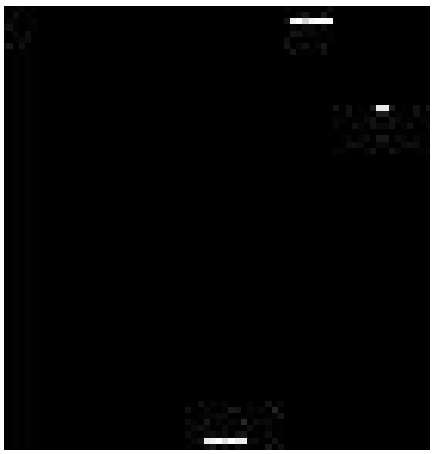
Нейронная сеть:

DQN сеть - сверточная сеть с тремя скрытыми слоями, аппроксимирующая нелинейную Q-функцию, которая играет важную роль в обучении с подкреплением. В данном случае, нейронная сеть применяется для игры в настольный теннис. Q-функция помогает оценивать значения для возможных действий , в данном случае - три действия возможны: “вверх”, “вниз” и “не делать ничего”. При тренировки нейросети используется уравнение Беллмана, которому должна удовлетворять Q-функция:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') \middle| s,a \right]$$

На вход нейронной сети подается текущее состояние - представляет собой четыре последние фотографии экрана:



Выход нейронной сети представляет собой оценку возможных действий. Сам же игрок выбирает действие с максимальной оценкой.

Вход нейронной сети: Массив размером [None, 84, 84, 4].
Первый скрытый слой состоит из 16 фильтров-матриц 8 x 8, с шагом 4 и применяет функцию активации max(0,x). Второй скрытый слой состоит из 32 фильтров-матриц 4 x 4, с шагом 2 и применяет функцию активации max(0,x). Последний скрытый слой является полностью соединенным линейным слоем с единственным выходом для каждого возможного действия.

Постановка задачи обучения:

Обучаемое множество - множество кортежей вида (s,a,s’,r), где s - состояние непосредственно перед действием a, s’ - состояние сразу после действия a, r - прибыль. Состояние - четыре фотографии экрана. Прибыль равна +1, если раунд завершился победой нейросети, -1 в другом случае. Цель обучения - аппроксимация Q-функции, таким образом, чтобы она была близка к оптимальной Q-функции, удовлетворяющей уравнению Беллмана. Если наш агент на каждом шаге будет

выбирать действие, которое оценивается с помощью Q функции лучше всех других, то его игра будет приближаться к идеальной.

Тренировка происходит пакетным образом, по 32 кортежа за раз. Кортежи выбираются случайным образом из базы данных. Таких тренировок происходит огромное множество и совершаются они в параллельном потоке.

Алгоритм обучения выглядит следующим образом : используя пакет из 32 кортежей, мы делаем шаг градиентного спуска , минимизируя квадрат разности:

$$(r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta') - Q(\phi_j, a_j; \theta))^2$$

где первая Q функция считается с помощью нейросети со старыми весами тета`, а новые веса для второй Q функции подбираются таким образом, чтобы минимизировать приведенную выше формулу.

Параметры обучения:

Размерность входных данных : [None, 72, 72, 4]
Размерность выходных данных : [None, 3]
Первый скрытый слой : 16 фильтров-матриц 8x8 с шагом 4 , функция активации max(0,x)
Второй скрытый слой : 32 фильтров-матриц 4x4, с шагом 2, функция активации max(0,x)
Третий скрытый слой : полностью соединенный линейный слой из 256 нейронов
Выход : массив [None,3]
Скорость обучения : 0.0001
Начальные веса для всех слоев инициализируются случайным образом из нормального распределения между (-0.01, 0.01)

Время :

а) Время обучения:
0.289614915848
1.86947488785
0.778320074081
0.992080926895

б) Время вычисления Q - функции :
0.011304101944
0.0588674163818
0.0440896987915

Проблемы, с которыми пришлось столкнуться:

1) Неправильно выбранные начальные веса не давали обучаться нейронной сети - в ходе тренировок менялся только последний слой

2) Как видно из основной статьи, технология, использованная мной, дает видимые результаты в игре Breakout только после 10 эпох. В каждой эпохе - 50000 мини-обновлений с размером 32. Получается, что всего должно быть около 500000 итераций, для этого в моем случае нужно трое суток тренировок. Компьютер зависает уже после нескольких часов тренировки. На это влияет большое количество данных, сохраняемых в базу данных MySQL, что влечет за собой длительное выполнение запросов к базе данных. Сейчас активно решаю эту проблему.

Литература:

https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf - основная статья
https://www.youtube.com/watch?v=WdhSqmO2Dy0 - как должна работать моя сеть