Flappy Bird

Leistungsnachweis Reinforcement Learning

Student: Mathias Moser

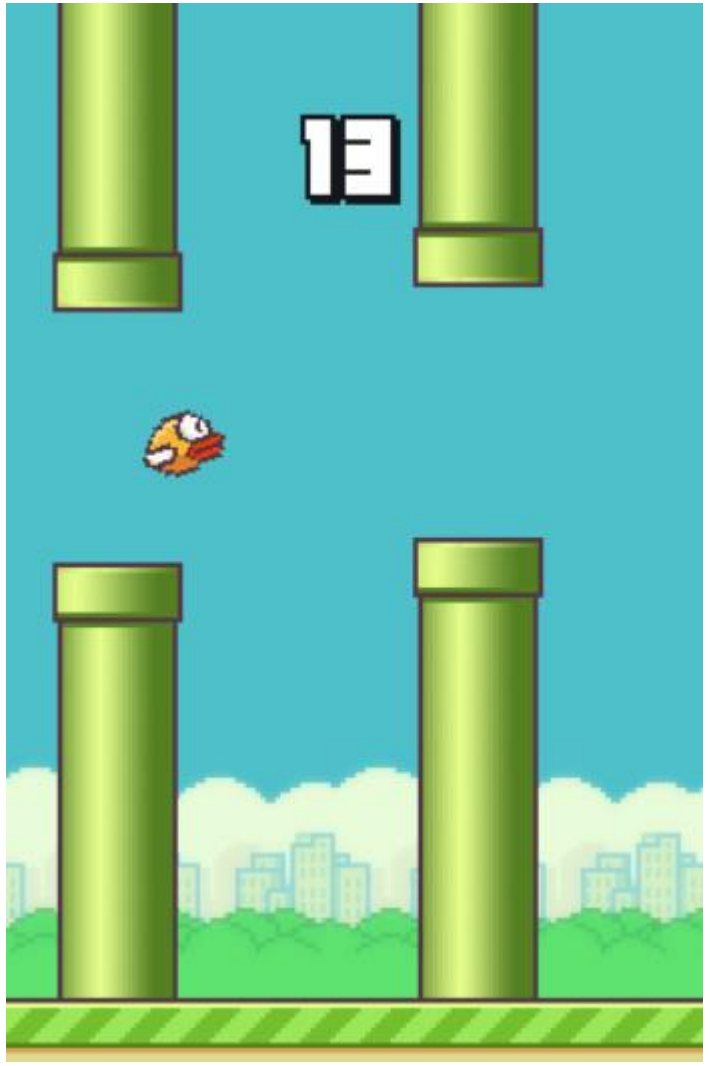
Dozent: Manuel Renold

https://github.com/Karakeen1/reinforcment_learning/tree/main/flappybird

Einleitung

- Was ist Flappy Bird?

- Aufgabenstellung:
  - Anhand voll Bildern soll ein Modell trainiert werden, welches Flappy Bird spielt
  - Human Top Score: 2’311

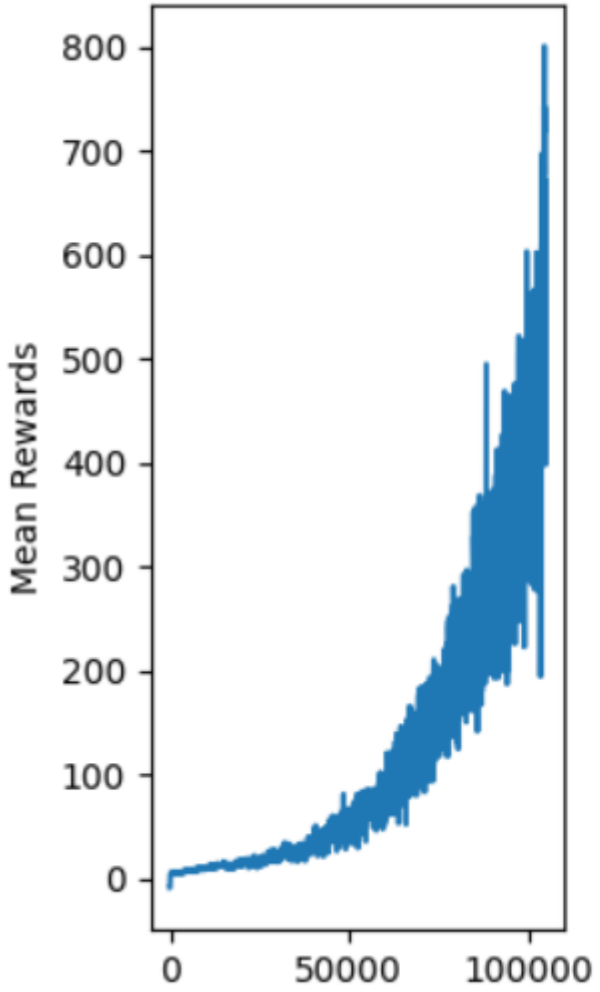


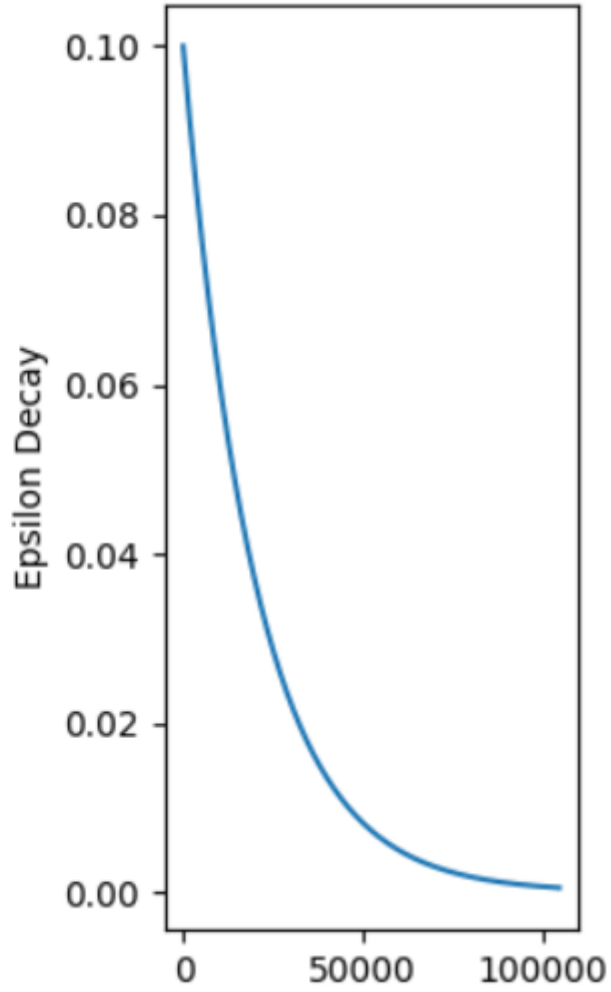
Starting simple

- DQN
  - Input 12
  - 1 hidden Layer 1024
  - Output 2 (no flap, flap)

States vom Environment:
the last pipe's horizontal position
the last top pipe's vertical position
the last bottom pipe's vertical position
the next pipe's horizontal position
the next top pipe's vertical position
the next bottom pipe's vertical position
the next next pipe's horizontal position
the next next top pipe's vertical position
the next next bottom pipe's vertical position
player's vertical position
player's vertical velocity
player's rotation

Zudem: *terminated* und *reward*





Flappy Bird environment: https://github.com/markub3327/flappy-bird-gymnasium/blob/main/README.md

Agent

- current_q: Modell vorhergesagte Q-Werte für a im gegebenen s
- target_q: angestrebte Q-Werte, aktueller reward und zukünftig Erwartung
- Replay Memory:
  - 100’000 Erfahrungstupel (state, action, new_state, reward, terminated)
  - Ein Minibatch von 32 tupeln wird gesampelt um Model stabiler zu trainieren
- Epsilon greedy

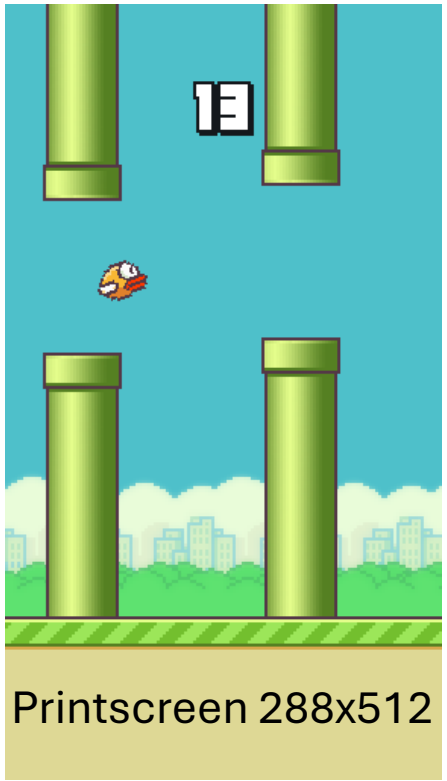
Modell

1 Frame schwarz/weiss

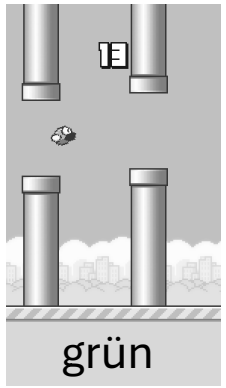

```
self.conv1 = nn.Conv2d(in_channels=1, out_channels=32, kernel_size=6, stride=3)
self.relu1 = nn.ReLU(inplace=True)
self.fc4 = nn.Linear(32*13*13, 1024) # 5’408
self.relu4 = nn.ReLU(inplace=True)
self.fc5 = nn.Linear(1024, self.number_of_actions) # 2
```

Bild als Input

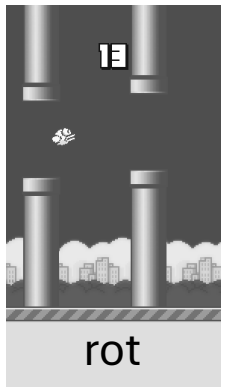
- Printscreen von jedem Environment-Step
- Preprocessing:



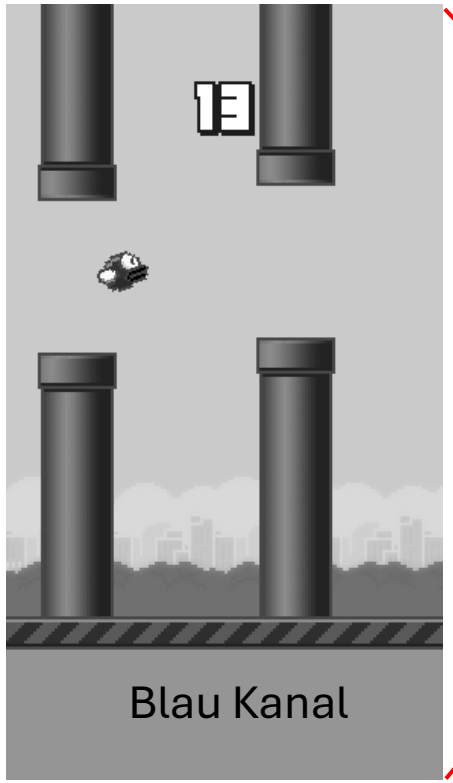
Printscreen 288x512



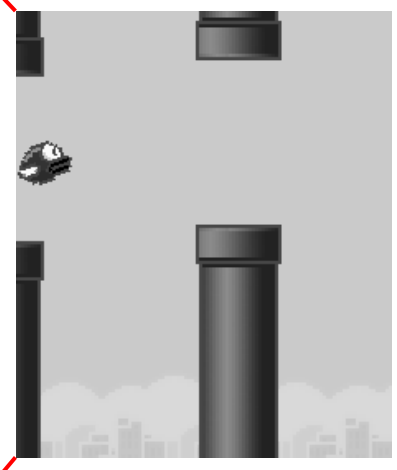
grün



rot

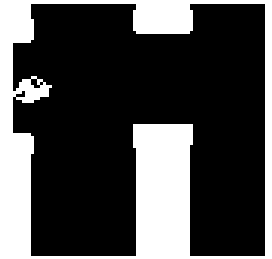


Blau Kanal



Zuschnitt auf
Relevanten Bereich

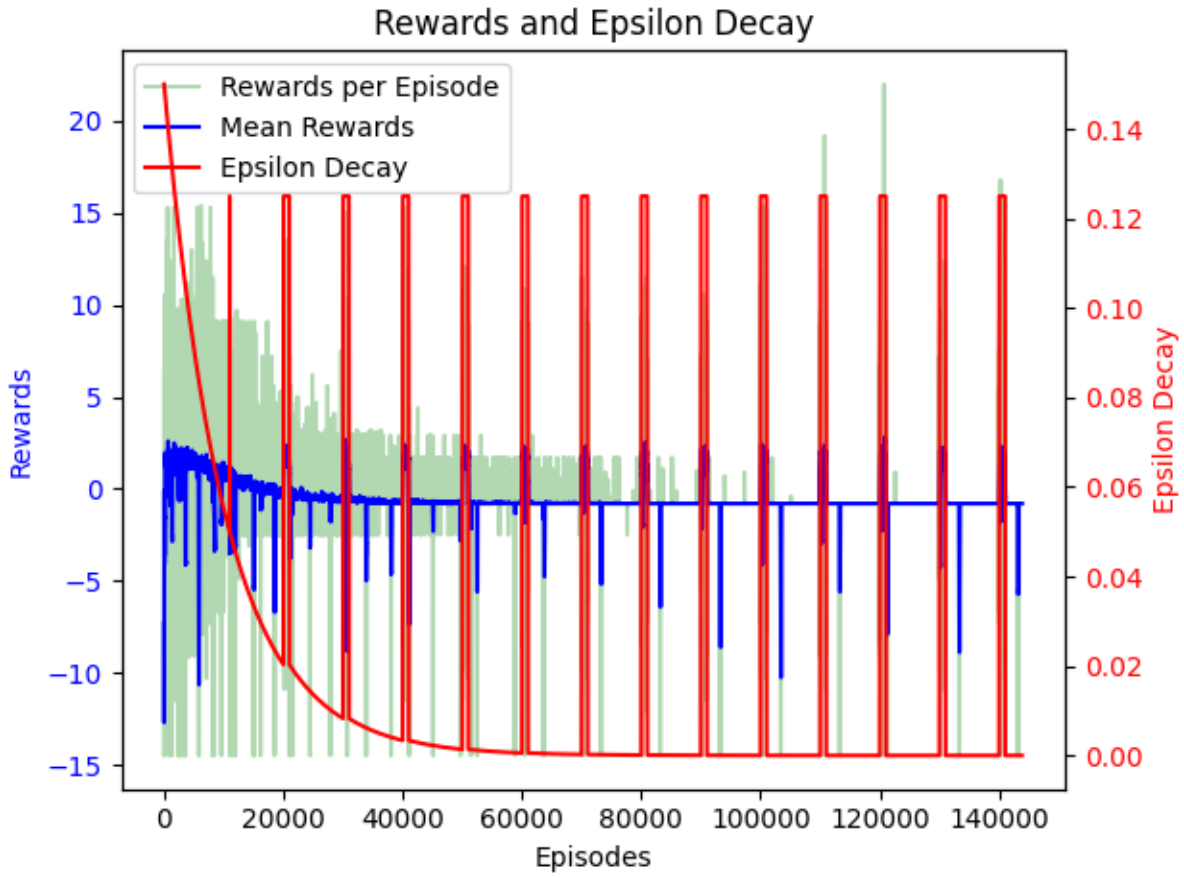
Threshold > 170 = 0, else 1
Hell zu schwarz, dunkel zu weiss
Bildgrösse reduzieren



Bildgrösse
84x84 Pixel

«Resultat»

- Modell lernt nicht
- Zufällige actions führten zu besserem Score
- Je 10’000 Episoden 1000 episoden mit hoher zufälligkeit

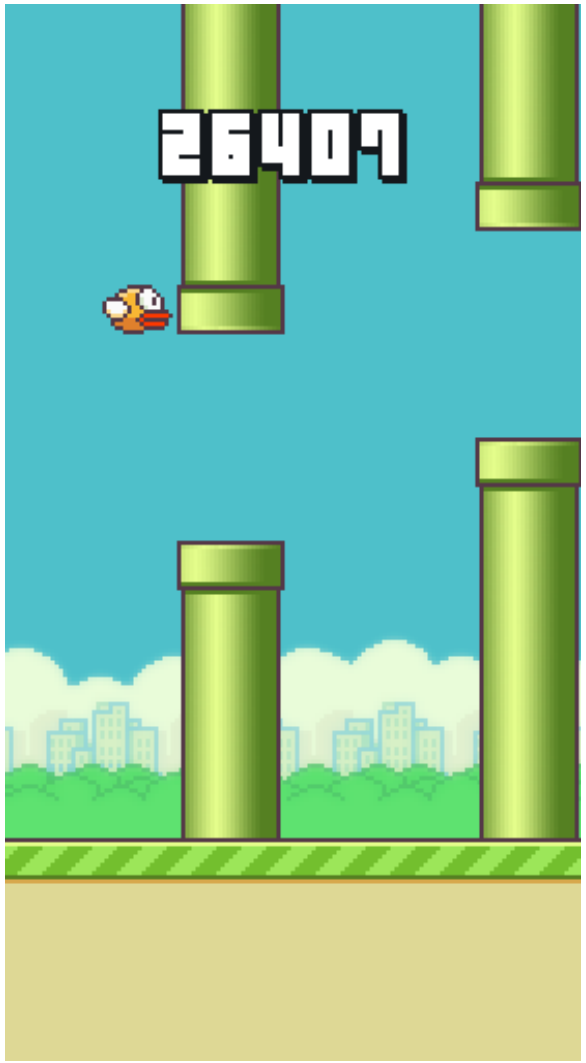


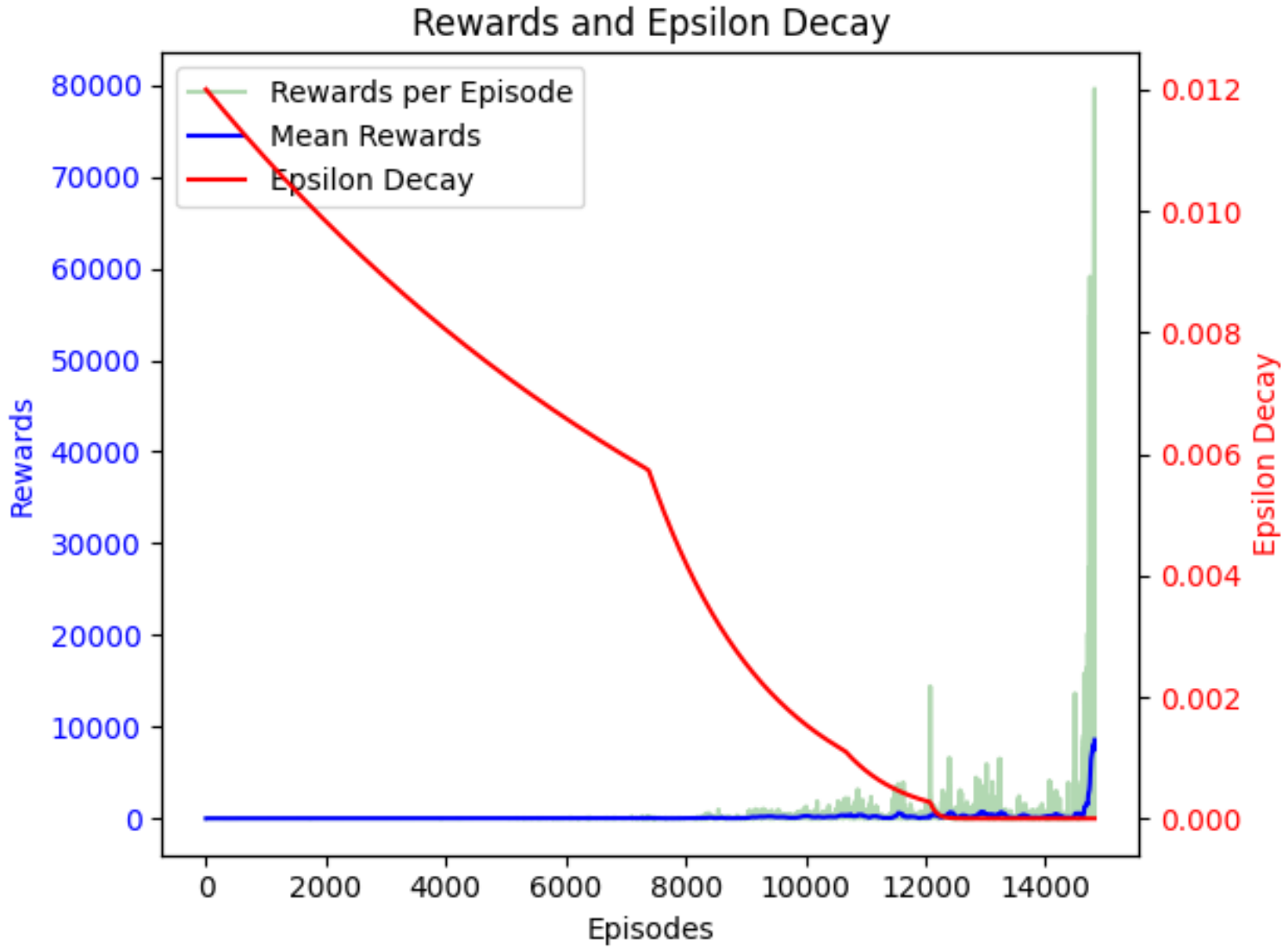
Optimierung

- 4 Frames stacked als Input
- Modell lernt Bewegungsrichtung und Beschleunigung
- Modell auf 3 Convolutional Layer erweitert

```
self.conv1 = nn.Conv2d(in_channels=4, out_channels=32, kernel_size=8, stride=4)
self.relu1 = nn.ReLU(inplace=True)    # 20x20x32
self.conv2 = nn.Conv2d(32, 64, 4, 2) # 9x9x64
self.relu2 = nn.ReLU(inplace=True)
self.conv3 = nn.Conv2d(64, 64, 3, 1) # 7x7x64
self.relu3 = nn.ReLU(inplace=True)
self.fc4 = nn.Linear(3136, 1024)
self.relu4 = nn.ReLU(inplace=True)
self.fc5 = nn.Linear(1024, self.number_of_actions) # actions = 2
```

Resultat


26407


Rewards and Epsilon Decay
Rewards per Episode
Mean Rewards
Epsilon Decay
Rewards
Epsilon Decay
Episodes
80000
70000
60000
50000
40000
30000
20000
10000
0
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
2000
4000
6000
8000
10000
12000
14000

Resultat Kontinuierlich (30 Episoden)