# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities


WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us?
## Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

# Can Reinforcement Learning Be Applied to Surgery?

Masakazu Sato, Kaori Koga, Tomoyuki Fujii and Yutaka Osuga

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.76146

**Abstract**

**Background**: Remarkable progress has recently been made in the field of artificial intelligence (AI).

**Objective**: We sought to investigate whether reinforcement learning could be used in surgery in the future.

**Methods**: We created simple 2D tasks (Tasks 1–3) that mimicked surgery. We used a neural network library, Keras, for reinforcement learning. In Task 1, a Mac OS X with an 8 GB memory (MacBook Pro, Apple, USA) was used. In Tasks 2 and 3, a Ubuntu 14. 04LTS with a 26 GB memory (Google Compute Engine, Google, USA) was used.

**Results**: In the task with a relatively small task area (Task 1), the simulated knife finally passed through all the target areas, and thus, the expected task was learned by AI. In contrast, in the task with a large task area (Task 2), a drastically increased amount of time was required, suggesting that learning was not achieved. Some improvement was observed when the CPU memory was expanded and inhibitory task areas were added (Task 3).

**Conclusions**: We propose the combination of reinforcement learning and surgery. Application of reinforcement learning to surgery may become possible by setting rules, such as appropriate rewards and playable (operable) areas, in simulated tasks.

**Keywords:** hysterectomy, robotic surgical procedures, deep learning, artificial intelligence, deep Q network (DQN), reinforcement learning

## 1. Introduction

Remarkable progress has recently been made in the field of artificial intelligence (AI), and new advancements are made almost every day [1–3]. Although it is not possible to discuss

IntechOpen

© 2018 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

all developments in AI, such as image recognition, automatic driving or trials of the game of Go, a particularly promising AI task is game playing [4–6]. AI can be trained to play games, such as the game Breakout, and can perform as well as an experienced human subject. In the context of game play, reinforcement learning is a common strategy [6, 7]. For instance, the objective of the game Breakout is to break blocks to score points. While the performance of the player at the instance of breaking a block should not necessarily be evaluated, the performance immediately prior to successfully breaking a block, such as the player's ability to direct the ball, should be evaluated. With this aim, the AI calculates the expected values of the scores (reward) that could be obtained by various actions and learns how to achieve the maximum reward via reinforcement learning [6]. Thus, the AI succeeds in breaking more blocks and obtaining higher scores.

The concept of reinforcement learning has a commonality with surgical procedures. A hysterectomy, which is one of the most basic gynecologic procedures [8], can be considered as an example. The aim of a hysterectomy is to enucleate the uterus; however, there are no specific guidelines regarding precisely how the knife should be used at the moment that the uterus is being removed. Rather, resecting the uterine arteries in advance, for instance, may reduce total blood loss, and this possibility should be evaluated. In addition, the resection of ligaments should also be evaluated, as this step precedes extraction of the uterus.

Altogether, we simulated and investigated whether reinforcement learning could be applied to surgery and whether AI could be possibly used to perform surgeries in the future.

In this study, we created simple 2D tasks that mimicked surgery (such as a hysterectomy simulation) and investigated whether the surgery could be performed as expected via reinforcement learning. During the task, the player (an imaginary knife) moved around the task areas. The player scored when passing target areas, such as imaginary ligaments and arteries, and lost points for other actions. The task was over when the player reached the uterus or moved outside the task area. We established rules and observed the process during which movement of the imaginary knife was learned and improved. In the task with a relatively small task area (Task 1), the knife finally passed all target areas, and the expected learning was achieved. In contrast, in the task with a large task area (Task 2), significantly more time was required to complete the task, suggesting that learning was not achieved. We addressed this problem by expanding the CPU memory and adding inhibitory task areas (Task 3), and some improvements were observed.

In this study, we applied the concept of reinforcement learning to surgical procedures and identified some commonalities between reinforcement learning and the way surgeons approach an operation. We found that various aspects of the techniques of efficient learning should be developed for applying reinforcement learning to surgery. These aspects include choosing a model that closely reproduces the surgical scene using a high-performance computer for deep learning and tuning of neural networks. Surgeons will need to understand the rules, such as when a reward can be earned and where the agent is allowed to move. Efficient learning would be possible by integrating these rules into the AI by engineers.

## 2. Materials and methods

### 2.1. 2D tasks

To create the 2D task, a publicly available game code for the game "Snake" was referenced and tuned (https://github.com/farizrahman4u/qlearning4k). The code for the operation task is provided in the **S7 Text**. The snake game was used for this investigation because it could be easily modified to the task we needed. Therefore, in particular, the original code should not necessarily be the snake game if one could create the task which mimics surgery.

### 2.2. Task 1

We created 9 × 9 squares of a simple 2D task representing a surgery (a hysterectomy) as shown in **Figure 1A**. The goal of the task is to remove the object (uterus, yellow); indirect goals, such as cutting ligaments and arteries, were also established. In the context of surgery, there are preferred cutting points that consider the densities of the arteries and the distance from the object, and these points were set as target points (green). Thus, the objective was for the tip of the knife (orange) to pass (cut) all the target points. In other words, if surgeons see the mass (tumors or uterus what so ever) to remove like yellow areas in **Figure 1A**, then they would cut the ligament (green areas) by using knife as drawing a circle, without approaching the area where the bleeding is expected to occur (red). Rewards, such as +1 for passing green areas, −1 for passing red areas, and 0 for passing blue areas (peritoneum), were defined. In addition, −1

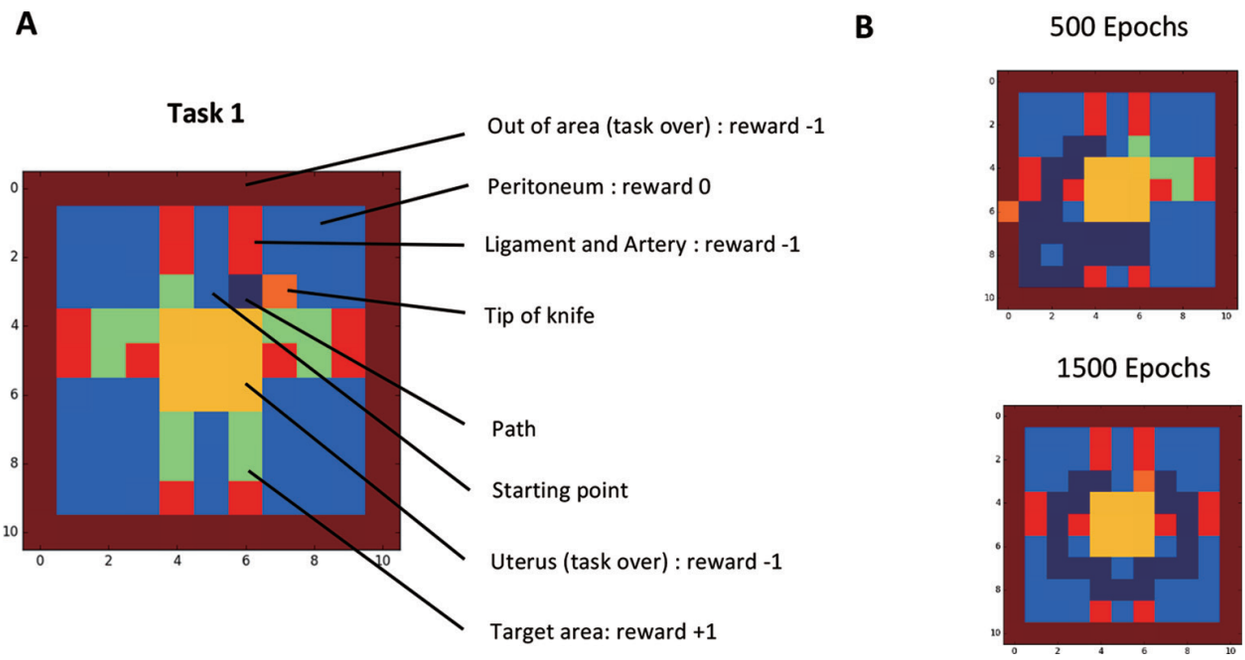


**Figure 1.** Creation of task. (A) Rules of the task. The name and reward of each area were as follows: orange, tip of knife; dark blue, path where tip of knife had passed; red, ligament or artery, −1; green, target area, +1; yellow, uterus, −1 (task over); blue, peritoneum, 0; brown, out of area, −1 (task over). Starting point was located in the upper middle of area. (B) Results of learning. The results after 500 epochs of learning (upper) and 1500 epochs of learning (lower) are shown. The knife passed all the green points in the shortest amount of time after learning.

was given and the task was defined as over if the knife passed over the yellow area (uterus) or if the knife passed outside the task area. Finally, −1 was given for passing areas were previously passed.

### 2.3. Task 2

We extended the task to 48 × 48 squares. This imposed significantly heavier burdens on the computer compared to Task 1, and we extended the CPU memory as described below.

### 2.4. Task 3

We added inhibitory areas (which would trigger the task to be over) to Task 2, and named it Task 3.

### 2.5. Deep Q network

We used a neural network library, Keras, for the reinforcement learning (https://keras.io) and TensorFlow as its backend (https://www.tensorflow.org) [9]. The codes for the agent can be found at https://github.com/farizrahman4u/qlearning4k.

The code simply obey the rules' Q(S, a) = r + gamma * Q(S', a')'.

Q(S, a) means the maximum score the agent will get by the end of the game, if it does action a, when the game is in state S. On performing action a, the game will jump to a new state S', giving the agent an immediate reward r.

In short, Q(S, a) = Immediate reward from this state + Max-score from the next state onwards.

Gamma is a discounting factor to give more weight to the immediate reward.

We are gynecologists and understand the very minimum of deep q learning, and we just tuned nb_frames (the number of frames should the agent remeber), batch_size (batch size for training), gamma (discounting factor) and nb_eopch (number of epochs to train). The original neural networks contained two convolutional layers and two dense layers. We slightly tuned the network by adding convolutional layers or changing the size of frames, but we did not see much improvement. Thus, the readers might find a more efficient way of learning. However, the objective of the present study is to propose the combination of reinforcement learning and surgery, and we did not investigate the tuning any further after obtaining conclusions.

### 2.6. Development environment

The development environments used in this study were Python 2.7.12, Keras 1.1.0, TensorFlow 0.8.0, and Matplotlib1.5.3. In Task 1, a Mac OS X with an Intel Core i5 processor and 8 GB memory (MacBook Pro, Apple, USA) was used. In Tasks 2 and 3, a Ubuntu 14. 04LTS with an Intel Xenon E5 v2 processor and 26 GB memory (Google Compute Engine, Google, USA) was used.

## 3. Results

### 3.1. Learning process and trials for appropriate learning

We created 9 × 9 squares of a simple 2D task representing the surgical scene (**Figure 1A**). We provide representative movies after 10 epochs, 500 epochs and 1500 epochs of learning (**Figure 1B** and **Movies 1–3**). The knife passed all the green points in the shortest amount of time after learning.

We next extended the task to 48 × 48 squares (**Movies 4** and **5**). This imposed significantly heavier burdens on the computer than in Task 1, and therefore, we expanded the CPU memory as described in the Section 2. The knife was located in limited areas after 500 epochs of learning (**Movie 4**), and the area of movement was extended after 1500 epochs of learning (**Movie 5**). Even after extending the CPU memory, the task required a long amount of time, and thus, learning was not necessarily achieved. Thus, we modified Task 2 to Task 3.

In Task 2, the knife moved to areas where no reward could be obtained (upper right area), which was hypothesized to prevent appropriate learning (**Movie 5**). We considered that some areas are not significant in an actual hysterectomy and added inhibitory areas (Task 3) that would trigger the task to end (dark blue areas in **Movie 6**). By adding these inhibitory areas, the knife reached the lower areas where it had never reached after 1500 epochs of learning (**Movie 6**). However, even with these limitations, a long time for learning was required within the simulated environments. These results suggest that efficient learning can be achieved by setting appropriate rules such as adding inhibitory areas. All video materials referenced in this section are available at: https://www.intechopen.com/download/index/process/160/authkey/f5524cabe6c92eff86119b2f0c58ba26

## 4. Discussion

In this study, we simulated and investigated whether reinforcement learning could be applied to surgery in the future and evaluated the types of hurdles that may exist.

Progress in AI has been remarkable, and AI is currently an essential part of the technology industry. Recent progress in AI can be attributed to the development of deep learning. Deep learning is a form of machine learning featured by the characteristic that the user does not have to choose features or representations while inputting data [10]. Deep learning is currently widely used in image recognition and audio recognition. While playing games with reinforcement learning has also been investigated, reinforcement learning has recently been combined with deep learning, resulting in drastic improvements in performance [4–6].

Thus, we sought to investigate whether reinforcement learning could be used in surgery in the future and developed appropriate simulations. Our team is composed of clinicians rather than engineers; however, we performed this study after studying deep learning and reinforcement learning.

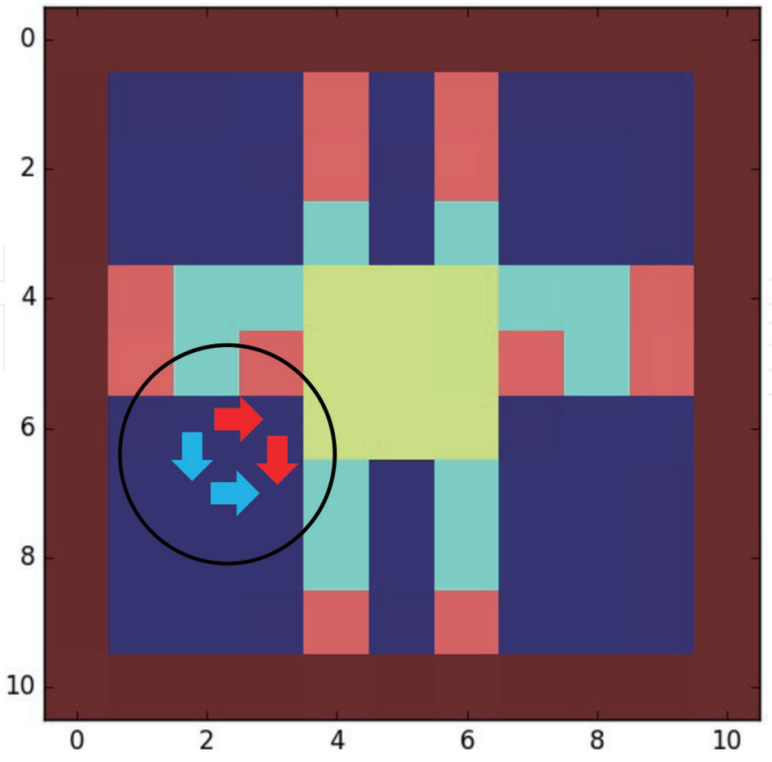


**Figure 2.** Choice of the shortest route. Blue and red arrows in the circle represent the shortest routes. Only the route of the blue arrows was chosen after several trials.

In reinforcement learning, it is preferable to obtain a reward during the immediately following action rather than having expectations to obtain rewards in distant future actions [6]. This can be understood by the results of Task 1, in which the knife passed all the green areas in the shortest possible time, even though a rule never inhibited a more roundabout approach (**Figure 1B**). This can be expected to as a way to shorten the surgical time. Furthermore, choosing the shortest route was also of interest. The blue and red arrows in the circle in **Figure 2** represent the shortest routes; the route of the blue arrows was chosen after several trials (**Figure 1B**). The method used to decide the next action in reinforcement learning is maximizing the expected values that would be obtained after subsequent actions [6]. Therefore, actions that have greater expected values are likely to be preferred. Therefore, we expected that the red arrows were not preferred because points would be deducted (red area) or the task would be ended (uterus) with subsequent actions. This approach is also considered superior in surgery. In other words, avoiding areas associated with point deductions would result in retaining the appropriate margins during surgery. Therefore, the characteristics of reinforcement learning seemed to be compatible with typical surgical approaches.

The other results obtained were expected. It was concluded that reinforcement learning could solve a simple 2D task; 3D models that more closely replicate surgery should be further considered. In addition, increasing the playable area provides significantly more options for actions, and high-performance computers and tuning of neural networks will be needed for more complex tasks. 3D operation models are currently being developed and are used in

practice for endoscopic surgery simulation [11–14]. The speed at which high-performance computers are being developed is astonishing, and deep learning is being thoroughly investigated by engineers both in academia and business [15–17]. Combining the progress in both fields may provide designs allowing AI to realistically perform surgery. In such areas, clinicians can contribute the essential aspects of the surgery, that is, the playable task areas and the appropriate scores. Although increasing reward areas or limiting playable areas would speed learning time, such restrictions could also prevent the AI from finding alternative paths. Thus, clinicians and engineers should work cooperatively define rules.

In this study, we applied the concept of reinforcement learning to surgical procedures and identified common points between reinforcement learning and the way surgeons approach an operation. Reinforcement learning is now exclusively used and studied in the context of game play; however, it could also be applied to performing surgeries now that robotic surgery is widely available. Although there are many hurdles to overcome, AI could be applied to surgery by setting the appropriate rules, such as defining rewards and playable (operable) areas. To realize this goal, it is important for clinicians to further study deep learning and reinforcement learning strategies.

## Acknowledgements

We greatly appreciate American Journal Experts for their generous help in editing the manuscript.

## Conflict of interest

Authors declare there is no conflict of interests.

## Supporting information

**Movie 1.** Result of Game 1 after 10 epochs of learning.

**Movie 2.** Result of Game 1 after 500 epochs of learning.

**Movie 3.** Result of Game 1 after 1500 epochs of learning.

**Movie 4.** Result of Game 2 after 500 epochs of learning.

**Movie 5.** Result of Game 2 after 1500 epochs of learning.

**Movie 6.** Result of Game 3 after 1500 epochs of learning.

https://www.intechopen.com/download/index/process/160/authkey/f5524cabe6c92eff86119b2f0c58ba26

**S7 Text.** The game codes for Game 1, 2 and 3 (python script). The code of agent.py, memory.py and game.py could be obtained in the link in Section 2.

```
# game1_test.py

from keras.models import Sequential
from keras.layers import *
from game1 import OP
from keras.optimizers import *
from agent import Agent

grid_size = 11
nb_frames = 5
nb_actions = 5

model = Sequential()
model.add(Convolution2D(16, 2, 2, activation='relu', input_shape=(nb_frames,
grid_size, grid_size)))
model.add(Convolution2D(32, 3, 3, activation='relu'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(nb_actions))
model.compile(RMSprop(), 'MSE')

operation = OP(grid_size)

agent = Agent(model=model, memory_size=-1, nb_frames=nb_frames)
agent.train(operation, batch_size=64, nb_epoch=500, gamma=0.8)
agent.play(operation)


# game1.py

__original_author__ = "Fariz Rahman"
# https://github.com/farizrahman4u/qlearning4k

# Modified by "Masakazu Sato" on 10th December,2016.
import numpy as np
from game import Game


actions = {0:'left', 1:'right', 2:'up', 3:'down', 4:'idle'}
forbidden_moves = [(0, 1), (1, 0), (2, 3), (3, 2)]

class OP(Game):

    def __init__(self, grid_size=11, knife_length=1):
        self.grid_size = grid_size
        self.knife_length = knife_length
        self.reset()
        self.state_changed = True
```

```
    @property
    def name(self):
        return "OP"
    @property
    def nb_actions(self):
        return 5

    def play(self, action):
        assert action in range(5), "Invalid action."
        self.plus_scored = False
        self.minus_scored = False
        self.non_scored = False
        self.move_knife(action)
        tip = self.knife[0]

        if self.hit_artery():
            self.minus_scored = True
            self.art_border.remove(tip)
        elif self.hit_marker():
            self.plus_scored = True
            self.mark_border.remove(tip)
        elif self.hit_peritoneum():
            self.non_scored = True
            self.per_border.remove(tip)
        elif self.hit_border():
            self.game_over = True
        else:
            self.minus_scored = True



    def arrange(self):
         self.artery = (1,4),(1,6),(2,4),(2,6),(4,1),(4,9),(5,1),(5,3),(5,7),
(5,9),(9,4),(9,6)
         self.marker = (3,4),(3,6),(4,2),(4,3),(4,7),(4,8),(5,2),(5,8),(7,4),
(7,6),(8,4),(8,6)
         self.uterus = (4,4),(4,5),(4,6),(5,4),(5,5),(5,6),(6,4),(6,5),(6,6)
      self.peritoneum = (1,1),(1,2),(1,3),(1,5),(1,7),(1,8),(1,9),(2,1),(2,2),
(2,3),(2,5),(2,7),(2,8),(2,9),(3,1),(3,2),(3,3),(3,5),(3,7),(3,8),(3,9),
(6,1),(6,2),(6,3),(6,7),(6,8),(6,9),(7,1),(7,2),(7,3),(7,5),(7,7),(7,8),
(7,9),(8,1),(8,2),(8,3),(8,5),(8,7),(8,8),(8,9),(9,1),(9,2),(9,3),(9,5),
(9,7),(9,8),(9,9)

    def new_arrange(self):
         self.artery = (1,4),(1,6),(2,4),(2,6),(4,1),(4,9),(5,1),(5,3),(5,7),
(5,9),(9,4),(9,6)
         self.marker = (3,4),(3,6),(4,2),(4,3),(4,7),(4,8),(5,2),(5,8),(7,4),
(7,6),(8,4),(8,6)
         self.uterus = (4,4),(4,5),(4,6),(5,4),(5,5),(5,6),(6,4),(6,5),(6,6)
        self.peritoneum = (1,1),(1,2),(1,3),(1,5),(1,7),(1,8),(1,9),(2,1),(2,
2),(2,3),(2,5),(2,7),(2,8),(2,9),(3,1),(3,2),(3,3),(3,5),(3,7),(3,8),(3,9),
```

```
(6,1),(6,2),(6,3),(6,7),(6,8),(6,9),(7,1),(7,2),(7,3),(7,5),(7,7),(7,8),(7,
9),(8,1),(8,2),(8,3),(8,5),(8,7),(8,8),(8,9),(9,1),(9,2),(9,3),(9,5),(9,7),
(9,8),(9,9)

        self.art_border = []
        self.art_border += self.artery
        self.mark_border = []
        self.mark_border += self.marker
        self.per_border = []
        self.per_border += self.peritoneum

    def move_knife(self, action):
        if action == 4 or (action, self.previous_action) in forbidden_moves:
            action = self.previous_action
        else:
            self.previous_action = action
        head = self.knife[0]
        if action == 0:
            p = (head[0] - 1, head[1])
        elif action == 1:
            p = (head[0] + 1, head[1])
        elif action == 2:
            p = (head[0], head[1] - 1)
        elif action == 3:
            p = (head[0], head[1] + 1)
        self.knife.insert(0, p)
        self.knife.pop()


    def get_state(self):
        canvas = np.ones((self.grid_size, ) * 2)
        canvas[1:-1, 1:-1] = 0.0
        for seg in self.knife:
            canvas[seg[0], seg[1]] = 0.8
        for seg in self.art_border:
            canvas[seg[0],seg[1]] = 0.9
        for seg in self.mark_border:
            canvas[seg[0], seg[1]] = 0.5
        for seg in self.ut_border:
            canvas[seg[0],seg[1]] = 0.7
        for seg in self.per_border:
            canvas[seg[0],seg[1]] = 0.2
        return canvas

    def get_score(self):
        if self.game_over:
            score = -3
        elif self.plus_scored:
            score = +3
        elif self.minus_scored:
            score = -2
        elif self.non_scored:
            score = +1
```

```
        else:
            score = -1
        return score

    def reset(self):
        grid_size = self.grid_size
        knife_length = self.knife_length
        self.knife = [(3,5)]
        self.game_over = False
        self.plus_scored = False
        self.minus_scored = False
        self.non_scored = False
        self.arrange()
        if np.random.randint(2) == 0:
            self.previous_action = 0
        else:
            self.previous_action = 1
        self.border = []
        for z in range(grid_size):
          self.border += [(z, 0), (z, grid_size - 1), (0, z), (grid_size - 1, z)]
        self.ut_border = []
        self.ut_border += self.uterus
        self.art_border = []
        self.art_border += self.artery
        self.mark_border = []
        self.mark_border += self.marker
        self.per_border = []
        self.per_border += self.peritoneum

    def left(self):
        self.play(0)
    def right(self):
        self.play(1)
    def up(self):
        self.play(2)
    def down(self):
        self.play(3)
    def idle(self):
        self.play(4)

    def hit_border(self):
       return self.knife[0] in self.border or self.knife[0] in self.ut_border
    def hit_artery(self):
        return self.knife[0] in self.art_border
    def hit_marker(self):
        return self.knife[0] in self.mark_border
    def hit_peritoneum(self):
        return self.knife[0] in self.per_border

    def is_over(self):
        return self.hit_border()
    def is_won(self):
        return len(self.mark_border) < 1
```

```
# game2_test.py

from keras.models import Sequential
from keras.layers import *
from game2 import OP
from keras.optimizers import *
from agent import Agent

grid_size = 50
nb_frames = 5
nb_actions = 5

model = Sequential()
model.add(Convolution2D(16, 2, 2, activation='relu', input_shape=(nb_frames,
grid_size, grid_size)))
model.add(Convolution2D(32, 3, 3, activation='relu'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(nb_actions))
model.compile(RMSprop(), 'MSE')

operation = OP(grid_size)

agent = Agent(model=model, memory_size=5, nb_frames=nb_frames)
agent.train(operation, batch_size=64, nb_epoch=500, gamma=0.80)
agent.play(operation)


# game2.py

__original_author__ = "Fariz Rahman"
# Modified by "Masakazu Sato" on 10th December,2016.
import numpy as np
from game import Game


actions = {0:'left', 1:'right', 2:'up', 3:'down', 4:'idle'}
forbidden_moves = [(0, 1), (1, 0), (2, 3), (3, 2)]

class OP(Game):

    def __init__(self, grid_size=50, knife_length=1):
        self.grid_size = grid_size
        self.knife_length = knife_length
        self.reset()
        self.state_changed = True

    @property
    def name(self):
        return "OP"
    @property
    def nb_actions(self):
        return 5
```

```
    def play(self, action):
        assert action in range(5), "Invalid action."
        self.plus_scored = False
        self.minus_scored = False
        self.non_scored = False
        self.move_knife(action)
        tip = self.knife[0]

        if self.hit_artery():
            self.minus_scored = True
            self.art_border.remove(tip)
        elif self.hit_marker():
            self.plus_scored = True
            self.mark_border.remove(tip)
        elif self.hit_peritoneum():
            self.non_scored = True
            self.per_border.remove(tip)
        elif self.hit_border():
            self.game_over = True
        else:
            self.minus_scored = True


    def arrange(self):
       self.artery = (1,21),(1,22),(2,21),(2,22),(3,21),(3,22),(4,21),(4,22),
(5,21),(5,22),(6,21),(6,22),(7,21),(7,22),(8,21),(8,22),(1,27),(1,28),(2,
27),(2,28),(3,27),(3,28),(4,27),(4,28),(5,27),(5,28),(6,27),(6,28),(7,27),
(7,28),(8,27),(8,28),(11,21),(11,22),(11,23),(12,21),(12,22),(12,23),(13,21),
(13,22),(13,23),(14,21),(14,22),(14,23),(15,21),(15,22),(15,23),(16,21),(16,
22),(16,23),(11,26),(11,27),(11,28),(12,26),(12,27),(12,28),(13,26),(13,27),
(13,28),(14,26),(14,27),(14,28),(15,26),(15,27),(15,28),(16,26),(16,27),(16,
28),(21,1),(21,2),(22,1),(22,2),(23,1),(23,2),(24,1),(24,2),(25,1),(25,2),
(26,1),(26,2),(27,1),(27,2),(28,1),(28,2),(23,3),(23,4),(23,5),(24,3),(24,4),
(24,5),(25,3),(25,4),(25,5),(26,3),(26,4),(26,5),(24,6),(24,7),(24,8),(24,9),
(25,6),(25,7),(25,8),(25,9),(23,12),(23,13),(23,14),(24,12),(24,13),(24,14),
(25,12),(25,13),(25,14),(26,12),(26,13),(26,14),(21,15),(21,16),(22,15),(22,
16),(23,15),(23,16),(24,15),(24,16),(25,15),(25,16),(26,15),(26,16),(27,15),
(27,16),(28,15),(28,16),(21,33),(21,34),(22,33),(22,34),(23,33),(23,34),(24,
33),(24,34),(25,33),(25,34),(26,33),(26,34),(27,33),(27,34),(28,33),(28,34),
(23,35),(23,36),(23,37),(24,35),(24,36),(24,37),(25,35),(25,36),(25,37),(26,
35),(26,36),(26,37),(24,40),(24,41),(24,42),(24,43),(25,40),(25,41),(25,42),
(25,43),(23,44),(23,45),(23,46),(24,44),(24,45),(24,46),(25,44),(25,45),(25,
46),(26,44),(26,45),(26,46),(21,47),(21,48),(22,47),(22,48),(23,47),(23,48),
(24,47),(24,48),(25,47),(25,48),(26,47),(26,48),(27,47),(27,48),(28,47),(28,
48),(33,21),(33,22),(33,23),(34,21),(34,22),(34,23),(35,21),(35,22),(35,23),
(40,21),(40,22),(40,23),(41,21),(41,22),(41,23),(42,21),(42,22),(42,23),(43,
21),(43,22),(43,23),(44,21),(44,22),(44,23),(45,21),(45,22),(45,23),(46,21),
(46,22),(46,23),(47,21),(47,22),(47,23),(48,21),(48,22),(48,23),(33,26),(33,
27),(33,28),(34,26),(34,27),(34,28),(35,26),(35,27),(35,28),(40,26),(40,27),
(40,28),(41,26),(41,27),(41,28),(42,26),(42,27),(42,28),(43,26),(43,27),(43,
28),(44,26),(44,27),(44,28),(45,26),(45,27),(45,28),(46,26),(46,27),(46,28),
(47,26),(47,27),(47,28),(48,26),(48,27),(48,28)
```

```
        self.marker = (9,21),(9,22),(10,21),(10,22),(9,27),(9,28),(10,27),(10,
28),(24,10),(24,11),(25,10),(25,11),(24,38),(24,39),(25,38),(25,39),(36,21),
(36,22),(36,23),(37,21),(37,22),(37,23),(38,21),(38,22),(38,23),(39,21),(39,
22),(39,23),(36,26),(36,27),(36,28),(37,26),(37,27),(37,28),(38,26),(38,27),
(38,28),(39,26),(39,27),(39,28)
        self.uterus = (21,17),(22,17),(23,17),(24,17),(25,17),(26,17),(27,17),
(28,17),(20,18),(21,18),(22,18),(23,18),(24,18),(25,18),(26,18),(27,18),(28,
18),(29,18),(19,19),(20,19),(21,19),(22,19),(23,19),(24,19),(25,19),(26,19),
(27,19),(28,19),(29,19),(30,19),(18,20),(19,20),(20,20),(21,20),(22,20),(23,
20),(24,20),(25,20),(26,20),(27,20),(28,20),(29,20),(30,20),(31,20),(17,21),
(17,22),(17,23),(17,24),(17,25),(17,26),(17,27),(17,28),(18,21),(18,22),(18,
23),(18,24),(18,25),(18,26),(18,27),(18,28),(19,21),(19,22),(19,23),(19,24),
(19,25),(19,26),(19,27),(19,28),(20,21),(20,22),(20,23),(20,24),(20,25),(20,
26),(20,27),(20,28),(21,21),(21,22),(21,23),(21,24),(21,25),(21,26),(21,27),
(21,28),(22,21),(22,22),(22,23),(22,24),(22,25),(22,26),(22,27),(22,28),(23,
21),(23,22),(23,23),(23,24),(23,25),(23,26),(23,27),(23,28),(24,21),(24,22),
(24,23),(24,24),(24,25),(24,26),(24,27),(24,28),(25,21),(25,22),(25,23),(25,
24),(25,25),(25,26),(25,27),(25,28),(26,21),(26,22),(26,23),(26,24),(26,25),
(26,26),(26,27),(26,28),(27,21),(27,22),(27,23),(27,24),(27,25),(27,26),(27,
27),(27,28),(28,21),(28,22),(28,23),(28,24),(28,25),(28,26),(28,27),(28,28),
(29,21),(29,22),(29,23),(29,24),(29,25),(29,26),(29,27),(29,28),(30,21),(30,
22),(30,23),(30,24),(30,25),(30,26),(30,27),(30,28),(31,21),(31,22),(31,23),
(31,24),(31,25),(31,26),(31,27),(31,28),(32,21),(32,22),(32,23),(32,24),(32,
25),(32,26),(32,27),(32,28),(18,29),(19,29),(20,29),(21,29),(22,29),(23,29),
(24,29),(25,29),(26,29),(27,29),(28,29),(29,29),(30,29),(31,29),(19,30),(20,
30),(21,30),(22,30),(23,30),(24,30),(25,30),(26,30),(27,30),(28,30),(29,30),
(30,30),(20,31),(21,31),(22,31),(23,31),(24,31),(25,31),(26,31),(27,31),(28,
31),(29,31),(21,32),(22,32),(23,32),(24,32),(25,32),(26,32),(27,32),(28,32)
         self.peritoneum = (22,3),(22,4),(22,5),(27,3),(27,4),(27,5),(22,35),
(22,36),(22,37),(27,35),(27,36),(27,37),(22,44),(22,45),(22,46),(27,44),(27,
45),(27,46),(6,17),(6,18),(6,19),(6,20),(7,17),(7,18),(7,19),(7,20),(8,17),
(8,18),(8,19),(8,20),(9,17),(9,18),(9,19),(9,20),(10,17),(10,18),(10,19),(10,
20),(11,17),(11,18),(11,19),(11,20),(12,17),(12,18),(12,19),(12,20),(13,17),
(13,18),(13,19),(13,20),(14,17),(14,18),(14,19),(14,20),(15,17),(15,18),(15,
19),(15,20),(16,17),(16,18),(16,19),(16,20),(17,7),(17,8),(17,9),(17,10),(17,
11),(17,12),(17,13),(17,14),(17,15),(17,16),(18,7),(18,8),(18,9),(18,10),
(18,11),(18,12),(18,13),(18,14),(18,15),(18,16),(19,7),(19,8),(19,9),(19,10),
(19,11),(19,12),(19,13),(19,14),(19,15),(19,16),(20,7),(20,8),(20,9),(20,10),
(20,11),(20,12),(20,13),(20,14),(20,15),(20,16),(17,17),(18,17),(19,17),(20,
17),(17,18),(18,18),(19,18),(17,19),(18,19),(17,20),(21,3),(21,4),(21,5),
(21,6),(21,7),(21,8),(21,9),(21,10),(21,11),(21,12),(21,13),(21,14),(22,6),
(22,7),(22,8),(22,9),(22,10),(22,11),(22,12),(22,13),(22,14),(23,6),(23,7),
(23,8),(23,9),(23,10),(23,11),(26,6),(26,7),(26,8),(26,9),(26,10),(26,11),
(27,6),(27,7),(27,8),(27,9),(27,10),(27,11),(27,12),(27,13),(27,14),(28,3),
(28,4),(28,5),(28,6),(28,7),(28,8),(28,9),(28,10),(28,11),(28,12),(28,13),(28,
14),(29,7),(29,8),(29,9),(29,10),(29,11),(29,12),(29,13),(29,14),(29,15),(29,
16),(30,7),(30,8),(30,9),(30,10),(30,11),(30,12),(30,13),(30,14),(30,15),(30,
16),(31,7),(31,8),(31,9),(31,10),(31,11),(31,12),(31,13),(31,14),(31,15),(31,
16),(32,7),(32,8),(32,9),(32,10),(32,11),(32,12),(32,13),(32,14),(32,15),(32,
```

16),(33,17),(33,18),(33,19),(33,20),(34,17),(34,18),(34,19),(34,20),(35,17),
(35,18),(35,19),(35,20),(36,17),(36,18),(36,19),(36,20),(37,17),(37,18),(37,
19),(37,20),(38,17),(38,18),(38,19),(38,20),(39,17),(39,18),(39,19),(39,20),
(40,17),(40,18),(40,19),(40,20),(29,17),(30,17),(31,17),(32,17),(30,18),(31,
18),(32,18),(31,19),(32,19),(32,20),(6,23),(6,24),(6,25),(6,26),(7,23),(7,24),
(7,25),(7,26),(8,23),(8,24),(8,25),(8,26),(9,23),(9,24),(9,25),(9,26),(10,23),
(10,24),(10,25),(10,26),(11,24),(11,25),(12,24),(12,25),(13,24),(13,25),(14,
24),(14,25),(15,24),(15,25),(16,24),(16,25),(33,24),(33,25),(34,24),(34,25),
(35,24),(35,25),(36,24),(36,25),(37,24),(37,25),(38,24),(38,25),(39,24),(39,
25),(40,24),(40,25),(6,29),(6,30),(6,31),(6,32),(7,29),(7,30),(7,31),(7,32),
(8,29),(8,30),(8,31),(8,32),(9,29),(9,30),(9,31),(9,32),(10,29),(10,30),(10,
31),(10,32),(11,29),(11,30),(11,31),(11,32),(12,29),(12,30),(12,31),(12,32),
(13,29),(13,30),(13,31),(13,32),(14,29),(14,30),(14,31),(14,32),(15,29),(15,
30),(15,31),(15,32),(16,29),(16,30),(16,31),(16,32),(33,29),(33,30),(33,31),
(33,32),(34,29),(34,30),(34,31),(34,32),(35,29),(35,30),(35,31),(35,32),(36,
29),(36,30),(36,31),(36,32),(37,29),(37,30),(37,31),(37,32),(38,29),(38,30),
(38,31),(38,32),(39,29),(39,30),(39,31),(39,32),(40,29),(40,30),(40,31),(40,
32),(17,33),(17,34),(17,35),(17,36),(17,37),(17,38),(17,39),(17,40),(17,41),
(17,42),(18,33),(18,34),(18,35),(18,36),(18,37),(18,38),(18,39),(18,40),(18,
41),(18,42),(19,33),(19,34),(19,35),(19,36),(19,37),(19,38),(19,39),(19,40),
(19,41),(19,42),(20,33),(20,34),(20,35),(20,36),(20,37),(20,38),(20,39),(20,
40),(20,41),(20,42),(29,33),(29,34),(29,35),(29,36),(29,37),(29,38),(29,39),
(29,40),(29,41),(29,42),(30,33),(30,34),(30,35),(30,36),(30,37),(30,38),(30,
39),(30,40),(30,41),(30,42),(31,33),(31,34),(31,35),(31,36),(31,37),(31,38),
(31,39),(31,40),(31,41),(31,42),(32,33),(32,34),(32,35),(32,36),(32,37),(32,
38),(32,39),(32,40),(32,41),(32,42),(21,35),(21,36),(21,37),(21,38),(21,39),
(21,40),(21,41),(21,42),(21,43),(21,44),(21,45),(21,46),(22,38),(22,39),(22,
40),(22,41),(22,42),(22,43),(23,38),(23,39),(23,40),(23,41),(23,42),(23,43),
(26,38),(26,39),(26,40),(26,41),(26,42),(26,43),(27,38),(27,39),(27,40),(27,
41),(27,42),(27,43),(28,35),(28,36),(28,37),(28,38),(28,39),(28,40),(28,41),
(28,42),(28,43),(28,44),(28,45),(28,46),(17,32),(18,32),(19,32),(20,32),(17,
31),(18,31),(19,31),(17,30),(18,30),(17,29),(29,32),(30,32),(31,32),(32,32),
(30,31),(31,31),(32,31),(31,30),(32,30),(32,29),(1,1),(1,2),(1,3),(1,4),(1,5),
(1,6),(2,1),(2,2),(2,3),(2,4),(2,5),(2,6),(3,1),(3,2),(3,3),(3,4),(3,5),(3,6),
(4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(5,1),(5,2),(5,3),(5,4),(5,5),(5,6),(6,1),
(6,2),(6,3),(6,4),(6,5),(6,6),(7,1),(7,2),(7,3),(7,4),(7,5),(7,6),(8,1),(8,2),
(8,3),(8,4),(8,5),(8,6),(9,1),(9,2),(9,3),(9,4),(9,5),(9,6),(10,1),(10,2),(10,
3),(10,4),(10,5),(10,6),(11,1),(11,2),(11,3),(11,4),(11,5),(11,6),(12,1),(12,
2),(12,3),(12,4),(12,5),(12,6),(13,1),(13,2),(13,3),(13,4),(13,5),(13,6),(14,
1),(14,2),(14,3),(14,4),(14,5),(14,6),(15,1),(15,2),(15,3),(15,4),(15,5),(15,
6),(16,1),(16,2),(16,3),(16,4),(16,5),(16,6),(17,1),(17,2),(17,3),(17,4),(17,
5),(17,6),(18,1),(18,2),(18,3),(18,4),(18,5),(18,6),(19,1),(19,2),(19,3),(19,
4),(19,5),(19,6),(20,1),(20,2),(20,3),(20,4),(20,5),(20,6),(1,7),(1,8),(1,9),
(1,10),(1,11),(1,12),(1,13),(1,14),(1,15),(1,16),(2,7),(2,8),(2,9),(2,10),(2,
11),(2,12),(2,13),(2,14),(2,15),(2,16),(3,7),(3,8),(3,9),(3,10),(3,11),(3,12),
(3,13),(3,14),(3,15),(3,16),(4,7),(4,8),(4,9),(4,10),(4,11),(4,12),(4,13),(4,
14),(4,15),(4,16),(5,7),(5,8),(5,9),(5,10),(5,11),(5,12),(5,13),(5,14),(5,15),
(5,16),(6,7),(6,8),(6,9),(6,10),(6,11),(6,12),(6,13),(6,14),(6,15),(6,16),(7,
7),(7,8),(7,9),(7,10),(7,11),(7,12),(7,13),(7,14),(7,15),(7,16),(8,7),(8,8),

(8,9),(8,10),(8,11),(8,12),(8,13),(8,14),(8,15),(8,16),(9,7),(9,8),(9,9),(9,10),(9,11),(9,12),(9,13),(9,14),(9,15),(9,16),(10,7),(10,8),(10,9),(10,10),(10,11),(10,12),(10,13),(10,14),(10,15),(10,16),(11,7),(11,8),(11,9),(11,10),(11,11),(11,12),(11,13),(11,14),(11,15),(11,16),(12,7),(12,8),(12,9),(12,10),(12,11),(12,12),(12,13),(12,14),(12,15),(12,16),(13,7),(13,8),(13,9),(13,10),(13,11),(13,12),(13,13),(13,14),(13,15),(13,16),(14,7),(14,8),(14,9),(14,10),(14,11),(14,12),(14,13),(14,14),(14,15),(14,16),(15,7),(15,8),(15,9),(15,10),(15,11),(15,12),(15,13),(15,14),(15,15),(15,16),(16,7),(16,8),(16,9),(16,10),(16,11),(16,12),(16,13),(16,14),(16,15),(16,16),(1,17),(1,18),(1,19),(1,20),(2,17),(2,18),(2,19),(2,20),(3,17),(3,18),(3,19),(3,20),(4,17),(4,18),(4,19),(4,20),(5,17),(5,18),(5,19),(5,20),(1,23),(1,24),(1,25),(1,26),(2,23),(2,24),(2,25),(2,26),(3,23),(3,24),(3,25),(3,26),(4,23),(4,24),(4,25),(4,26),(5,23),(5,24),(5,25),(5,26),(1,29),(1,30),(1,31),(1,32),(2,29),(2,30),(2,31),(2,32),(3,29),(3,30),(3,31),(3,32),(4,29),(4,30),(4,31),(4,32),(5,29),(5,30),(5,31),(5,32),(1,33),(1,34),(1,35),(1,36),(1,37),(1,38),(1,39),(1,40),(1,41),(1,42),(1,43),(1,44),(1,45),(1,46),(1,47),(1,48),(2,33),(2,34),(2,35),(2,36),(2,37),(2,38),(2,39),(2,40),(2,41),(2,42),(2,43),(2,44),(2,45),(2,46),(2,47),(2,48),(3,33),(3,34),(3,35),(3,36),(3,37),(3,38),(3,39),(3,40),(3,41),(3,42),(3,43),(3,44),(3,45),(3,46),(3,47),(3,48),(4,33),(4,34),(4,35),(4,36),(4,37),(4,38),(4,39),(4,40),(4,41),(4,42),(4,43),(4,44),(4,45),(4,46),(4,47),(4,48),(5,33),(5,34),(5,35),(5,36),(5,37),(5,38),(5,39),(5,40),(5,41),(5,42),(5,43),(5,44),(5,45),(5,46),(5,47),(5,48),(6,33),(6,34),(6,35),(6,36),(6,37),(6,38),(6,39),(6,40),(6,41),(6,42),(6,43),(6,44),(6,45),(6,46),(6,47),(6,48),(7,33),(7,34),(7,35),(7,36),(7,37),(7,38),(7,39),(7,40),(7,41),(7,42),(7,43),(7,44),(7,45),(7,46),(7,47),(7,48),(8,33),(8,34),(8,35),(8,36),(8,37),(8,38),(8,39),(8,40),(8,41),(8,42),(8,43),(8,44),(8,45),(8,46),(8,47),(8,48),(9,33),(9,34),(9,35),(9,36),(9,37),(9,38),(9,39),(9,40),(9,41),(9,42),(9,43),(9,44),(9,45),(9,46),(9,47),(9,48),(10,33),(10,34),(10,35),(10,36),(10,37),(10,38),(10,39),(10,40),(10,41),(10,42),(10,43),(10,44),(10,45),(10,46),(10,47),(10,48),(11,33),(11,34),(11,35),(11,36),(11,37),(11,38),(11,39),(11,40),(11,41),(11,42),(11,43),(11,44),(11,45),(11,46),(11,47),(11,48),(12,33),(12,34),(12,35),(12,36),(12,37),(12,38),(12,39),(12,40),(12,41),(12,42),(12,43),(12,44),(12,45),(12,46),(12,47),(12,48),(13,33),(13,34),(13,35),(13,36),(13,37),(13,38),(13,39),(13,40),(13,41),(13,42),(13,43),(13,44),(13,45),(13,46),(13,47),(13,48),(14,33),(14,34),(14,35),(14,36),(14,37),(14,38),(14,39),(14,40),(14,41),(14,42),(14,43),(14,44),(14,45),(14,46),(14,47),(14,48),(15,33),(15,34),(15,35),(15,36),(15,37),(15,38),(15,39),(15,40),(15,41),(15,42),(15,43),(15,44),(15,45),(15,46),(15,47),(15,48),(16,33),(16,34),(16,35),(16,36),(16,37),(16,38),(16,39),(16,40),(16,41),(16,42),(16,43),(16,44),(16,45),(16,46),(16,47),(16,48),(17,43),(17,44),(17,45),(17,46),(17,47),(17,48),(18,43),(18,44),(18,45),(18,46),(18,47),(18,48),(19,43),(19,44),(19,45),(19,46),(19,47),(19,48),(20,43),(20,44),(20,45),(20,46),(20,47),(20,48),(29,1),(29,2),(29,3),(29,4),(29,5),(29,6),(30,1),(30,2),(30,3),(30,4),(30,5),(30,6),(31,1),(31,2),(31,3),(31,4),(31,5),(31,6),(32,1),(32,2),(32,3),(32,4),(32,5),(32,6),(33,1),(33,2),(33,3),(33,4),(33,5),(33,6),(34,1),(34,2),(34,3),(34,4),(34,5),(34,6),(35,1),(35,2),(35,3),(35,4),(35,5),(35,6),(36,1),(36,2),(36,3),(36,4),(36,5),(36,6),(37,1),(37,2),(37,3),(37,4),(37,5),(37,6),(38,1),(38,2),(38,3),(38,4),(38,5),(38,6),(39,1),(39,2),(39,3),(39,4),(39,5),(39,6),(40,1),(40,2),(40,3),(40,4),(40,5),(40,6),(41,1),(41,2),(41,3),(41,4),(41,5),(41,6),(42,1),(42,2),(42,3),(42,4),(42,5),

(42,6),(43,1),(43,2),(43,3),(43,4),(43,5),(43,6),(44,1),(44,2),(44,3),(44,4),
(44,5),(44,6),(45,1),(45,2),(45,3),(45,4),(45,5),(45,6),(46,1),(46,2),(46,3),
(46,4),(46,5),(46,6),(47,1),(47,2),(47,3),(47,4),(47,5),(47,6),(48,1),(48,2),
(48,3),(48,4),(48,5),(48,6),(33,7),(33,8),(33,9),(33,10),(33,11),(33,12),(33,
13),(33,14),(33,15),(33,16),(34,7),(34,8),(34,9),(34,10),(34,11),(34,12),(34,
13),(34,14),(34,15),(34,16),(35,7),(35,8),(35,9),(35,10),(35,11),(35,12),(35,
13),(35,14),(35,15),(35,16),(36,7),(36,8),(36,9),(36,10),(36,11),(36,12),(36,
13),(36,14),(36,15),(36,16),(37,7),(37,8),(37,9),(37,10),(37,11),(37,12),(37,
13),(37,14),(37,15),(37,16),(38,7),(38,8),(38,9),(38,10),(38,11),(38,12),(38,
13),(38,14),(38,15),(38,16),(39,7),(39,8),(39,9),(39,10),(39,11),(39,12),(39,
13),(39,14),(39,15),(39,16),(40,7),(40,8),(40,9),(40,10),(40,11),(40,12),(40,
13),(40,14),(40,15),(40,16),(41,7),(41,8),(41,9),(41,10),(41,11),(41,12),(41,
13),(41,14),(41,15),(41,16),(42,7),(42,8),(42,9),(42,10),(42,11),(42,12),(42,
13),(42,14),(42,15),(42,16),(43,7),(43,8),(43,9),(43,10),(43,11),(43,12),(43,
13),(43,14),(43,15),(43,16),(44,7),(44,8),(44,9),(44,10),(44,11),(44,12),(44,
13),(44,14),(44,15),(44,16),(45,7),(45,8),(45,9),(45,10),(45,11),(45,12),(45,
13),(45,14),(45,15),(45,16),(46,7),(46,8),(46,9),(46,10),(46,11),(46,12),(46,
13),(46,14),(46,15),(46,16),(47,7),(47,8),(47,9),(47,10),(47,11),(47,12),(47,
13),(47,14),(47,15),(47,16),(48,7),(48,8),(48,9),(48,10),(48,11),(48,12),(48,
13),(48,14),(48,15),(48,16),(41,17),(41,18),(41,19),(41,20),(42,17),(42,18),
(42,19),(42,20),(43,17),(43,18),(43,19),(43,20),(44,17),(44,18),(44,19),(44,
20),(45,17),(45,18),(45,19),(45,20),(46,17),(46,18),(46,19),(46,20),(47,17),
(47,18),(47,19),(47,20),(48,17),(48,18),(48,19),(48,20),(41,24),(41,25),(42,
24),(42,25),(43,24),(43,25),(44,24),(44,25),(45,24),(45,25),(46,24),(46,25),
(47,24),(47,25),(48,24),(48,25),(41,29),(41,30),(41,31),(41,32),(42,29),(42,
30),(42,31),(42,32),(43,29),(43,30),(43,31),(43,32),(44,29),(44,30),(44,31),
(44,32),(45,29),(45,30),(45,31),(45,32),(46,29),(46,30),(46,31),(46,32),(47,
29),(47,30),(47,31),(47,32),(48,29),(48,30),(48,31),(48,32),(33,33),(33,34),
(33,35),(33,36),(33,37),(33,38),(33,39),(33,40),(33,41),(33,42),(34,33),(34,
34),(34,35),(34,36),(34,37),(34,38),(34,39),(34,40),(34,41),(34,42),(35,33),
(35,34),(35,35),(35,36),(35,37),(35,38),(35,39),(35,40),(35,41),(35,42),(36,
33),(36,34),(36,35),(36,36),(36,37),(36,38),(36,39),(36,40),(36,41),(36,42),
(37,33),(37,34),(37,35),(37,36),(37,37),(37,38),(37,39),(37,40),(37,41),(37,
42),(38,33),(38,34),(38,35),(38,36),(38,37),(38,38),(38,39),(38,40),(38,41),
(38,42),(39,33),(39,34),(39,35),(39,36),(39,37),(39,38),(39,39),(39,40),(39,
41),(39,42),(40,33),(40,34),(40,35),(40,36),(40,37),(40,38),(40,39),(40,40),
(40,41),(40,42),(41,33),(41,34),(41,35),(41,36),(41,37),(41,38),(41,39),(41,
40),(41,41),(41,42),(42,33),(42,34),(42,35),(42,36),(42,37),(42,38),(42,39),
(42,40),(42,41),(42,42),(43,33),(43,34),(43,35),(43,36),(43,37),(43,38),(43,
39),(43,40),(43,41),(43,42),(44,33),(44,34),(44,35),(44,36),(44,37),(44,38),
(44,39),(44,40),(44,41),(44,42),(45,33),(45,34),(45,35),(45,36),(45,37),(45,
38),(45,39),(45,40),(45,41),(45,42),(46,33),(46,34),(46,35),(46,36),(46,37),
(46,38),(46,39),(46,40),(46,41),(46,42),(47,33),(47,34),(47,35),(47,36),(47,
37),(47,38),(47,39),(47,40),(47,41),(47,42),(48,33),(48,34),(48,35),(48,36),
(48,37),(48,38),(48,39),(48,40),(48,41),(48,42),(29,43),(29,44),(29,45),(29,
46),(29,47),(29,48),(30,43),(30,44),(30,45),(30,46),(30,47),(30,48),(31,43),
(31,44),(31,45),(31,46),(31,47),(31,48),(32,43),(32,44),(32,45),(32,46),(32,
47),(32,48),(33,43),(33,44),(33,45),(33,46),(33,47),(33,48),(34,43),(34,44),
(34,45),(34,46),(34,47),(34,48),(35,43),(35,44),(35,45),(35,46),(35,47),(35,

```
48),(36,43),(36,44),(36,45),(36,46),(36,47),(36,48),(37,43),(37,44),(37,45),
(37,46),(37,47),(37,48),(38,43),(38,44),(38,45),(38,46),(38,47),(38,48),(39,
43),(39,44),(39,45),(39,46),(39,47),(39,48),(40,43),(40,44),(40,45),(40,46),
(40,47),(40,48),(41,43),(41,44),(41,45),(41,46),(41,47),(41,48),(42,43),(42,
44),(42,45),(42,46),(42,47),(42,48),(43,43),(43,44),(43,45),(43,46),(43,47),
(43,48),(44,43),(44,44),(44,45),(44,46),(44,47),(44,48),(45,43),(45,44),(45,
45),(45,46),(45,47),(45,48),(46,43),(46,44),(46,45),(46,46),(46,47),(46,48),
(47,43),(47,44),(47,45),(47,46),(47,47),(47,48),(48,43),(48,44),(48,45),(48,
46),(48,47),(48,48)
```

```
    def move_knife(self, action):
        if action == 4 or (action, self.previous_action) in forbidden_moves:
            action = self.previous_action
        else:
            self.previous_action = action
        head = self.knife[0]
        if action == 0:
            p = (head[0] - 1, head[1])
        elif action == 1:
            p = (head[0] + 1, head[1])
        elif action == 2:
            p = (head[0], head[1] - 1)
        elif action == 3:
            p = (head[0], head[1] + 1)
        self.knife.insert(0, p)
        self.knife.pop()

    def get_state(self):
        canvas = np.ones((self.grid_size, ) * 2)
        canvas[1:-1, 1:-1] = 0.0
        for seg in self.knife:
            canvas[seg[0], seg[1]] = 0.8
        for seg in self.art_border:
            canvas[seg[0],seg[1]] = 0.9
        for seg in self.mark_border:
            canvas[seg[0], seg[1]] = 0.5
        for seg in self.ut_border:
            canvas[seg[0],seg[1]] = 0.7
        for seg in self.per_border:
            canvas[seg[0],seg[1]] = 0.1
        return canvas

    def get_score(self):
        if self.game_over:
            score = -1
        elif self.plus_scored:
            score = +1
        elif self.minus_scored:
            score = -1
```

```
    elif self.non_scored:
        score = 0
    else:
        score = -1
    return score

def reset(self):
    grid_size = self.grid_size
    knife_length = self.knife_length
    self.knife = [(10,24)]
    self.game_over = False
    self.plus_scored = False
    self.minus_scored = False
    self.non_scored = False
    self.arrange()
    if np.random.randint(2) == 0:
        self.previous_action = 0
    else:
        self.previous_action = 1
    self.border = []
    for z in range(grid_size):
      self.border += [(z, 0), (z, grid_size - 1), (0, z), (grid_size - 1, z)]
    self.ut_border = []
    self.ut_border += self.uterus
    self.art_border = []
    self.art_border += self.artery
    self.mark_border = []
    self.mark_border += self.marker
    self.per_border = []
    self.per_border += self.peritoneum

def left(self):
    self.play(0)
def right(self):
    self.play(1)
def up(self):
    self.play(2)
def down(self):
    self.play(3)
def idle(self):
    self.play(4)

def hit_border(self):
   return self.knife[0] in self.border or self.knife[0] in self.ut_border
def hit_artery(self):
    return self.knife[0] in self.art_border
def hit_marker(self):
    return self.knife[0] in self.mark_border
```

```
    def hit_peritoneum(self):
        return self.knife[0] in self.per_border

    def is_over(self):
        return self.hit_border()
    def is_won(self):
        return len(self.mark_border) < 1


# game3_test.py

from keras.models import Sequential
from keras.layers import *
from game3 import OP
from keras.optimizers import *
from agent import Agent

grid_size = 50
nb_frames = 5
nb_actions = 5

model = Sequential()
model.add(Convolution2D(16, 2, 2, activation='relu', input_shape=(nb_frames,
grid_size, grid_size)))
model.add(Convolution2D(32, 3, 3, activation='relu'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(nb_actions))
model.compile(RMSprop(), 'MSE')

operation = OP(grid_size)

agent = Agent(model=model, memory_size=5, nb_frames=nb_frames)
agent.train(operation, batch_size=64, nb_epoch=500, gamma=0.80)
agent.play(operation)


# game3.py

__original_author__ = "Fariz Rahman"
# Modified by "Masakazu Sato" on 10th December,2016.
import numpy as np
from game import Game


actions = {0:'left', 1:'right', 2:'up', 3:'down', 4:'idle'}
forbidden_moves = [(0, 1), (1, 0), (2, 3), (3, 2)]

class OP(Game):

    def __init__(self, grid_size=50, knife_length=1):
        self.grid_size = grid_size
```

```
        self.knife_length = knife_length
        self.reset()
        self.state_changed = True

    @property
    def name(self):
        return "OP"
    @property
    def nb_actions(self):
        return 5

    def play(self, action):
        assert action in range(5), "Invalid action."
        self.plus_scored = False
        self.minus_scored = False
        self.non_scored = False
        self.move_knife(action)
        tip = self.knife[0]

        if self.hit_artery():
            self.minus_scored = True
            self.art_border.remove(tip)
        elif self.hit_marker():
            self.plus_scored = True
            self.mark_border.remove(tip)
        elif self.hit_peritoneum():
            self.non_scored = True
            self.per_border.remove(tip)
        elif self.hit_border():
            self.game_over = True
        else:
            self.minus_scored = True


    def arrange(self):
       self.artery = (1,21),(1,22),(2,21),(2,22),(3,21),(3,22),(4,21),(4,22),
(5,21),(5,22),(6,21),(6,22),(7,21),(7,22),(8,21),(8,22),(1,27),(1,28),(2,
27),(2,28),(3,27),(3,28),(4,27),(4,28),(5,27),(5,28),(6,27),(6,28),(7,27),
(7,28),(8,27),(8,28),(11,21),(11,22),(11,23),(12,21),(12,22),(12,23),(13,
21),(13,22),(13,23),(14,21),(14,22),(14,23),(15,21),(15,22),(15,23),(16,21),
(16,22),(16,23),(11,26),(11,27),(11,28),(12,26),(12,27),(12,28),(13,26),(13,
27),(13,28),(14,26),(14,27),(14,28),(15,26),(15,27),(15,28),(16,26),(16,27),
(16,28),(21,1),(21,2),(22,1),(22,2),(23,1),(23,2),(24,1),(24,2),(25,1),(25,
2),(26,1),(26,2),(27,1),(27,2),(28,1),(28,2),(23,3),(23,4),(23,5),(24,3),
(24,4),(24,5),(25,3),(25,4),(25,5),(26,3),(26,4),(26,5),(24,6),(24,7),(24,
8),(24,9),(25,6),(25,7),(25,8),(25,9),(23,12),(23,13),(23,14),(24,12),(24,
13),(24,14),(25,12),(25,13),(25,14),(26,12),(26,13),(26,14),(21,15),(21,16),
(22,15),(22,16),(23,15),(23,16),(24,15),(24,16),(25,15),(25,16),(26,15),(26,
16),(27,15),(27,16),(28,15),(28,16),(21,33),(21,34),(22,33),(22,34),(23,33),
(23,34),(24,33),(24,34),(25,33),(25,34),(26,33),(26,34),(27,33),(27,34),(28,
33),(28,34),(23,35),(23,36),(23,37),(24,35),(24,36),(24,37),(25,35),(25,36),
```

```
(25,37),(26,35),(26,36),(26,37),(24,40),(24,41),(24,42),(24,43),(25,40),(25,
41),(25,42),(25,43),(23,44),(23,45),(23,46),(24,44),(24,45),(24,46),(25,44),
(25,45),(25,46),(26,44),(26,45),(26,46),(21,47),(21,48),(22,47),(22,48),(23,
47),(23,48),(24,47),(24,48),(25,47),(25,48),(26,47),(26,48),(27,47),(27,48),
(28,47),(28,48),(33,21),(33,22),(33,23),(34,21),(34,22),(34,23),(35,21),(35,
22),(35,23),(40,21),(40,22),(40,23),(41,21),(41,22),(41,23),(42,21),(42,22),
(42,23),(43,21),(43,22),(43,23),(44,21),(44,22),(44,23),(45,21),(45,22),(45,
23),(46,21),(46,22),(46,23),(47,21),(47,22),(47,23),(48,21),(48,22),(48,23),
(33,26),(33,27),(33,28),(34,26),(34,27),(34,28),(35,26),(35,27),(35,28),(40,
26),(40,27),(40,28),(41,26),(41,27),(41,28),(42,26),(42,27),(42,28),(43,26),
(43,27),(43,28),(44,26),(44,27),(44,28),(45,26),(45,27),(45,28),(46,26),(46,
27),(46,28),(47,26),(47,27),(47,28),(48,26),(48,27),(48,28)
      self.marker = (9,21),(9,22),(10,21),(10,22),(9,27),(9,28),(10,27),(10,
28),(24,10),(24,11),(25,10),(25,11),(24,38),(24,39),(25,38),(25,39),(36,21),
(36,22),(36,23),(37,21),(37,22),(37,23),(38,21),(38,22),(38,23),(39,21),(39,
22),(39,23),(36,26),(36,27),(36,28),(37,26),(37,27),(37,28),(38,26),(38,27),
(38,28),(39,26),(39,27),(39,28)
      self.uterus = (21,17),(22,17),(23,17),(24,17),(25,17),(26,17),(27,17),
(28,17),(20,18),(21,18),(22,18),(23,18),(24,18),(25,18),(26,18),(27,18),(28,
18),(29,18),(19,19),(20,19),(21,19),(22,19),(23,19),(24,19),(25,19),(26,19),
(27,19),(28,19),(29,19),(30,19),(18,20),(19,20),(20,20),(21,20),(22,20),(23,
20),(24,20),(25,20),(26,20),(27,20),(28,20),(29,20),(30,20),(31,20),(17,21),
(17,22),(17,23),(17,24),(17,25),(17,26),(17,27),(17,28),(18,21),(18,22),(18,
23),(18,24),(18,25),(18,26),(18,27),(18,28),(19,21),(19,22),(19,23),(19,24),
(19,25),(19,26),(19,27),(19,28),(20,21),(20,22),(20,23),(20,24),(20,25),(20,
26),(20,27),(20,28),(21,21),(21,22),(21,23),(21,24),(21,25),(21,26),(21,27),
(21,28),(22,21),(22,22),(22,23),(22,24),(22,25),(22,26),(22,27),(22,28),(23,
21),(23,22),(23,23),(23,24),(23,25),(23,26),(23,27),(23,28),(24,21),(24,22),
(24,23),(24,24),(24,25),(24,26),(24,27),(24,28),(25,21),(25,22),(25,23),(25,
24),(25,25),(25,26),(25,27),(25,28),(26,21),(26,22),(26,23),(26,24),(26,25),
(26,26),(26,27),(26,28),(27,21),(27,22),(27,23),(27,24),(27,25),(27,26),(27,
27),(27,28),(28,21),(28,22),(28,23),(28,24),(28,25),(28,26),(28,27),(28,28),
(29,21),(29,22),(29,23),(29,24),(29,25),(29,26),(29,27),(29,28),(30,21),(30,
22),(30,23),(30,24),(30,25),(30,26),(30,27),(30,28),(31,21),(31,22),(31,23),
(31,24),(31,25),(31,26),(31,27),(31,28),(32,21),(32,22),(32,23),(32,24),(32,
25),(32,26),(32,27),(32,28),(18,29),(19,29),(20,29),(21,29),(22,29),(23,29),
(24,29),(25,29),(26,29),(27,29),(28,29),(29,29),(30,29),(31,29),(19,30),(20,
30),(21,30),(22,30),(23,30),(24,30),(25,30),(26,30),(27,30),(28,30),(29,30),
(30,30),(20,31),(21,31),(22,31),(23,31),(24,31),(25,31),(26,31),(27,31),(28,
31),(29,31),(21,32),(22,32),(23,32),(24,32),(25,32),(26,32),(27,32),(28,32)
     self.peritoneum = (22,3),(22,4),(22,5),(27,3),(27,4),(27,5),(22,35),(22,
36),(22,37),(27,35),(27,36),(27,37),(22,44),(22,45),(22,46),(27,44),(27,45),
(27,46),(6,17),(6,18),(6,19),(6,20),(7,17),(7,18),(7,19),(7,20),(8,17),(8,
18),(8,19),(8,20),(9,17),(9,18),(9,19),(9,20),(10,17),(10,18),(10,19),(10,
20),(11,17),(11,18),(11,19),(11,20),(12,17),(12,18),(12,19),(12,20),(13,17),
(13,18),(13,19),(13,20),(14,17),(14,18),(14,19),(14,20),(15,17),(15,18),(15,
19),(15,20),(16,17),(16,18),(16,19),(16,20),(17,7),(17,8),(17,9),(17,10),
(17,11),(17,12),(17,13),(17,14),(17,15),(17,16),(18,7),(18,8),(18,9),(18,
10),(18,11),(18,12),(18,13),(18,14),(18,15),(18,16),(19,7),(19,8),(19,9),
```

```
(19,10),(19,11),(19,12),(19,13),(19,14),(19,15),(19,16),(20,7),(20,8),(20,
9),(20,10),(20,11),(20,12),(20,13),(20,14),(20,15),(20,16),(17,17),(18,17),
(19,17),(20,17),(17,18),(18,18),(19,18),(17,19),(18,19),(17,20),(21,3),(21,
4),(21,5),(21,6),(21,7),(21,8),(21,9),(21,10),(21,11),(21,12),(21,13),(21,
14),(22,6),(22,7),(22,8),(22,9),(22,10),(22,11),(22,12),(22,13),(22,14),(23,
6),(23,7),(23,8),(23,9),(23,10),(23,11),(26,6),(26,7),(26,8),(26,9),(26,10),
(26,11),(27,6),(27,7),(27,8),(27,9),(27,10),(27,11),(27,12),(27,13),(27,14),
(28,3),(28,4),(28,5),(28,6),(28,7),(28,8),(28,9),(28,10),(28,11),(28,12),
(28,13),(28,14),(29,7),(29,8),(29,9),(29,10),(29,11),(29,12),(29,13),(29,
14),(29,15),(29,16),(30,7),(30,8),(30,9),(30,10),(30,11),(30,12),(30,13),
(30,14),(30,15),(30,16),(31,7),(31,8),(31,9),(31,10),(31,11),(31,12),(31,
13),(31,14),(31,15),(31,16),(32,7),(32,8),(32,9),(32,10),(32,11),(32,12),
(32,13),(32,14),(32,15),(32,16),(33,17),(33,18),(33,19),(33,20),(34,17),(34,
18),(34,19),(34,20),(35,17),(35,18),(35,19),(35,20),(36,17),(36,18),(36,19),
(36,20),(37,17),(37,18),(37,19),(37,20),(38,17),(38,18),(38,19),(38,20),(39,
17),(39,18),(39,19),(39,20),(40,17),(40,18),(40,19),(40,20),(29,17),(30,17),
(31,17),(32,17),(30,18),(31,18),(32,18),(31,19),(32,19),(32,20),(6,23),(6,
24),(6,25),(6,26),(7,23),(7,24),(7,25),(7,26),(8,23),(8,24),(8,25),(8,26),
(9,23),(9,24),(9,25),(9,26),(10,23),(10,24),(10,25),(10,26),(11,24),(11,25),
(12,24),(12,25),(13,24),(13,25),(14,24),(14,25),(15,24),(15,25),(16,24),(16,
25),(33,24),(33,25),(34,24),(34,25),(35,24),(35,25),(36,24),(36,25),(37,24),
(37,25),(38,24),(38,25),(39,24),(39,25),(40,24),(40,25),(6,29),(6,30),(6,
31),(6,32),(7,29),(7,30),(7,31),(7,32),(8,29),(8,30),(8,31),(8,32),(9,29),
(9,30),(9,31),(9,32),(10,29),(10,30),(10,31),(10,32),(11,29),(11,30),(11,
31),(11,32),(12,29),(12,30),(12,31),(12,32),(13,29),(13,30),(13,31),(13,32),
(14,29),(14,30),(14,31),(14,32),(15,29),(15,30),(15,31),(15,32),(16,29),(16,
30),(16,31),(16,32),(33,29),(33,30),(33,31),(33,32),(34,29),(34,30),(34,31),
(34,32),(35,29),(35,30),(35,31),(35,32),(36,29),(36,30),(36,31),(36,32),(37,
29),(37,30),(37,31),(37,32),(38,29),(38,30),(38,31),(38,32),(39,29),(39,30),
(39,31),(39,32),(40,29),(40,30),(40,31),(40,32),(17,33),(17,34),(17,35),(17,
36),(17,37),(17,38),(17,39),(17,40),(17,41),(17,42),(18,33),(18,34),(18,35),
(18,36),(18,37),(18,38),(18,39),(18,40),(18,41),(18,42),(19,33),(19,34),(19,
35),(19,36),(19,37),(19,38),(19,39),(19,40),(19,41),(19,42),(20,33),(20,34),
(20,35),(20,36),(20,37),(20,38),(20,39),(20,40),(20,41),(20,42),(29,33),(29,
34),(29,35),(29,36),(29,37),(29,38),(29,39),(29,40),(29,41),(29,42),(30,33),
(30,34),(30,35),(30,36),(30,37),(30,38),(30,39),(30,40),(30,41),(30,42),(31,
33),(31,34),(31,35),(31,36),(31,37),(31,38),(31,39),(31,40),(31,41),(31,42),
(32,33),(32,34),(32,35),(32,36),(32,37),(32,38),(32,39),(32,40),(32,41),(32,
42),(21,35),(21,36),(21,37),(21,38),(21,39),(21,40),(21,41),(21,42),(21,43),
(21,44),(21,45),(21,46),(22,38),(22,39),(22,40),(22,41),(22,42),(22,43),(23,
38),(23,39),(23,40),(23,41),(23,42),(23,43),(26,38),(26,39),(26,40),(26,41),
(26,42),(26,43),(27,38),(27,39),(27,40),(27,41),(27,42),(27,43),(28,35),(28,
36),(28,37),(28,38),(28,39),(28,40),(28,41),(28,42),(28,43),(28,44),(28,45),
(28,46),(17,32),(18,32),(19,32),(20,32),(17,31),(18,31),(19,31),(17,30),(18,
30),(17,29),(29,32),(30,32),(31,32),(32,32),(30,31),(31,31),(32,31),(31,30),
(32,30),(32,29)
        self.preborder = (1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,1),(2,2),(2,
3),(2,4),(2,5),(2,6),(3,1),(3,2),(3,3),(3,4),(3,5),(3,6),(4,1),(4,2),(4,3),
(4,4),(4,5),(4,6),(5,1),(5,2),(5,3),(5,4),(5,5),(5,6),(6,1),(6,2),(6,3),(6,
```

4),(6,5),(6,6),(7,1),(7,2),(7,3),(7,4),(7,5),(7,6),(8,1),(8,2),(8,3),(8,4),
(8,5),(8,6),(9,1),(9,2),(9,3),(9,4),(9,5),(9,6),(10,1),(10,2),(10,3),(10,4),
(10,5),(10,6),(11,1),(11,2),(11,3),(11,4),(11,5),(11,6),(12,1),(12,2),(12,
3),(12,4),(12,5),(12,6),(13,1),(13,2),(13,3),(13,4),(13,5),(13,6),(14,1),
(14,2),(14,3),(14,4),(14,5),(14,6),(15,1),(15,2),(15,3),(15,4),(15,5),(15,
6),(16,1),(16,2),(16,3),(16,4),(16,5),(16,6),(17,1),(17,2),(17,3),(17,4),
(17,5),(17,6),(18,1),(18,2),(18,3),(18,4),(18,5),(18,6),(19,1),(19,2),(19,
3),(19,4),(19,5),(19,6),(20,1),(20,2),(20,3),(20,4),(20,5),(20,6),(1,7),(1,
8),(1,9),(1,10),(1,11),(1,12),(1,13),(1,14),(1,15),(1,16),(2,7),(2,8),(2,9),
(2,10),(2,11),(2,12),(2,13),(2,14),(2,15),(2,16),(3,7),(3,8),(3,9),(3,10),
(3,11),(3,12),(3,13),(3,14),(3,15),(3,16),(4,7),(4,8),(4,9),(4,10),(4,11),
(4,12),(4,13),(4,14),(4,15),(4,16),(5,7),(5,8),(5,9),(5,10),(5,11),(5,12),
(5,13),(5,14),(5,15),(5,16),(6,7),(6,8),(6,9),(6,10),(6,11),(6,12),(6,13),
(6,14),(6,15),(6,16),(7,7),(7,8),(7,9),(7,10),(7,11),(7,12),(7,13),(7,14),
(7,15),(7,16),(8,7),(8,8),(8,9),(8,10),(8,11),(8,12),(8,13),(8,14),(8,15),
(8,16),(9,7),(9,8),(9,9),(9,10),(9,11),(9,12),(9,13),(9,14),(9,15),(9,16),
(10,7),(10,8),(10,9),(10,10),(10,11),(10,12),(10,13),(10,14),(10,15),(10,
16),(11,7),(11,8),(11,9),(11,10),(11,11),(11,12),(11,13),(11,14),(11,15),
(11,16),(12,7),(12,8),(12,9),(12,10),(12,11),(12,12),(12,13),(12,14),(12,
15),(12,16),(13,7),(13,8),(13,9),(13,10),(13,11),(13,12),(13,13),(13,14),
(13,15),(13,16),(14,7),(14,8),(14,9),(14,10),(14,11),(14,12),(14,13),(14,
14),(14,15),(14,16),(15,7),(15,8),(15,9),(15,10),(15,11),(15,12),(15,13),
(15,14),(15,15),(15,16),(16,7),(16,8),(16,9),(16,10),(16,11),(16,12),(16,
13),(16,14),(16,15),(16,16),(1,17),(1,18),(1,19),(1,20)  ,(2,17),(2,18),(2,
19),(2,20),(3,17),(3,18),(3,19),(3,20),(4,17),(4,18),(4,19),(4,20),(5,17),
(5,18),(5,19),(5,20),(1,23),(1,24),(1,25),(1,26),(2,23),(2,24),(2,25),(2,
26),(3,23),(3,24),(3,25),(3,26),(4,23),(4,24),(4,25),(4,26),(5,23),(5,24),
(5,25),(5,26),(1,29),(1,30),(1,31),(1,32),(2,29),(2,30),(2,31),(2,32),(3,
29),(3,30),(3,31),(3,32),(4,29),(4,30),(4,31),(4,32),(5,29),(5,30),(5,31),
(5,32),(1,33),(1,34),(1,35),(1,36),(1,37),(1,38),(1,39),(1,40),(1,41),(1,
42),(1,43),(1,44),(1,45),(1,46),(1,47),(1,48),(2,33),(2,34),(2,35),(2,36),
(2,37),(2,38),(2,39),(2,40),(2,41),(2,42),(2,43),(2,44),(2,45),(2,46),(2,
47),(2,48),(3,33),(3,34),(3,35),(3,36),(3,37),(3,38),(3,39),(3,40),(3,41),
(3,42),(3,43),(3,44),(3,45),(3,46),(3,47),(3,48),(4,33),(4,34),(4,35),(4,
36),(4,37),(4,38),(4,39),(4,40),(4,41),(4,42),(4,43),(4,44),(4,45),(4,46),
(4,47),(4,48),(5,33),(5,34),(5,35),(5,36),(5,37),(5,38),(5,39),(5,40),(5,
41),(5,42),(5,43),(5,44),(5,45),(5,46),(5,47),(5,48),(6,33),(6,34),(6,35),
(6,36),(6,37),(6,38),(6,39),(6,40),(6,41),(6,42),(6,43),(6,44),(6,45),(6,
46),(6,47),(6,48),(7,33),(7,34),(7,35),(7,36),(7,37),(7,38),(7,39),(7,40),
(7,41),(7,42),(7,43),(7,44),(7,45),(7,46),(7,47),(7,48),(8,33),(8,34),(8,
35),(8,36),(8,37),(8,38),(8,39),(8,40),(8,41),(8,42),(8,43),(8,44),(8,45),
(8,46),(8,47),(8,48),(9,33),(9,34),(9,35),(9,36),(9,37),(9,38),(9,39),(9,
40),(9,41),(9,42),(9,43),(9,44),(9,45),(9,46),(9,47),(9,48),(10,33),(10,34),
(10,35),(10,36),(10,37),(10,38),(10,39),(10,40),(10,41),(10,42),(10,43),(10,
44),(10,45),(10,46),(10,47),(10,48),(11,33),(11,34),(11,35),(11,36),(11,37),
(11,38),(11,39),(11,40),(11,41),(11,42),(11,43),(11,44),(11,45),(11,46),(11,
47),(11,48),(12,33),(12,34),(12,35),(12,36),(12,37),(12,38),(12,39),(12,40),
(12,41),(12,42),(12,43),(12,44),(12,45),(12,46),(12,47),(12,48),(13,33),(13,
34),(13,35),(13,36),(13,37),(13,38),(13,39),(13,40),(13,41),(13,42),(13,43),

(13,44),(13,45),(13,46),(13,47),(13,48),(14,33),(14,34),(14,35),(14,36),(14,37),(14,38),(14,39),(14,40),(14,41),(14,42),(14,43),(14,44),(14,45),(14,46),(14,47),(14,48),(15,33),(15,34),(15,35),(15,36),(15,37),(15,38),(15,39),(15,40),(15,41),(15,42),(15,43),(15,44),(15,45),(15,46),(15,47),(15,48),(16,33),(16,34),(16,35),(16,36),(16,37),(16,38),(16,39),(16,40),(16,41),(16,42),(16,43),(16,44),(16,45),(16,46),(16,47),(16,48),(17,43),(17,44),(17,45),(17,46),(17,47),(17,48),(18,43),(18,44),(18,45),(18,46),(18,47),(18,48),(19,43),(19,44),(19,45),(19,46),(19,47),(19,48),(20,43),(20,44),(20,45),(20,46),(20,47),(20,48),(29,1),(29,2),(29,3),(29,4),(29,5),(29,6),(30,1),(30,2),(30,3),(30,4),(30,5),(30,6),(31,1),(31,2),(31,3),(31,4),(31,5),(31,6),(32,1),(32,2),(32,3),(32,4),(32,5),(32,6),(33,1),(33,2),(33,3),(33,4),(33,5),(33,6),(34,1),(34,2),(34,3),(34,4),(34,5),(34,6),(35,1),(35,2),(35,3),(35,4),(35,5),(35,6),(36,1),(36,2),(36,3),(36,4),(36,5),(36,6),(37,1),(37,2),(37,3),(37,4),(37,5),(37,6),(38,1),(38,2),(38,3),(38,4),(38,5),(38,6),(39,1),(39,2),(39,3),(39,4),(39,5),(39,6),(40,1),(40,2),(40,3),(40,4),(40,5),(40,6),(41,1),(41,2),(41,3),(41,4),(41,5),(41,6),(42,1),(42,2),(42,3),(42,4),(42,5),(42,6),(43,1),(43,2),(43,3),(43,4),(43,5),(43,6),(44,1),(44,2),(44,3),(44,4),(44,5),(44,6),(45,1),(45,2),(45,3),(45,4),(45,5),(45,6),(46,1),(46,2),(46,3),(46,4),(46,5),(46,6),(47,1),(47,2),(47,3),(47,4),(47,5),(47,6),(48,1),(48,2),(48,3),(48,4),(48,5),(48,6),(33,7),(33,8),(33,9),(33,10),(33,11),(33,12),(33,13),(33,14),(33,15),(33,16),(34,7),(34,8),(34,9),(34,10),(34,11),(34,12),(34,13),(34,14),(34,15),(34,16),(35,7),(35,8),(35,9),(35,10),(35,11),(35,12),(35,13),(35,14),(35,15),(35,16),(36,7),(36,8),(36,9),(36,10),(36,11),(36,12),(36,13),(36,14),(36,15),(36,16),(37,7),(37,8),(37,9),(37,10),(37,11),(37,12),(37,13),(37,14),(37,15),(37,16),(38,7),(38,8),(38,9),(38,10),(38,11),(38,12),(38,13),(38,14),(38,15),(38,16),(39,7),(39,8),(39,9),(39,10),(39,11),(39,12),(39,13),(39,14),(39,15),(39,16),(40,7),(40,8),(40,9),(40,10),(40,11),(40,12),(40,13),(40,14),(40,15),(40,16),(41,7),(41,8),(41,9),(41,10),(41,11),(41,12),(41,13),(41,14),(41,15),(41,16),(42,7),(42,8),(42,9),(42,10),(42,11),(42,12),(42,13),(42,14),(42,15),(42,16),(43,7),(43,8),(43,9),(43,10),(43,11),(43,12),(43,13),(43,14),(43,15),(43,16),(44,7),(44,8),(44,9),(44,10),(44,11),(44,12),(44,13),(44,14),(44,15),(44,16),(45,7),(45,8),(45,9),(45,10),(45,11),(45,12),(45,13),(45,14),(45,15),(45,16),(46,7),(46,8),(46,9),(46,10),(46,11),(46,12),(46,13),(46,14),(46,15),(46,16),(47,7),(47,8),(47,9),(47,10),(47,11),(47,12),(47,13),(47,14),(47,15),(47,16),(48,7),(48,8),(48,9),(48,10),(48,11),(48,12),(48,13),(48,14),(48,15),(48,16),(41,17),(41,18),(41,19),(41,20),(42,17),(42,18),(42,19),(42,20),(43,17),(43,18),(43,19),(43,20),(44,17),(44,18),(44,19),(44,20),(45,17),(45,18),(45,19),(45,20),(46,17),(46,18),(46,19),(46,20),(47,17),(47,18),(47,19),(47,20),(48,17),(48,18),(48,19),(48,20),(41,24),(41,25),(42,24),(42,25),(43,24),(43,25),(44,24),(44,25),(45,24),(45,25),(46,24),(46,25),(47,24),(47,25),(48,24),(48,25),(41,29),(41,30),(41,31),(41,32),(42,29),(42,30),(42,31),(42,32),(43,29),(43,30),(43,31),(43,32),(44,29),(44,30),(44,31),(44,32),(45,29),(45,30),(45,31),(45,32),(46,29),(46,30),(46,31),(46,32),(47,29),(47,30),(47,31),(47,32),(48,29),(48,30),(48,31),(48,32),(33,33),(33,34),(33,35),(33,36),(33,37),(33,38),(33,39),(33,40),(33,41),(33,42),(34,33),(34,34),(34,35),(34,36),(34,37),(34,38),(34,39),(34,40),(34,41),(34,42),(35,33),(35,34),(35,35),(35,36),(35,37),(35,38),(35,39),(35,40),(35,41),(35,42),(36,33),(36,34),(36,35),(36,36),(36,37),(36,38),(36,39),(36,40),(36,41),(36,

```
42),(37,33),(37,34),(37,35),(37,36),(37,37),(37,38),(37,39),(37,40),(37,41),
(37,42),(38,33),(38,34),(38,35),(38,36),(38,37),(38,38),(38,39),(38,40),(38,
41),(38,42),(39,33),(39,34),(39,35),(39,36),(39,37),(39,38),(39,39),(39,40),
(39,41),(39,42),(40,33),(40,34),(40,35),(40,36),(40,37),(40,38),(40,39),(40,
40),(40,41),(40,42),(41,33),(41,34),(41,35),(41,36),(41,37),(41,38),(41,39),
(41,40),(41,41),(41,42),(42,33),(42,34),(42,35),(42,36),(42,37),(42,38),(42,
39),(42,40),(42,41),(42,42),(43,33),(43,34),(43,35),(43,36),(43,37),(43,38),
(43,39),(43,40),(43,41),(43,42),(44,33),(44,34),(44,35),(44,36),(44,37),(44,
38),(44,39),(44,40),(44,41),(44,42),(45,33),(45,34),(45,35),(45,36),(45,37),
(45,38),(45,39),(45,40),(45,41),(45,42),(46,33),(46,34),(46,35),(46,36),(46,
37),(46,38),(46,39),(46,40),(46,41),(46,42),(47,33),(47,34),(47,35),(47,36),
(47,37),(47,38),(47,39),(47,40),(47,41),(47,42),(48,33),(48,34),(48,35),(48,
36),(48,37),(48,38),(48,39),(48,40),(48,41),(48,42),(29,43),(29,44),(29,45),
(29,46),(29,47),(29,48),(30,43),(30,44),(30,45),(30,46),(30,47),(30,48),(31,
43),(31,44),(31,45),(31,46),(31,47),(31,48),(32,43),(32,44),(32,45),(32,46),
(32,47),(32,48),(33,43),(33,44),(33,45),(33,46),(33,47),(33,48),(34,43),(34,
44),(34,45),(34,46),(34,47),(34,48),(35,43),(35,44),(35,45),(35,46),(35,47),
(35,48),(36,43),(36,44),(36,45),(36,46),(36,47),(36,48),(37,43),(37,44),(37,
45),(37,46),(37,47),(37,48),(38,43),(38,44),(38,45),(38,46),(38,47),(38,48),
(39,43),(39,44),(39,45),(39,46),(39,47),(39,48),(40,43),(40,44),(40,45),(40,
46),(40,47),(40,48),(41,43),(41,44),(41,45),(41,46),(41,47),(41,48),(42,43),
(42,44),(42,45),(42,46),(42,47),(42,48),(43,43),(43,44),(43,45),(43,46),(43,
47),(43,48),(44,43),(44,44),(44,45),(44,46),(44,47),(44,48),(45,43),(45,44),
(45,45),(45,46),(45,47),(45,48),(46,43),(46,44),(46,45),(46,46),(46,47),(46,
48),(47,43),(47,44),(47,45),(47,46),(47,47),(47,48),(48,43),(48,44),(48,45),
(48,46),(48,47),(48,48)

    def move_knife(self, action):
        if action == 4 or (action, self.previous_action) in forbidden_moves:
            action = self.previous_action
        else:
            self.previous_action = action
        head = self.knife[0]
        if action == 0:
            p = (head[0] - 1, head[1])
        elif action == 1:
            p = (head[0] + 1, head[1])
        elif action == 2:
            p = (head[0], head[1] - 1)
        elif action == 3:
            p = (head[0], head[1] + 1)
        self.knife.insert(0, p)
        self.knife.pop()

    def get_state(self):
        canvas = np.ones((self.grid_size, ) * 2)
        canvas[1:-1, 1:-1] = 0.0
        for seg in self.knife:
            canvas[seg[0], seg[1]] = 0.8
```

```
        for seg in self.art_border:
            canvas[seg[0],seg[1]] = 0.9
        for seg in self.mark_border:
            canvas[seg[0], seg[1]] = 0.5
        for seg in self.ut_border:
            canvas[seg[0],seg[1]] = 0.7
        for seg in self.per_border:
            canvas[seg[0],seg[1]] = 0.1
        return canvas

    def get_score(self):
        if self.game_over:
            score = -1
        elif self.plus_scored:
            score = +1
        elif self.minus_scored:
            score = -1
        elif self.non_scored:
            score = 0
        else:
            score = -1
        return score

    def reset(self):
        grid_size = self.grid_size
        knife_length = self.knife_length
        self.knife = [(10,24)]
        self.game_over = False
        self.plus_scored = False
        self.minus_scored = False
        self.non_scored = False
        self.arrange()
        if np.random.randint(2) == 0:
            self.previous_action = 0
        else:
            self.previous_action = 1
        self.border = []
        for z in range(grid_size):
          self.border += [(z, 0), (z, grid_size - 1), (0, z), (grid_size - 1, z)]
        self.border += self.preborder
        self.ut_border = []
        self.ut_border += self.uterus
        self.art_border = []
        self.art_border += self.artery
        self.mark_border = []
        self.mark_border += self.marker
        self.per_border = []
        self.per_border += self.peritoneum
```

```
def left(self):
    self.play(0)
def right(self):
    self.play(1)
def up(self):
    self.play(2)
def down(self):
    self.play(3)
def idle(self):
    self.play(4)

def hit_border(self):
   return self.knife[0] in self.border or self.knife[0] in self.ut_border
def hit_artery(self):
    return self.knife[0] in self.art_border
def hit_marker(self):
    return self.knife[0] in self.mark_border
def hit_peritoneum(self):
    return self.knife[0] in self.per_border

def is_over(self):
    return self.hit_border()
def is_won(self):
    return len(self.mark_border) < 1
```

## Author details

Masakazu Sato*, Kaori Koga, Tomoyuki Fujii and Yutaka Osuga

*Address all correspondence to: masakasatou-tky@umin.ac.jp

Department of Obstetrics and Gynecology, Graduate School of Medicine, The University of Tokyo, Bunkyo-ku, Tokyo, Japan

## References

[1] Ravi D, Wong C, Deligianni F, Berthelot M, Andreu Perez J, Lo B, et al. Deep learning for health informatics. IEEE Journal of Biomedical and Health Informatics. Jan 2017;**21**(1): 4-21. DOI: 10.1109/JBHI.2016.2636665. PubMed PMID: 28055930

[2] Scholkopf B. Artificial intelligence: Learning to see and act. Nature. 2015;**518**(7540):486-487. DOI: 10.1038/518486a. Epub 2015/02/27, PubMed PMID: 25719660

[3] Zhang YC, Kagen AC. Machine learning Interface for medical image analysis. Journal of Digital Imaging. Oct 2017;**30**(5):615-621. DOI: 10.1007/s10278-016-9910-0. PubMed PMID: 27730415

[4] Gibney E. DeepMind algorithm beats people at classic video games. Nature. 2015;**518**(7540): 465-466. DOI: 10.1038/518465a. Epub 2015/02/27, PubMed PMID: 25719643

[5] Gibney E. Google AI algorithm masters ancient game of Go. Nature. 2016;**529**(7587): 445-446. DOI: 10.1038/529445a. Epub 2016/01/29, PubMed PMID: 26819021

[6] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015;**518**(7540):529-533. DOI: 10.1038/nature14236. PubMed PMID: 25719670

[7] Littman ML. Reinforcement learning improves behaviour from evaluative feedback. Nature. 2015;**521**(7553):445-451. DOI: 10.1038/nature14540. PubMed PMID: 26017443

[8] Wallace SK, Fazzari MJ, Chen H, Cliby WA, Chalas E. Outcomes and postoperative complications after hysterectomies performed for Benign compared with malignant indications. Obstetrics and Gynecology. 2016;**128**(3):467-475. DOI: 10.1097/AOG.0000000000001591. PubMed PMID: 27500339

[9] Rampasek L, Goldenberg A. TensorFlow: Biology's gateway to deep learning? Cell Systems. 2016;**2**(1):12-14. DOI: 10.1016/j.cels.2016.01.009. PubMed PMID: 27136685

[10] Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks. 2015;**61**:85-117. DOI: 10.1016/j.neunet.2014.09.003. PubMed PMID: 25462637

[11] Beyer-Berjot L, Berdah S, Hashimoto DA, Darzi A, Aggarwal R. A virtual reality training curriculum for laparoscopic colorectal surgery. Journal of Surgical Education. 2016;**73**(6):932-941. DOI: 10.1016/j.jsurg.2016.05.012. Epub 2016/06/28; PubMed PMID: 27342755

[12] Khan ZA, Kamal N, Hameed A, Mahmood A, Zainab R, Sadia B, et al. SmartSIM—A virtual reality simulator for laparoscopy training using a generic physics engine. The International Journal of Medical Robotics + Computer Assisted Surgery: MRCAS. 2016;**16**:437. DOI: 10.1002/rcs.1771. Epub 2016/09/28; PubMed PMID: 27671920

[13] Li XL, Du DF, Jiang H. The learning curves of robotic and three-dimensional laparoscopic surgery in cervical cancer. Journal of Cancer. 2016;**7**(15):2304-2308. DOI: 10.7150/jca.16653. PubMed PMID: 27994668; PubMed Central PMCID: PMCPMC5166541

[14] Romero-Loera S, Cárdenas-Lailson LE, de la Concha-Bermejillo F, Crisanto-Campos BA, Valenzuela-Salazar C, Moreno-Portillo M. Skills comparison using a 2D vs. 3D laparoscopic simulator. Cirugia y Cirujanos. 2016;**84**(1):37-44. DOI: 10.1016/j.circen.2015.12.012. (English Edition)

[15] Kusy M, Zajdel R. Application of reinforcement learning algorithms for the adaptive computation of the smoothing parameter for probabilistic neural network. IEEE Transactions on Neural Networks and Learning Systems. 2015;**26**(9):2163-2175. DOI: 10.1109/TNNLS.2014.2376703. PubMed PMID: 25532211

[16] Senda K, Hattori S, Hishinuma T, Kohda T. Acceleration of reinforcement learning by policy evaluation using nonstationary iterative method. IEEE Transactions on Cybernetics. 2014;**44**(12):2696-2705. DOI: 10.1109/TCYB.2014.2313655. PubMed PMID: 24733037

[17] Xu B, Yang C, Shi Z. Reinforcement learning output feedback NN control using deterministic learning technique. IEEE Transactions on Neural Networks and Learning Systems. 2014;**25**(3):635-641. DOI: 10.1109/TNNLS.2013.2292704. PubMed PMID: 24807456

IntechOpen

IntechOpen