function approximator로 fixed policy를 사용할 때 또는 제한적인 Q-learning 변형으로 linear function approximation과 함께 control policy를 학습할 때 수렴함을 증명하였다.
6. 그러나 nonlinear control까지는 확장되지 못하였다.
   * neural fitted Q-learning (NFQ):
7. RPROP gradient descent algorithm을 사용해 Q-network의 파라미터를 업데이트함으로써 loss function을 최적화시킨다.
8. batch update: 데이터셋 사이즈에 비례하여 연산량 발생. 반면, 해당 논문은 stochastic gradient update로 연산량을 줄였다.
9. deep autoencoder: task의 low dimensional representation을 학습하여 간단한 real-world control task에 성공적으로 적용시켰다. 반면 해당 논문은 end-to-end learning으로 visual input을 직접 사용하였다.

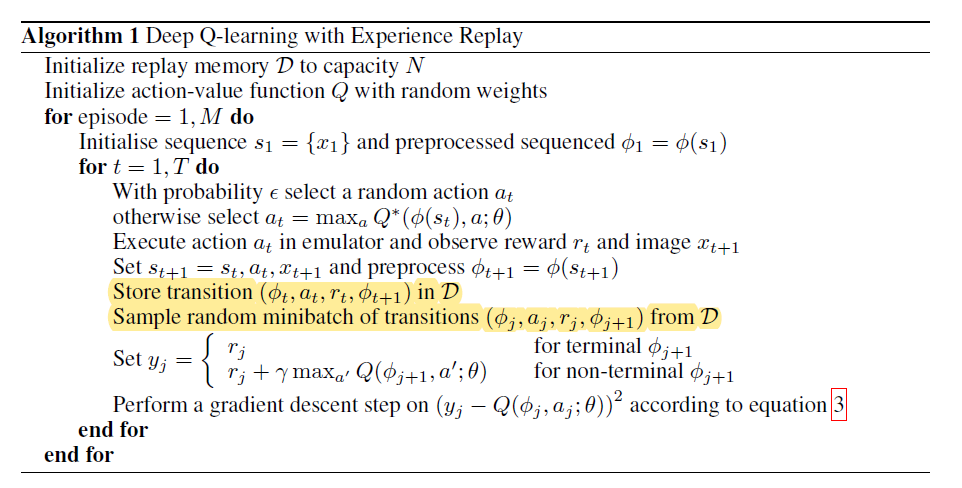
* **4. Deep Reinforcement Learning**
  + RL을 deep neural network와 연결해 RGB 이미지를 직접 다루고, stochastic gradient update를 사용해 데이터를 효율적으로 처리하는 것이 이 논문의 목표이다.
  + 적용된 기법: Experience Replay

1. Agent가 매 time-step마다 했던 experience(episode)들을 replay memory에 저장한다.

\* 전처리 함수 ф를 사용하여 고정 길이의 history(state)를 입력으로 받는다.

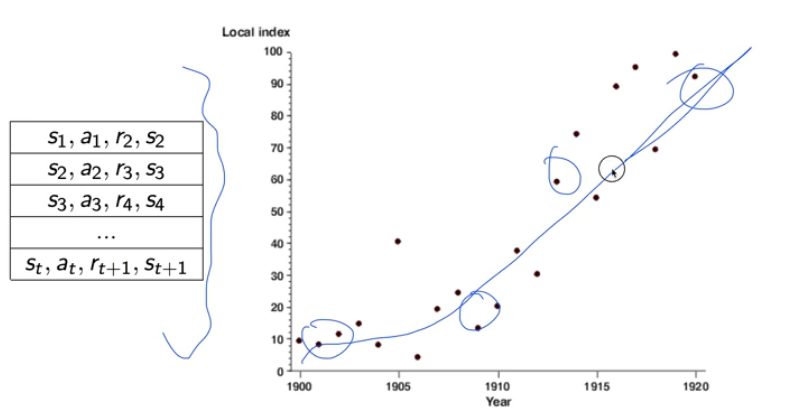
1. replay memory로부터 랜덤하게 하나를 추출해 학습에 적용시킨다.
2. experience replay 이후 agent는 e-greedy에 따라 행동을 선택하고 수행한다.





* 장점:

1. data efficiency: each step의 episode가 잠재적으로 많은 weight update에 재사용되므로 효율적이다.
2. break correlations: 연속적인 sample로부터 학습을 진행하는 것은 데이터들 간의 high correlation 때문에 비효율적이다. 에피소드의 순서를 무작위로 하여 데이터들 간의 상관관계를 break함으로써 update의 variance를 줄인다.



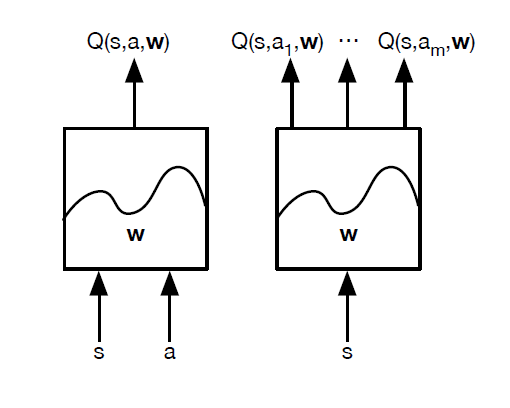
1. avoiding oscillations or divergence: off-policy 사용 전제

\* 개선점: replay memory buffer는 transition(episode) 간에 차별성을 두지 않는다. 즉, buffer는 N개의 제한된 크기를 갖고 있으므로, 최근의 transition으로 덮어씌우는 한계가 있다. 정교한 sampling을 위해서는 중요한 에피소드일 경우 우선순위를 매길 필요가 있다.

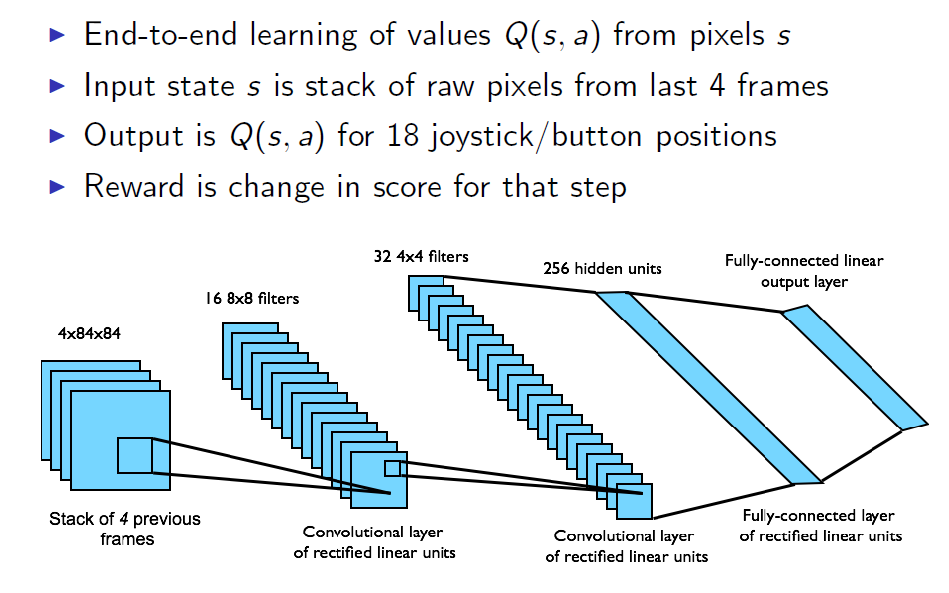
* **4-1. Preprocessing and Model Architecture**
  + 210x160 pixel, 128 color palette인 image를 사용하는 것은 많은 연산량이 요구되므로, input의 dimensionality를 줄이기 위해 basic preprocessing step 과정을 거친다.
  + RGB -> gray-scale로 변환 & 110x84로 down-sampling한 후 84x84로 cropping(GPU연산 위해)
  + 전처리 함수 ф에서 history의 마지막 4개의 frame만 전처리하여 stack에 두고 input 대한 Q-function을 구하기 위해 사용한다.
  + Q-value를 구하는 방법

1. (오른쪽 그림) history와 action을 input으로 하여 Q-value 예측
2. (왼쪽 그림) history만을 input으로 하여 각 행동에 대한 Q-value 예측

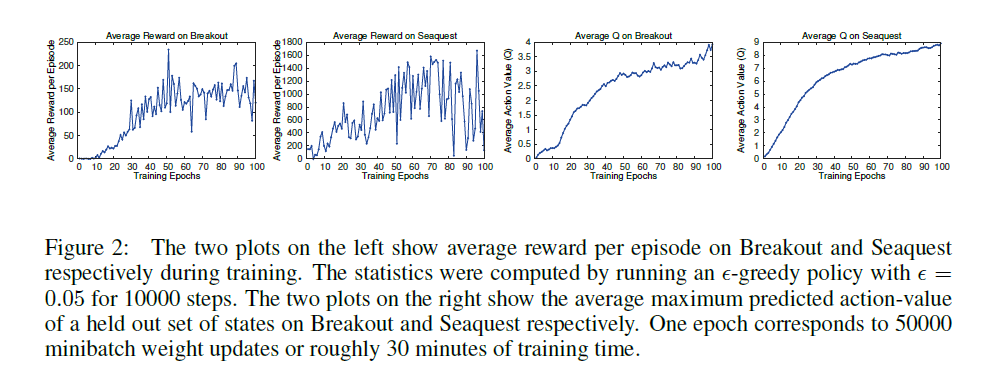
(single forward pass 이점 때문에 해당 논문에서는 두번째 방법을 사용하였다. 참고로, 첫번째 방법을 사용할 경우, 들어온 action에 따라 연산의 양도 linear하게 증가하게 된다.)



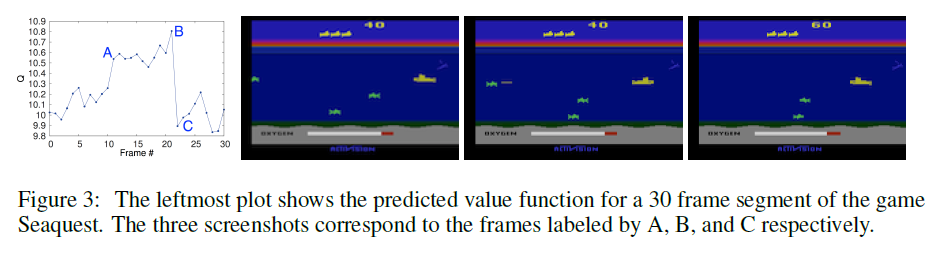
* + DQN in Atari



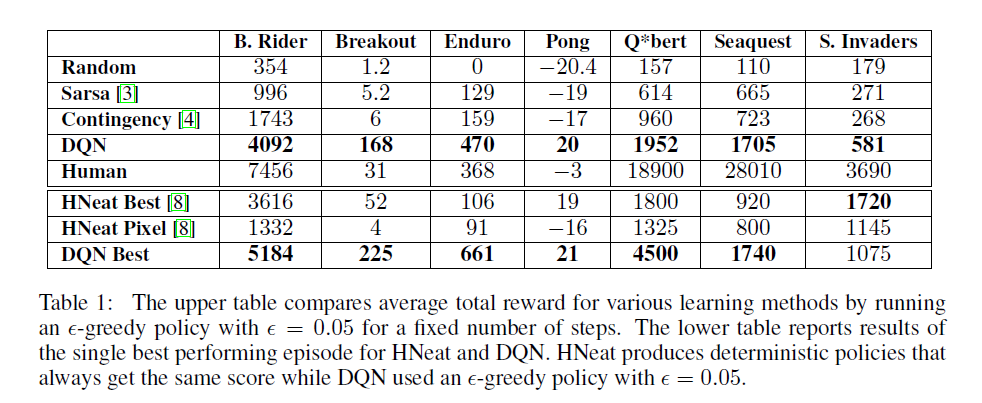
* **5. Experiments**
  + 6개의 Atari games에 대해 실험했고, 이 때 같은 network architecture, learning algorithm, hyperparameters setting을 사용했다. 즉, 다양한 게임에서도 robust하게 작동한다는 것을 보여준다.
  + training을 하면서 reward structure에 한가지 변화를 주었는데, score의 scale에 있어 양의 보상은 1, 음의 보상은 -1, 변화없음은 0으로 고정하였다. 이를 통해 error derivatives의 스케일을 제한하고 모든 게임에 동일한 learning rate를 사용하기에 용이했다.
  + minibatch size가 32인 RMSProp algorithm 사용
  + 행동 정책에서 e-greedy를 할 때, 100만번째 frame까지는 1에서 0.1까지 동일한 비율로 감소하는 epsilon 값을 사용했고, 이후에는 0.1로 고정했다.
  + frame-skipping technique 사용: agent가 모든 frame을 보는 것이 아니라 k번째 frame을 보고 액션을 선택한다. 그리고 마지막 행동은 skipped frames에 반복 적용시킨다. 이를 통해 k배 더 많이 게임을 진행시킬 수 있었다.
* **5-1. Training and Stability** 
  + Supervised learning과 달리 RL은 agent의 progress를 평가하는 것이 어렵다. 해당 논문에서는 평가의 척도로 average total reward를 사용한다.
  + 학습을 진행하면서 policy의 weight을 작게 변화시키더라도 state의 distribution에는 큰 변화를 미치므로 average total reward값은 변동이 크다.
  + 이론적인 증명은 부족하지만, Q-network를 사용해도 Q(s,a)가 수렴한다는 것을 Figure 2를 통해 보여준다.



* **5-2. Visualizing the Value Function** 
  + 게임이 진행되는 화면 A,B,C에 따라 value function이 잘 반영하고 있는 것을 Figure3를 통해 보여준다.



* **5-3. Main Evaluation**
  + DQN의 성능이 다른 알고리즘보다 좋았음을 Tabel1을 통해 보여준다. (Space Invaders 제외)
  + Breakout, Enduro, Pong 게임에서는 사람보다 잘했다.



* **6. Conclusion**
  + RL을 위한 새로운 deep learning model을 소개하였다.
  + raw pixels input만으로 어려운 control policy를 학습하였다.
  + SGD와 experience replay를 적용한 Q-learning의 변형을 보였다.
  + architecture 또는 hyperparameters의 변화없이 7개의 게임 중 6개의 게임에서 뛰어난 성능을 보여주었다.
* 참고
  + Sung Kim, 유튜브 강의

<https://www.youtube.com/watch?v=V7_cNTfm2i8&feature=youtu.be>

<https://www.youtube.com/watch?v=S1Y9eys2bdg&t=358s>

* + David Silver, Tutorial: Deep Reinforcement Learning

<https://icml.cc/2016/tutorials/deep_rl_tutorial.pdf>

* + Model-free based RL

<https://brunch.co.kr/@kakao-it/73>

[기초부터 시작하는 강화학습/신경망 알고리즘] 도서

* + experience replay

<https://poqw.github.io/DQN/>

* + 논문 번역

<https://mangkyu.tistory.com/60>