카테고리 없음

[Euron] 10week_ Rainbow: Combining Improvements in...

yejjji 2025. 5. 12. 12:17

0. Abstract

그동안 DQN에 대한 다양한 개선 기법들이 제안되어 왔지만, 어떤 조합이 효과적인지는 불분명하였다.
본 논문은 **DQN의 6가지 대표 확장 기법을 선택하여 이를 통합한 Rainbow 에이전트를 제안한다.**

실험을 통해, 이 통합된 기법이 Atari 2600 벤치마크에서 매우 높은 성능을 보이며,
데이터 효율성과 최종 성능 정확도 모두 향상된다는 사실을 입증하였다.
또한 Ablation Study를 통해 각 구성요소의 개별적인 성능 기여도도 체계적으로 분석하였다.

1. Introduction

강화학습의 다양한 최신 성공 사례들은 DQN 알고리즘에서 시작되었다.
DQN은 Q-learning, 합성곱 신경망(CNN), 경험 재연를 결합하여 픽셀 이미지로
학습하여 인간 수준의 Atari 게임 수행을 가능하게 하였다.
이후, 속도나 안정성을 향상시키는 다양한 DQN 확장 기법들이 제안되었다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

1. Double DQN (DDQN)
Q-learning의 과도한 가치 추정(overestimation bias) 문제를 해결한다.
Bootstrap 대상 행동 선택과 평가를 분리하여 오차를 감소시킨다.

2. Prioritized Experience Replay
학습 가치가 높은 transition을 더 자주 리플레이하여 최종적으로 데이터 효율성이 향상 가능하게 한다.

3. Dueling Network Architecture
상태 가치와 행동 우위를 분리하여 표현하여 행동 간의 일반화 능력을 향상시킨다.

4. Multi-step Learning
n-step Bootstrap으로 보상 전파 속도를 향상시킨다.
Bias-variance trade-off 조절에 효과적이다.

5. Distributional Q-learning
보상의 평균값만 추정 대신에 discounted된 return 전체를 학습한다.
이를 통해 보다 더 정교한 예측이 가능하다.

6. Noisy DQN
확률적 네트워크를 통해 탐색을 수행한다,
greedy 과정 없이도 효과적으로 행동 다양성 확보가 가능하다.

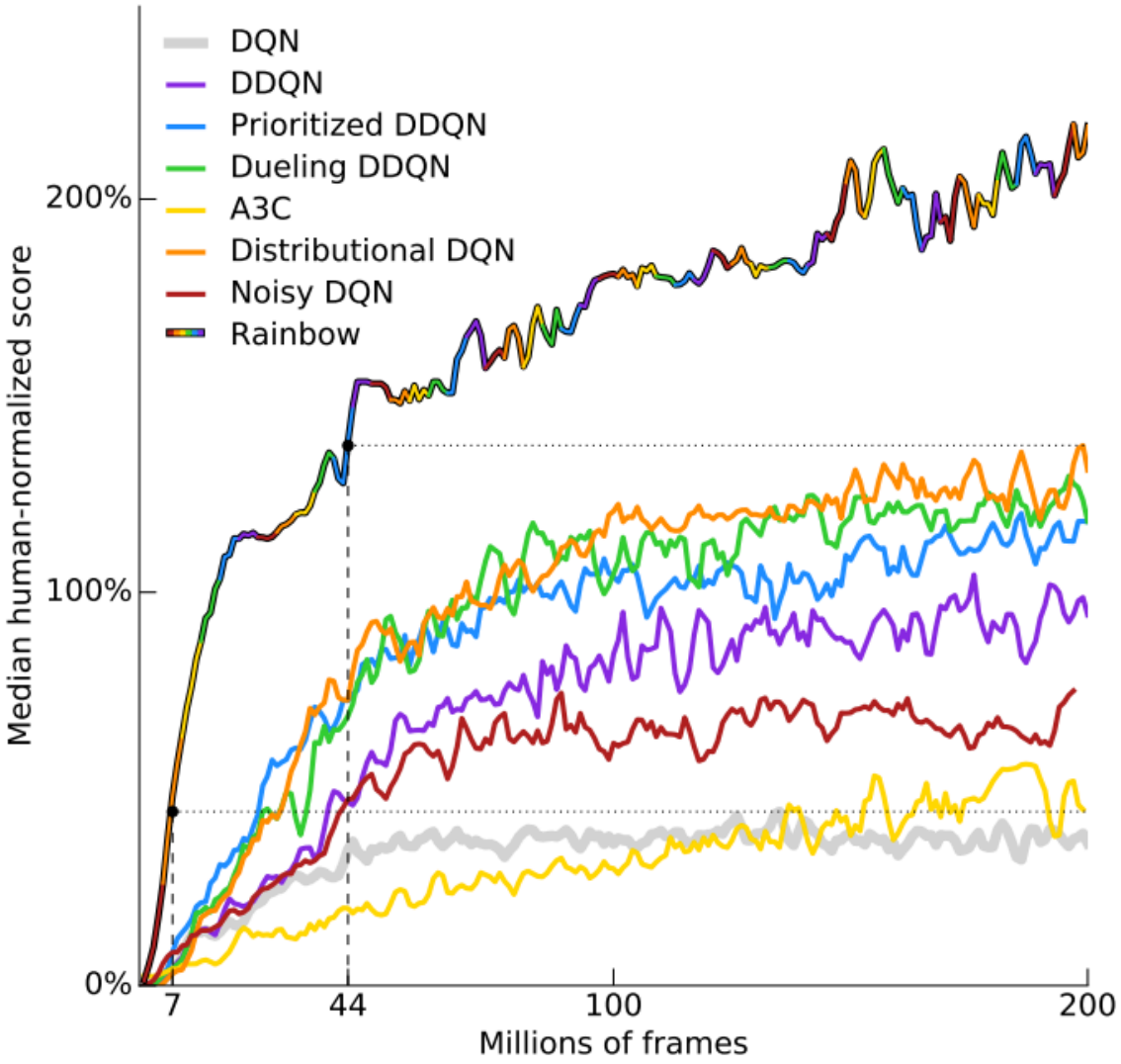
위 6가지의 기법들은 각기 다른 문제들을 해결하고 모두 공통된 DQN 프레임워크를 기반으로 한다.

이러한 이유로 인해 **6가지 기법들을 상호 보완적으로 통합되어 사용될 가능성이 다.**

메뉴바를 클릭하면
계정을 관리할 수 있어요

실제로 일부 기법들은 이미 결합되어 사용된다.

본 논문은 위 6가지 기법들을 모두 결합한 형태의 에이전트를 설계하고자 한다.



Rainbow와 다른 모델들의 성능 비교

실제 실험 결과 57개의 Atari 2600 게임에서 데이터 효율성과 최종 성능 측면 모두에서 최신의 최고 성능을 달성하였다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

2. Background

강화학습은 명시적인 정답 없이 Reward 만으로 에이전트가 알아서 환경 내에서 최적의 행동 방식을 학습하는 방법이다.
즉, 어떤 행동이 최선인지 감독 없이 스스로 시도하고 평가하면서 배워야 한다.

Agents and environments

에이전트를 환경에서 상호작용을 통해 행동, 다음 상태를 반환한다.
상호작용은 MDP라는 마르코프 결정 과정으로 형식화된다.

$$\langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$$

다음과 같이 형식화되는데, 환경이 agent에게 상태 St를 제공하고, 에이전트가 그에 따라 행동 At를 선택하며, 환경이 다음 보상인 Rt+1, 할인율 rt+1, 다음 상태 St_1를 반환하는 꼴이다.

MDP는 에피소드 형태로서 rt = r로 상수값을 일반적으로 가지나, 에피소드가 종결되는 경우에만 rt = 0로 설정된다.

에이전트는 정책 파이를 활용하여 각 상태에 대해 행동이 발생할 확률을 정한다.
정책을 통해 상태에서 행동을 선택할 때

$$G_t = \sum_{k=0}^{\infty} \gamma_t^{(k)} R_{t+k+1}$$

누적된 할인 보상값

의 기댓값을 최대화하는 정책을 찾는다.

Policy는 크게 직접 학습하는 경우와 다른 이미 학습된 값들의 식으로 학습하는 경우로 나뉜다.
이러한 Policy는 Exploration이라는 형태를 가능하게 하지만, 장기적인 전략을 발견하기에는 어렵다는 단점이 존재한다.

Deep reinforcement learning and DQN

상태 공간이 매우 크거나 연속적인 경우, Q값을 테이블 형식으로 저장하는 것은 [illegible]
다.
이로 인해 **깊은 강화학습에서는 신경망을 통해 가치 함수나 정책 함수를 근사한[illegible]**

메뉴바를 클릭하면
계정을 관리할 수 있어요

DQN는 강화학습과 합성곱 신경망을 결합하여, 픽셀 이미지 입력만으로도 Atari 게임을 인간 수준으로 학습이 가능하게 한다.
입력값으로는 최근의 프레임 스택값을, 출력값으로는 각 행동에 대한 Q값을 출력한다.

먼저, greedy 행동 선택 과정에서 현재 상태의 Q값에 기반하여 행동을 결정한다.
다음으로, (St,At,Rt+1,γt+1,St+1) 전이를 메모리에 저장하고, 무작위 샘플링 과정으로 과적합을 방지한다.
네트워크 파라미터인 seta를 일정 주기로 복사하여 고정된 Q 타깃값을 생성하고,
최종적으로

$$\mathcal{L} = \left(R_{t+1} + \gamma_{t+1} \max_{a'} q_\theta(S_{t+1}, a') - q_\theta(S_t, A_t) \right)^2$$

손실함수

로 계산된 손실을 mini-batch로 평균화하고 RMSprop optimizer로 업데이트시킨다.

이러한 구조는 학습 안정성 확보에 큰 영향을 끼친다.

3. Extensions to DQN

DQN(Deep Q-Network)은 강화학습의 주요한 이정표로 여겨지지만, 몇 가지 한계점이 알려지면서 이를 보완한 다양한 확장 기법들이 제안되었다. 여러 문제점들을 해결하기 위한 총 7가지의 대표적인 확장 기법을 소개한다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

1. Double Q-learning

문제점: 기존 Q-learning은 최대값을 취하는 과정에서 **과도한 가치 추정(overestimation bias)** 문제가 발생함

해결책: **행동 선택(action selection)**과 **행동 평가(action evaluation)**를 분리하여 과대 추정을 줄임

$$(R_{t+1}+\gamma_{t+1}q_{\overline{\theta}}(S_{t+1},\operatorname*{argmax}_{a'}q_\theta(S_{t+1},a'))-q_\theta(S_t,A_t))^2$$

DQN에 적용되는 형태

이를 통해 과도한 Q값 추정을 줄여 안정적인 학습이 유도 가능하다.

2. Prioritized Experience Replay

문제점: 기존의 DQN은 리플레이 버퍼에서 샘플을 **무작위로 균등하게** 선택함

해결책: TD 오차 값이 더 큰 transition을 더 자주 선택하도록 우선순위를 부여함

$$p_t \propto \left|R_{t+1}+\gamma_{t+1}\max_{a'}q_{\overline{\theta}}(S_{t+1},a')-q_\theta(S_t,A_t)\right|^\omega$$

샘플링 확률식

이때, w는 분포의 모양을 결정하는 하이퍼파라미터를 의미한다.
새로운 TD 오차 값이 더 큰 transition의 경우 최대 우선순위로 추가되고, 이로 인해 좀 더 최신의 데이터에 초점을 맞춘다.

이를 통해 학습 가능성이 높은 전이를 더 자주 학습하여 효율성을 높일 수 있다.

3. Dueling Networks

문제점: 일부 상태에서는 행동 선택이 별로 중요하지 않지만 모든 상태-행동 쌍을 항상 동일하게 평가함

해결책: Q값을 V(s)라는 상태의 가치, A(s, a)라는 해당 상태에서 특정 행동의 상대적 우위 2개의 부분으로 나누어 고려함

$$q_\theta(s,a) = v_\eta(f_\xi(s)) + a_\psi(f_\xi(s),a) - \frac{\sum_{a'} a_\psi(f_\xi}{N_{acti}}$$

메뉴바를 클릭하면
계정을 관리할 수 있어요

2개의 부분으로 나눈 Q값 형태

이를 통해 상태의 가치와행동 간의 상대적 우위를 분리하여 좀 더 정확한 Q값 추정이 가능해진다.

4. Multi-step Learning

문제점: 기존의 Q-learning은 **1단계 보상값**만 보고 다음 상태로 boot-strap함
해결책: 여러 단계를 고려하는 n-step return을 사용하여 학습 속도를 향상시킴

$$R_t^{(n)} \equiv \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}$$

n-step 수식

DQN의 milti-step variant의 경우 다음과 같은 loss값을 최소화함으로서 정의된다.

$$(R_t^{(n)} + \gamma_t^{(n)} \max_{a'} q_{\overline{\theta}}(S_{t+n}, a') - q_\theta(S_t, A_t))^2$$

Alternative Loss값

이를 통해 여러 보상값을 동시에 고려함으로써 더 빠르게 수렴 가능하고, 장기적인 전략 학습이 가능해진다.

5. Distributional RL

문제점: 기존의 Q-learning은 **기댓값만 학습**하며, 보상의 분산값이나 불확실성을 반영하지 못함
해결책: 보상의 확률 분포 자체를 학습하는 방식을 활용함

- Q값은 고정된 지점(zi)에 확률 질량(probability mass)로 분포를 표현한다.

$$z^i = v_{\min} + (i-1)\frac{v_{\max}-v_{\min}}{N_{\text{atoms}}-1}$$

메뉴바를 클릭하면
계정을 관리할 수 있어요

- 최적 정책에 따라 다음 상태의 분포를 discount, 이동시킨 후에 KL Divergence
여 그 차이를 최소화시킨다.

$$d'_t \equiv (R_{t+1} + \gamma_{t+1}\boldsymbol{z},\ \boldsymbol{p}_{\overline{\theta}}(S_{t+1}, \overline{a}^*_{t+1})),$$
$$D_{\text{KL}}(\Phi_{\boldsymbol{z}} d'_t || d_t)\,.$$

이를 통해 보상의 불확실성까지 고려할 수 있어 전보다 더 안정적인 정책 학습이 가능하다.

6. Noisy Nets

문제점: greedy 정책의 경우 복잡한 탐색 과정이 필요한 환경에서는 매우 비효율적인 정책임

해결책: 신경망의 가중치에 노이즈를 추가하여 상태에 따라 다양한 행동을 유도함

$$\boldsymbol{y} = (\boldsymbol{b} + \mathbf{W}\boldsymbol{x}) + (\boldsymbol{b}_{noisy} \odot \epsilon^b + (\mathbf{W}_{noisy} \odot \epsilon^w)\boldsymbol{x})$$

노이즈값을 추가한 가중치 식

이를 통해 greedy 정책 없이도 상태 기반의 탐색 수행이 가능하며, 학습이 진행되면서 노이즈 값을 자동으로 조절 가능하다.

4. The Integrated Agent

본 논문의 Rainbow 모델은 section 3에서 소개한 DQN의 6가지 주요 확장 기법을 **하나의 프레임워크로 통합**한 모델이다.

먼저, **다단계 분포 손실 측면**이다.

$$d'_t \equiv (R_{t+1} + \gamma_{t+1}\boldsymbol{z},\ \boldsymbol{p}_{\overline{\theta}}(S_{t+1}, \overline{a}^*_{t+1})),$$
$$D_{\text{KL}}(\Phi_{\boldsymbol{z}} d'_t || d_t)\,.$$

기존의 분포 기반 손실값

기존의 분포 기반의 손실값을 n-step 보상을 반영한 값으로 대체한다.
즉, St+n에서의 분포를 n-step discount를 적용하여 감소시키고, 누적된 보상을
분포를 우측으로 이동시킨다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

$$d_t^{(n)} = (R_t^{(n)} + \gamma_t^{(n)} \boldsymbol{z}, \quad \boldsymbol{p}_{\bar{\theta}}(S_{t+n}, a_{t+n}^*))$$

목표로 하는 분포(target distribution)

목표로 하는 분포를 위와 같이 정의할 때,

$$D_{\mathrm{KL}}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t)$$

손실 함수

loss fuction은 위와 같다.
이 과정으로 n-step 장기 보상을 분포 기반으로 고려하여 더 정확하고 안정적인 학습이 가능해진다.

다음으로, **Double Q-learning과의 결합**이다.
Bootstrap 과정 시 사용되는 행동 a*_{t+n}는 현재 학습 중인 네트워크로 선택되는데,
해당 행동의 가치는 타깃 네트워크로 평가된다.

이를 통해 과도한 Q값 추정을 방지할 수 있고, 안정성도 동시에 유지 가능하다.

다음으로, **Prioritized Replay와 KL divergence 손실 기반의 우선순위 측면**이다.
기존의 전통적인 우선순위 리플레이의 경우 TD 오차를 기준으로 하여 샘플링 확률을 결정했는데,
Rainbow 모델의 경우 그 대신에 KL Divergence 손실값 자체를 우선순위 기준으로 사용한다.

$$p_t \propto \left(D_{\mathrm{KL}}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t) \right)^{\omega}$$

우선순위 샘플링 확률

KL Divergence 손실을 기준으로 하여 위와 같은 식으로 샘플링 확률을 결정하게 된다.
KL 기반의 우선순위는 noisy한 환경에서 기존의 TD 오차 기반의 방법보다 더 안
밀한 학습 필요성 반영이 가능하다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

다음으로 **dueling 네트워크 구조와 분포 예측 부분**이다.
Rainbow 모델은 분포 기반의 학습을 위해 Dueling 아키텍처를 확장한다.

$$p_\theta^i(s,a) = \frac{\exp(v_\eta^i(\phi) + a_\psi^i(\phi,a) - \overline{a}_\psi^i(s))}{\sum_j \exp(v_\eta^j(\phi) + a_\psi^j(\phi,a) - \overline{a}_\psi^j(s))}$$

분포 출력 계산식

이를 통해 각 행동의 보상 분포를 정확하게 추정이 가능하다.

마지막으로, Noisy Nets 사용 부분이다.
여기서는 모든 기존의 선형층을 노이즈가 포함된 선형층으로 대체하고,
Factorized Gaussian Noise를 사용하여 계산 효율성을 높인다.

이를 통해 탐색 과정을 더 효과적으로 수행이 가능하다.

즉 Rainbolw는 기존의 6가지의 기술들의 장점들을 조합하여 보다 강력한 강화학습이 가능한 모델이라는 것이다.

5. Experimental Methods

Evaluation Methodology

실험 대상: 총 57종의 Atari 2600 게임

평가 시 1M(100만 개)의 스텝마다 평가하고, 108,000 프레임(약 30분 분량) 이후에 중단한다.

평가 시 성능 지표로 Normalized Score를 사용한다.
Normalized Score, 즉 정규화 점수의 경우 0은 Random 에이전트의 성능을, 100은 인간 전문가 정도의 평균 점수를 의미한다.
정규화 점수를 기반으로 모든 57종의 게임들에서 성능을 종합적으로 비교한다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

이때 일반적으로 중앙값 정규화 성능이 사용된다.
평균값 대신에 중앙값을 사용하는 이유는 특정 게임들에서 평균값은 에이전트가 인간 전문가보다 훨씬 높은 점수를 얻기 때문에 왜곡될 가능성이 있어 중앙값을 사용한다.

추가로 비율 기반의 분석도 함계 사용하여 몇 개의 게임에서 인간 전문가 성능의 일정 비율을 달성했는지를 비교한다.

테스트 시에는 에피소드 시작 시 최대 30개의 무작위 no-op 행동을 삽입한다.
에피소드 시작 상태를 인간 전문가 플레이 경로 중 임의의 지점에서 샘플링한다.

Hyper-parameter tuning

Rainbow 모델은 다양한 기술, 구성 요소들을 통합한 모델로 하이퍼파라미터 조합이 매우 많다.
이에 전체 탐색은 매우 어려워 주요 파라미터들만 수동으로 조정한다.

Parameter	Value
Min history to start learning	80K frames
Adam learning rate	0.0000625
Exploration ϵ	0.0
Noisy Nets σ_0	0.5
Target Network Period	32K frames
Adam ϵ	1.5×10^{-4}
Prioritization type	proportional
Prioritization exponent ω	0.5
Prioritization importance sampling β	$0.4 \rightarrow 1.0$
Multi-step returns n	3
Distributional atoms	51
Distributional min/max values	$[-10, 10]$

Rainbow 모델의 hyper-parameters

탐색 방법은 Noisy Nets 사용 여부에 따라 달라진다.

먼저, Noisy Nets를 사용하는 경우 완전 탐욕 정책으로서 입실론값이 0이다.
이 경우 노이즈를 0.5로 초기화하여 사용한다.

반면에 Noisy Nets를 사용하지 않는 경우에는 greedy 정책을 사용하여 입실론값
빠른 시간 내에 0.01까지 감소시킨다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

최적화 시에는 Adam optimized을 사용한다.

가장 중요한 Replay Prioritization 부분이다.
Proportional Prioritization을 사용하여 우선순위 지수 w값으로 0.4, 0.5, 0.7을 비교하였고, 최종적으로 0.5를 사용하였다.
중요도 샘플링 보정 계수인 베타값은 학습 과정에서 0.4에서 1.0으로 선형 증가하였다.
KL Divergence를 사용하여 우선순위를 결정하기 때문에 noisy한 환경에서도 견고하다.

Multi-step 학습 과정에서 n값으로 1, 3, 5일 때를 실험하였다.
학습 초기에는 n = 3과 n = 5일 때가 유사하였으니, 최종적으로는 성능이 n = 3일 때가 가장 우수하다는 것을 확인할 수 있었다.

실험 대상인 총 57종의 Atari 2600 게임에서 모두 동일한 하이퍼파라미터를 사용하였다.
즉, Rainbow 모델의 경우 각 게임별로 특화하는 과정 없이 **기존의 단일 구성만으로도 전반적으로 우수한 성능을 발휘 가능**하다.

6. Analysis

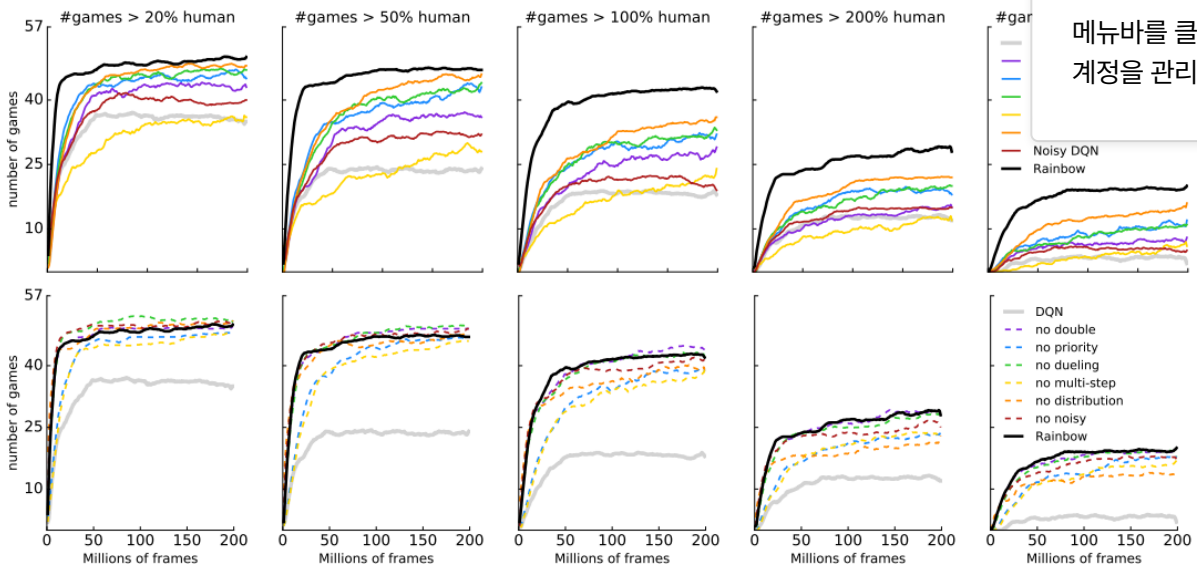
주요한 실험 결과들에 대해 분석해보고자 한다.

Comparison to published baselines

A3C, DQN, DDQN, Prioritized DDQN, Dueling DDQN, Distributional DQN, Noisy DQN 등과 비교하여
Rainbow 모델의 성능을 평가하였다.

성능 측정으로 전체 게임에 대해 human normalized score의 중앙값을 활용하여 성능값을 서로 비교하였다.

메뉴바를 클릭하면
계정을 관리할 수 있어요



Rainbow와 다른 기존의 모델들과의 성능 비교 그래프(1행: Rainbow vs baselines / 2행: Rainbow vs ablations)

바로 위의 그래프들에서 볼 수 있듯이, **Rainbow 모델은 학습 효율성과 최종 성능 모두에서 모든 기존 에이전트를 능가한다**는 사실을 확인할 수 있었다.

특히 1행의 그래프들의 경우 왼쪽부터 순서대로 20%, 50%, 100%, 200%, 500%의 특정 성능 임계값 이상에 도달한 게임의 수를 비교한 것인데, 모든 특정 성능 임계값에서 다른 모델들보다 더 많은 게임에서 좋은 성능을 기록하였다.

Agent	no-ops	human starts
DQN	79%	68%
DDQN (*)	117%	110%
Prioritized DDQN (*)	140%	128%
Dueling DDQN (*)	151%	117%
A3C (*)	-	116%
Noisy DQN	118%	102%
Distributional DQN	164%	125%
Rainbow	223%	153%

최종 평가 결과

no-ops, human starts 2개의 부분 모두 Rainbow의 성능이 가장 좋다.

Rainbow 모델의 경우 DQN의 최종 성능을 700만 프레임 내에 도달하였고, 4400만 프레임에서 모든 베이스라인들을 뛰어넘었다.

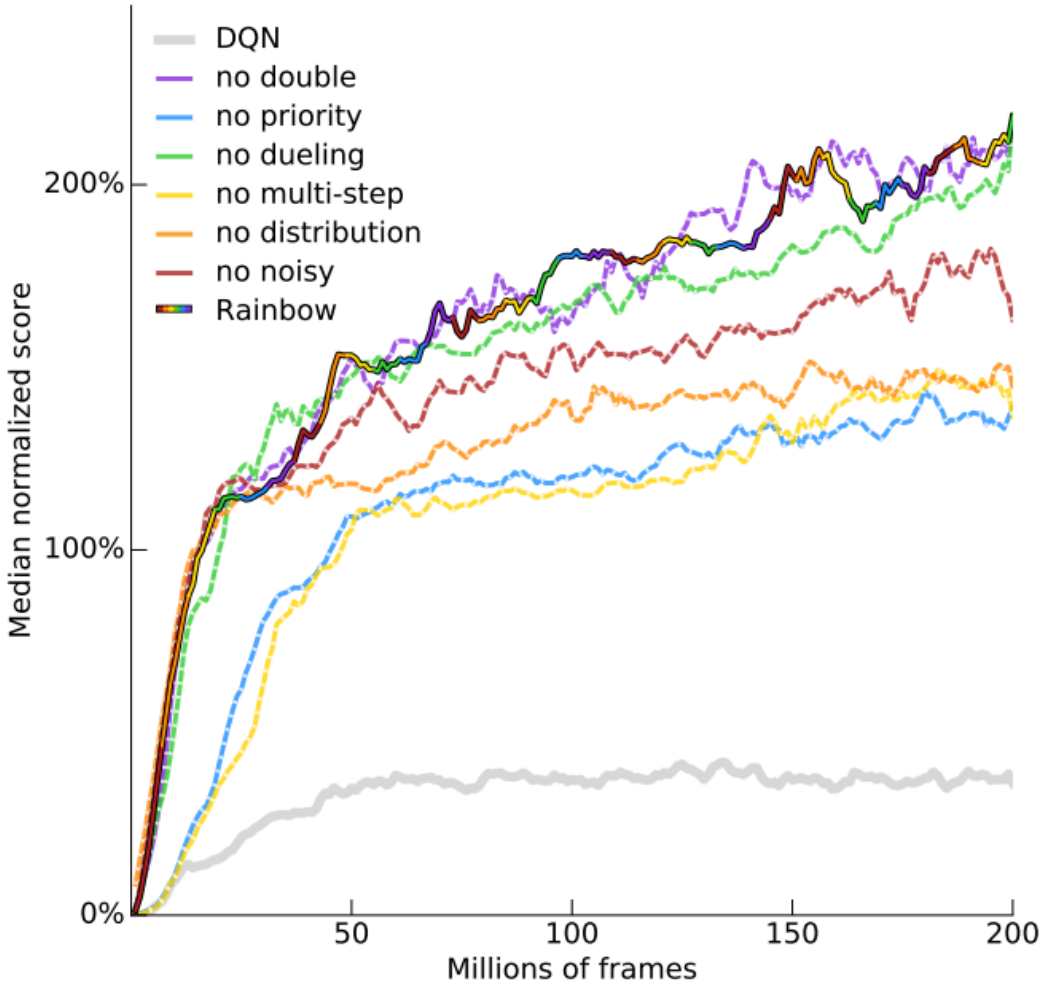
메뉴바를 클릭하면
계정을 관리할 수 있어요

Learning Speed

DQN과 동일하게 단일 GPU로 학습하였다.
700만 프레임을 학습하는 데에 걸린 시간은 약 10시간, 전체 2억 개의 프레임을 학습하는 데에 걸린 시간은 약 10일이었다.

Ablation Studies

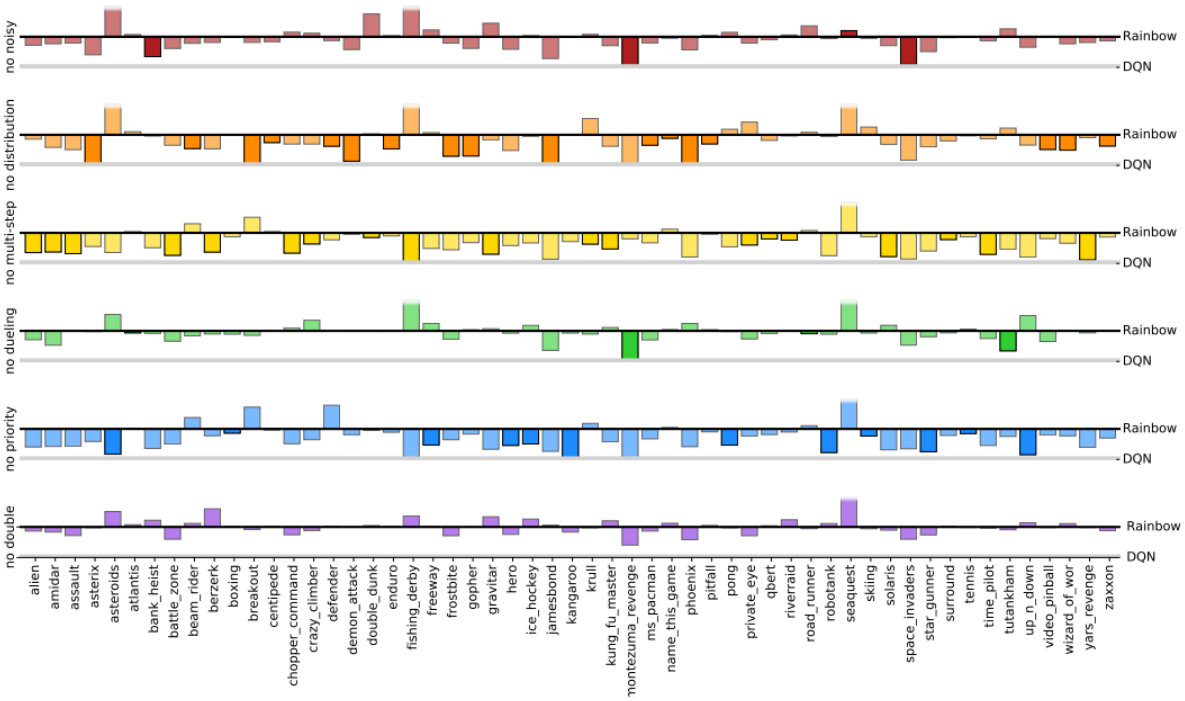
여러 개의 구성 요소들을 통합한 모델인 Rainbow는 각 요소의 중요도를 파악하기 위해 Ablation Study를 수행하였다.
이를 위해서 각각의 구성요소들을 1개씩 제거한 Rainbow의 변형된 모델들과 성능을 비교하였다.



DQN vs Ablation vs Rainbow

성능 비교 결과 **Prioritized Replay와 Multi-step Learning 2개의 요소가 제거된 Ablation 이 성능이 가장 낮았다.**
즉 Rainbow의 성능에 가장 큰 영향을 미친다는 사실을 확인할 수 있었다.

메뉴바를 클릭하면
계정을 관리할 수 있어요



각각의 Atari 게임들에 대한 Ablation agent의 성능 저하

특히 **Multi-step Learning을 제거할 경우 학습 초기의 성능뿐만이 아니라 최종 성능도 크게 저하**되는 사실을 확인할 수 있었다.
실제로, 57개의 Atari Games 중에서 53개의 게임들에서 원래의 완전한 Rainbow 모델이 Ablation보다 성능이 우위에 있다는 사실을 확인할 수 있었다.

추가적으로 6개의 기존 기법들 중에서 Distributional Q-learning과 Noisy Nets의 영향에 대해서도 살펴보고자 한다.
Prioritized Replay와 Multi-step Learning 요소처럼 매우 중요한 영향을 끼치는 요인들은 아니지만, 게임에 따라 선택적으로 강력한 기여를 한다.

먼저, Distributional Q-learning의 경우 초기 학습에서는 큰 차이가 없으나, 40M 프레임 이후부터 성능 차이가 발생한다.
특히나 인간 이상의 수준의 성능이 요구되는 게임들에서 Distributional Q-learning이 제거된 Rainbow 모델의 성능이 좋지 않다.

다음으로 Noisy Nets의 경우 제거 시 greedy 탐색으로 대체되나 성능 저하가 발생하게 된다.
일부 게임들에서는 성능이 향상되기도 하였으나 전체적으로는 성능이 하락하게 된다.

메뉴바를 클릭하면
계정을 관리할 수 있어요

7. Discussion

Rainbow는 여러 DQN 기반의 개선 기법들을 통합하여 **최신 수준의 성능**을 달성하는 강화학습 알고리즘을 성공적으로 구현하였다.

Rainbow에 통합된 여러 구성요소들은 대부분 성능 향상에 기여했고, 하나를 제외하고는 모두 긍정적인 효과를 보였다.
하지만, 아직 Rainbow에 포함되지 않은 많은 강화학습 기법들이 있으며, **이 기법들 중에서 일부는 추후에 또다른 통합형 에이전트의 성능을 더욱 향상시킬 가능성이 크다.**

실제로 Rainbow는 Q-learning 계열의 가치 기반으로 하는 방법들에 집중하였으나,
추가적으로 TRPO, Actor-Critic 계열의 A3C, ACER과 같은 방법들도 고려할 만하다.

추가적으로 순차 데이터 활용 방법으로서 Optimality Tightening과 같이 n-step 보상을 기존의 목표값을 대체하는 데에 활용하지 않고 부등식 제약 조건으로도 사용할 수 있고, 다양한 길이의 n-step 보상을 부드럽게 결합 가능한 Eligibility Traces도 활용 가능하다.

이에 더불어
1) 기억 기반의 학습인 Episodic Control을 통해 초기 학습 효율성이 증가하고 성공적인 행동 시퀀스를 즉시 재현 가능하고,
2) 기존의 DQN 방식처럼 단일 환경, 단일 에이전트로 학습 대신에 병렬 환경을 사용하면 학습 속도를 추가로 개선 가능하고,
3) 프레임 스택 기반의 상태 표현 사용 대신에 Pixel/Feature Control 방법 처럼 입력을 조절하는 보조 과제로 상태 학습 개선하거나, Supervised Prediction 방법으로 미래 예측을 통해 표현을 학습하는 과정으로 상태 표현 정도를 개선 가능하는 등

추가적으로 개선할 수 있는 방법들이 아주 많다.