DQN report

Anton Kudinov

Environment

Agents compete in shooter arena versus each other. Each of them can perform several actions from the list per turn:

- Move left/right
- Move forward/back
- Turn left/right
- Switch to pistol/shotgun/rocker launcher/machinegun
- Shoot

Different weapons create different missiles and deal different amount of damage. The other elements of the game are bonuses. There are different types of bonuses: bullet packs, shell packs, rockets, armor vests, medkits.

Three target metrics are: kills, deaths, differences (kills - deaths). Kills and differences should be maximized and deaths minimized.

Agent

Agent chooses which actions to perform every 5 ticks (repeating same actions the other 4 ticks). This decision is up to DQN which bases on agent`s vision. Agent`s vision consists of 13 vectors of length 17. Each vector represents special distance indicators of one type of objects or status bar. Their order:

- Walls
- Enemies
- Medkits
- Armor vests
- Bullet packs
- Shell packs
- Rockets
- Fired missiles
- Health bar
- Armor bar
- Bullets bar
- Shells bar
- Rockets bar

Distance indicators are ray-casted for 17 rays from –pi/6 to pi/6. So one equals 1 if object if very close and 0.05 if distance equals to agent`s vision range (was set to 300 pixels).

Neural network architecture

DQN consisting of following layers showed the best achieved results:

- Convolution layer (32 kernels of size 3 with relu activation)
- Max pooling (size 2)

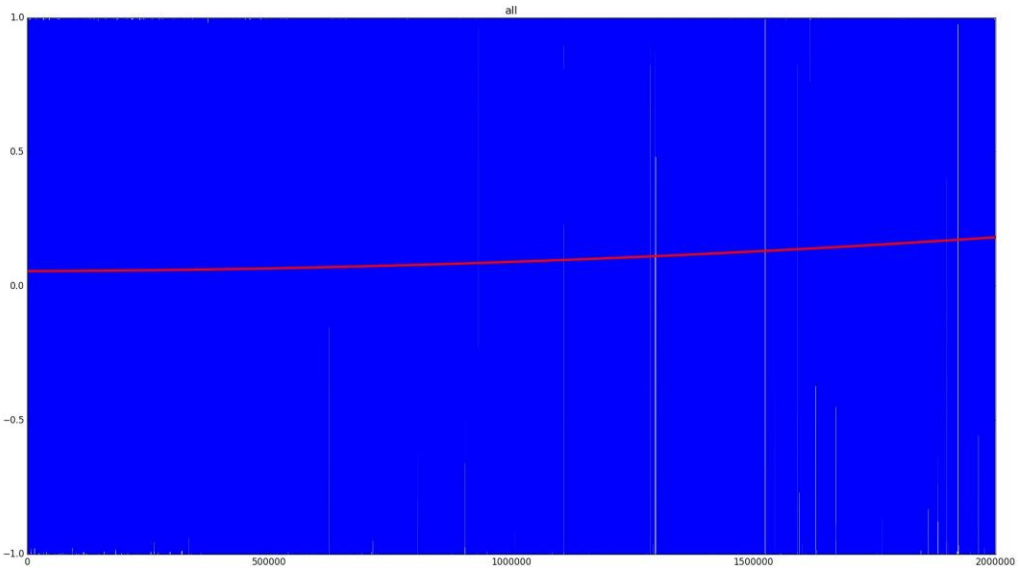
- Batch normalization
- Convolution layer (64 kernels of size 3 with sigmoid activation)
- Max pooling (size 2)
- Batch normalization
- Dense (64 neurons with sigmoid activation)
- Batch normalization
- Dense (32 neurons with relu activation)
- Dense (11 neurons with step 0.1 activation)

DQN rewards

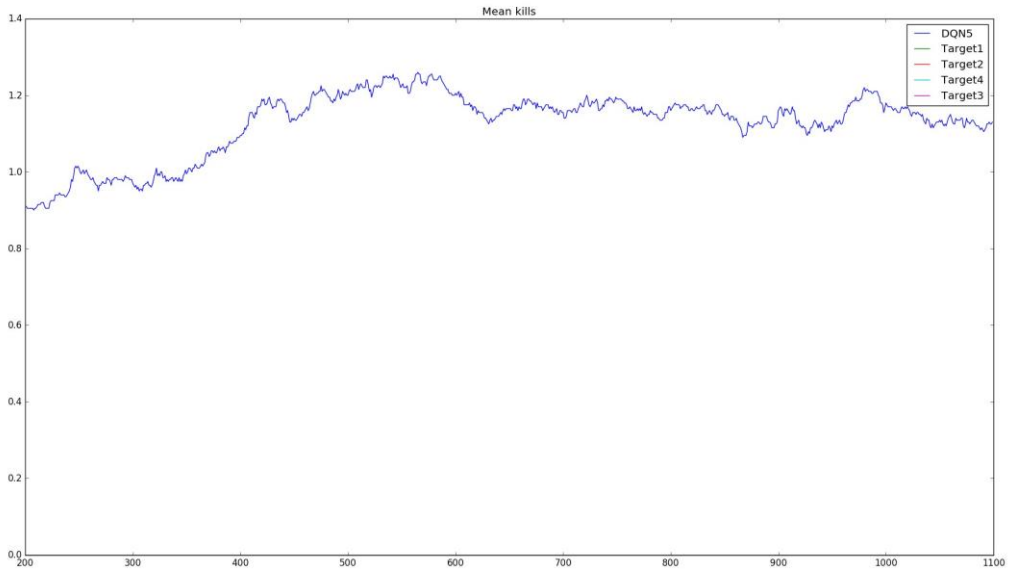
+ 1 for each accurate shot
+ 0.5 for each bonus received
+ 0.05 for each tick
- 0.2 for missed shot
- 0.25 for attempt to shoot without ammunition
- 1 for death

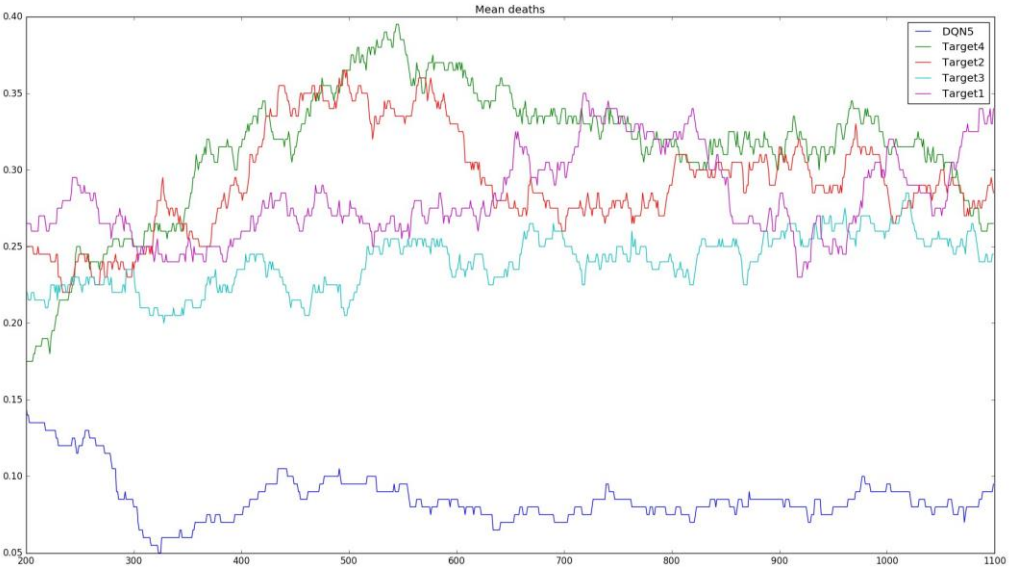
Results

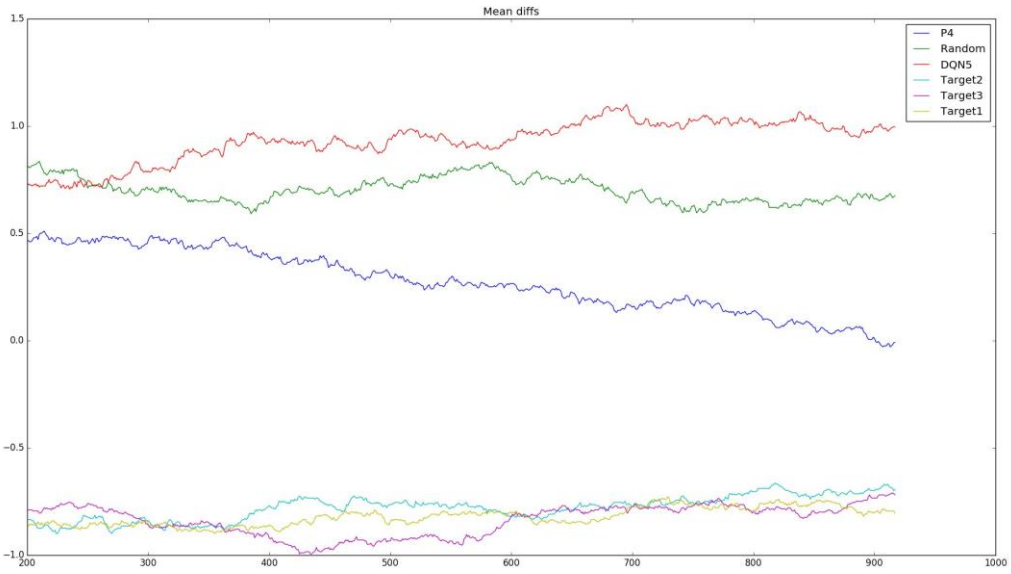
Well, it doesn`t really know how to play. But it definitely plays better than random. My bot was learning to play on map with 4 targets (same as bots, but do nothing). It`s rewards grew steadily, but slow (red line is quadratic linear regression, x - ticks):



It`s target metrics (average for last 200 episodes):


Mean kills
DQN5
Target1
Target2
Target4
Target3
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
200
300
400
500
600
700
800
900
1000
1100


Mean deaths
DQN5
Target4
Target2
Target3
Target1
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
200
300
400
500
600
700
800
900
1000
1100



After learning my agent was challenged by random agent. After 1000 episodes random`s mean diff score was about -0.3. My agent`s was about 0.1, which proves it`s supremacy.

If you don`t trust pictures

https://github.com/162/2D-shooter-enviroment

This repository is prepared for demonstration. You need to have pygame, h5py, theano/tensorflow, keras to be able to run project. Simply run main.py to see epic fight between my bot and random bot.