Playing

발표 순서

발표 순서

개요

- **팀명**
- **딥러닝 목적**

데이터 저장과 전처리

- 데이터 읽어오기
- 훈련과 테스트 데이터 나누기
- 정규화 등의 전처리

딥러닝 모델

- 입력, 중간(은닉), 출력 층, 패러미터 수
- 모델 종류
- 전체 코드

시연

- 결과와 데이터 위치

소개

박윤석

자료 조사 및 PPT

박종근

자료 조사 및 PPT

김지환

자료 조사 및 발표

딥러닝 목적

2016년 세계의 이목을 두게 하는 알파고의 등장

알파고는 딥러닝을 이용한 자체 학습 및 데이터를 누적하여 만들어진 기기

간단하고 쉽게 접근 할 수 있는 보드게임인 오목을 딥러닝하여 AI의 학습데이터를 누적시켜 점점 난이도가 어려워지는 게임을 제작

분류를 사용하여 오목의 룰 처럼 연속적인 오목알이 5개가 모이면 게임이 끝나고 승리

게임이 끝나게 되면 훈련 데이터를 지정된 경로에 저장

2

데이터 저장과 전처리

- 데이터 읽어오기
- 데이터 수, 속성 수 등 공개
- 훈련과 테스트 데이터 나누기
- 정규화 등의 전처리

데이터 읽어오기

```
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
sess.run(tf.global_variables_initializer())

print("Learning started, It takes sometime.")
for epoch in range(training_epochs):
    avg_cost = 0

    batch_xs = np.zeros([batch_size,board_size*board_size],dtype='f')
    batch_ys = np.zeros([batch_size,board_size*board_size],dtype='f')


    for i in range(total_batch):

        for j in range(batch_size):

            batch_xs[j,:] = board_x_stack[j+i*batch_size]
            batch_ys[j,:] = board_y_stack[j+i*batch_size]


        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c/total_batch

    print('Epoch;', '%04d' % (epoch +1), 'cost =', '{:.9f}'.format(avg_cost))

print('학습이 완료 되었습니다.')
```

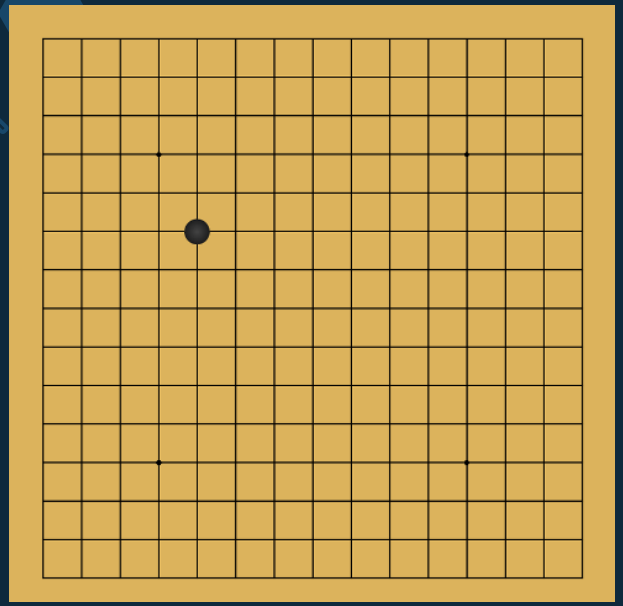
```
Learning started, It takes sometime.
Epoch; 0001 cost = 4.756234674
Epoch; 0002 cost = 3.939140008
Epoch; 0003 cost = 3.033379101
Epoch; 0004 cost = 2.369022113
Epoch; 0005 cost = 1.913035587
Epoch; 0006 cost = 1.545134771
Epoch; 0007 cost = 1.266280774
Epoch; 0008 cost = 1.079903150
Epoch; 0009 cost = 0.914493147
```

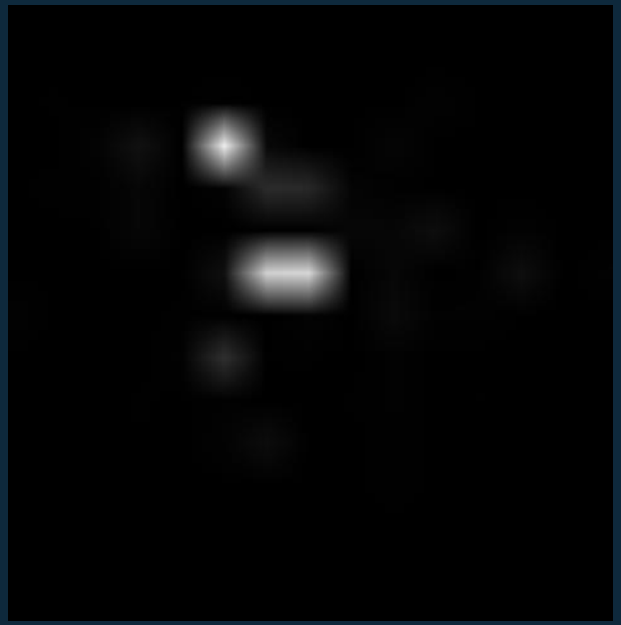
훈련과 테스트

기본 승리 데이터가 없으므로 자신이 혼자 오목을 하며 데이터를 쌓은 후, 어느 정도의 데이터가 학습이 되면 나 혼자가 아닌 AI와 오목을 하며 학습데이터를 쌓는다.

AI와 오목을 하고 게임이 끝나게 되면 오른쪽과 같이 이미지 파일 및 좌표를 txt파일에 행렬로 남게 된다.

정규화



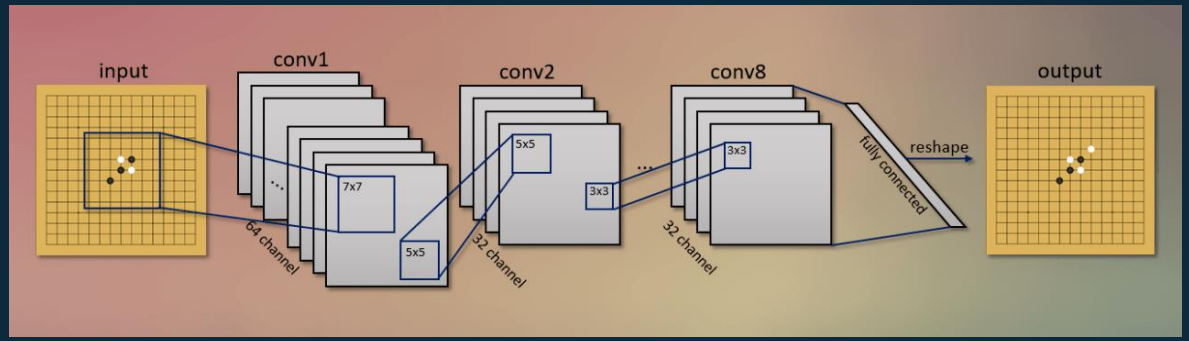


3

딥러닝 모델

- 입력, 중간(은닉), 출력 층, 패러미터 수
- 모델 종류
- 전체 코드

각 층 및 패러미터 수



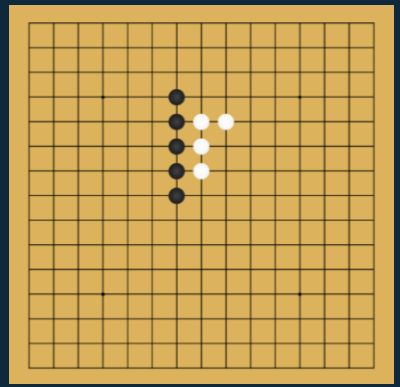

```
learning_rate = 0.001
training_epochs = 100
batch_size = 100
board_size = 15
file_path = 'training_data/txt/'
save_file = 'model/model.ckpt'
```

-모델 종류

cnn

모델이 직접 이미지, 비디오, 텍스트 또는 사운드를 분류하는 머신 러닝의 한 유형인 딥러닝에 가장 많이 사용되는 알고리즘이며 이미지에서 객체 얼굴, 장면을 인식하기 위해 패턴을 찾는 데 특히 유용하다.

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 2. 4. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 5. 6. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 7. 8. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0.]
```



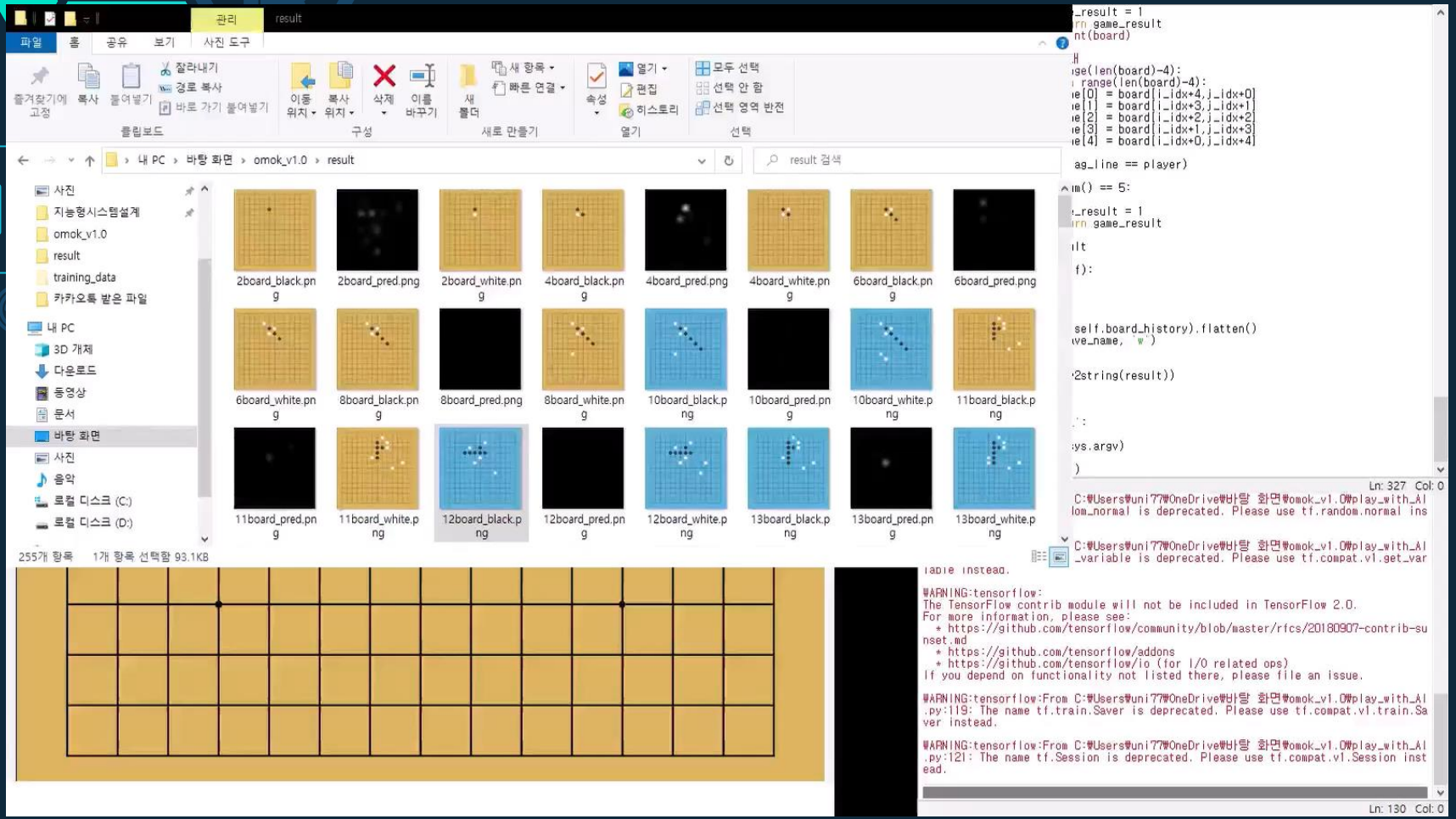
```python
import tensorflow as tf
import random
import os
import numpy as np


def readfile(file_name_path):

    board_vec = np.zeros([board_size*board_size],dtype='f')

    f = open(file_name_path, 'r')
    cnt = 0

    # 오목 학습데이터 (기보) 읽기
    while True:
        line = f.readline()
        if not line: break
        split_ini = line.split('.')

        for i in range(len(split_ini)):

            split_fur = split_ini[i].split(' ')

            if board_size*board_size <= cnt:
                break;

            elif split_fur[-1] != '\n':
                board_vec[cnt] = split_fur[-1]
                cnt = cnt + 1
    f.close()
    return board_vec

def data_aug(board_data): # 1개의 데이터를 회전, 반전으로 12개로 증강
    board_stack = []
    board_mat = board_data.reshape([board_size,board_size])
    # rotate
    board_stack.append(board_mat)
    board_stack.append(np.rot90(board_mat))
    board_stack.append(np.rot90(board_mat,2))
    board_stack.append(np.rot90(board_mat,3))
    board_fliplr = np.fliplr(board_mat)
    board_flipud = np.flipud(board_mat)
    # rotate and filp
    board_stack.append(board_fliplr)
    board_stack.append(np.rot90(board_fliplr))
    board_stack.append(np.rot90(board_fliplr,2))
    board_stack.append(np.rot90(board_fliplr,3))
    board_stack.append(board_flipud)
    board_stack.append(np.rot90(board_flipud))
    board_stack.append(np.rot90(board_flipud,2))
    board_stack.append(np.rot90(board_flipud,3))
    return board_stack
```

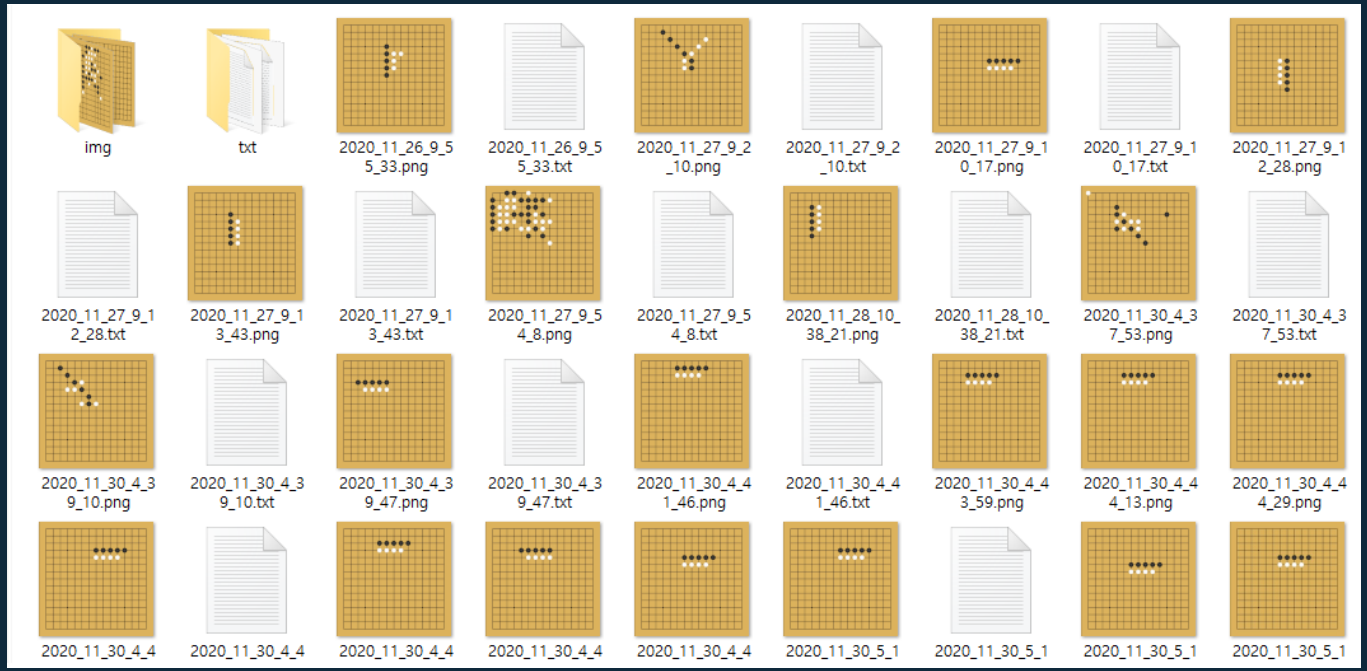
```python
### set parameters
learning_rate = 0.001
training_epochs = 100
batch_size = 100
board_size = 15
file_path = 'training_data/txt/' # 학습 기보 저장된 폴더
save_file = 'model/model.ckpt'    # 모델 저장 폴더 / 파일명

### 학습 데이터 저장 공간
board_x_stack = []
board_y_stack = []

keep_prob = tf.placeholder(tf.float32)


### set network
X = tf.placeholder(tf.float32, [None, board_size*board_size])
X_img = tf.reshape(X, [-1, board_size, board_size, 1])
Y = tf.placeholder(tf.float32, [None, board_size*board_size])
W1 = tf.Variable(tf.random_normal([7,7,1,64], stddev=0.1))
L1 = tf.nn.conv2d(X_img, W1, strides=[1,1,1,1], padding='SAME')
L1 = tf.nn.relu(L1)
#L1 = tf.nn.dropout(L1, keep_prob=keep_prob)
W2 = tf.Variable(tf.random_normal([5,5,64,32], stddev = 0.1))
L2 = tf.nn.conv2d(L1, W2, strides=[1,1,1,1], padding='SAME')
L2 = tf.nn.relu(L2)
#L2 = tf.nn.dropout(L2, keep_prob=keep_prob)
W3 = tf.Variable(tf.random_normal([3,3,32,32], stddev = 0.1))
L3 = tf.nn.conv2d(L2, W3, strides=[1,1,1,1], padding='SAME')
L3 = tf.nn.relu(L3)
#L3 = tf.nn.dropout(L3, keep_prob=keep_prob)
W4 = tf.Variable(tf.random_normal([3,3,32,32], stddev = 0.1))
L4 = tf.nn.conv2d(L3, W4, strides=[1,1,1,1], padding='SAME')
L4 = tf.nn.relu(L4)
#L4 = tf.nn.dropout(L4, keep_prob=keep_prob)
W5 = tf.Variable(tf.random_normal([3,3,32,32], stddev = 0.1))
L5 = tf.nn.conv2d(L4, W5, strides=[1,1,1,1], padding='SAME')
L5 = tf.nn.relu(L5)
#L5 = tf.nn.dropout(L5, keep_prob=keep_prob)
W6 = tf.Variable(tf.random_normal([3,3,32,32], stddev = 0.1))
L6 = tf.nn.conv2d(L5, W6, strides=[1,1,1,1], padding='SAME')
L6 = tf.nn.relu(L6)
W7 = tf.Variable(tf.random_normal([3,3,32,32], stddev = 0.1))
L7 = tf.nn.conv2d(L6, W7, strides=[1,1,1,1], padding='SAME')
L7 = tf.nn.relu(L7)
W8 = tf.Variable(tf.random_normal([3,3,32,32], stddev = 0.1))
L8 = tf.nn.conv2d(L7, W8, strides=[1,1,1,1], padding='SAME')
L8 = tf.nn.relu(L8)
W9 = tf.Variable(tf.random_normal([3,3,32,32], stddev = 0.1))
L9 = tf.nn.conv2d(L8, W9, strides=[1,1,1,1], padding='SAME')
L9 = tf.nn.relu(L9)
#L6 = tf.nn.dropout(L6, keep_prob=keep_prob)
L9 = tf.reshape(L9, [-1, board_size * board_size * 32])
W10 = tf.get_variable("W10", shape=[board_size * board_size * 32, board_size*board_size],initializer
b10 = tf.Variable(tf.random_normal([board_size * board_size]))
logits = tf.matmul(L9,W10) + b10
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits = logits, labels = Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# data load / augmentation
files = os.listdir(file_path)

for file_name in files:
    file_name_path = os.path.join(file_path,file_name)
    # read data
    board_data = readfile(file_name_path)
    # data augmentation
    board_stack = data_aug(board_data)
    for board_i in range(len(board_stack)):
        board_vec = board_stack[board_i].flatten()
        max_step = np.max(board_vec)
        for data_cnt in range(1,int(round(max_step*0.8))):
            board_x = np.zeros([board_size*board_size],dtype='f')
            board_y = np.zeros([board_size*board_size],dtype='f')
            for i in range(len(board_vec)):

                ############ 흑/백 학습 데이터 생성
                # 학습시에는 어느턴이든 항상 자기 돌은 1 상대돌은 0.5 으로 프로하게 한다
                if data_cnt%2 !=0:
                    if board_vec[i] > 0 and board_vec[i] <= data_cnt: # 해당 skip
                        if board_vec[i]%2 != 0:
                            # 흑
                            board_x[i] = 1
                        else:
                            board_x[i] = 2

                    if board_vec[i] == data_cnt + 1:
                        board_y[i] = 1

                else:
                    if board_vec[i] > 0 and board_vec[i] <= data_cnt: # 해당 skip
                        if board_vec[i]%2 == 0:
                            # 백
                            board_x[i] = 1
                        else:
                            board_x[i] = 2

                    if board_vec[i] == data_cnt + 1:
                        board_y[i] = 1

            if (np.sum(board_y)) != 1:
                print('학습데이터에 문제가 있습니다.')

            board_x_stack.append(board_x/2)
            board_y_stack.append(board_y)
```

```python
total_batch = (round(len(board_x_stack)/100)) - 1
print('학습데이터 생성이 완료 되었습니다: ', total_batch)

######### 학습 시작
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
sess.run(tf.global_variables_initializer())

print("Learning started, It takes sometime.")
for epoch in range(training_epochs):
    avg_cost = 0

    batch_xs = np.zeros([batch_size,board_size*board_size],dtype='f')
    batch_ys = np.zeros([batch_size,board_size*board_size],dtype='f')


    for i in range(total_batch):

        for j in range(batch_size):

            batch_xs[j,:] = board_x_stack[j+i*batch_size]
            batch_ys[j,:] = board_y_stack[j+i*batch_size]


        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c/total_batch

    print('Epoch:', '%04d' % (epoch +1), 'cost =', '{:.9f}'.format(avg_cost))

print('학습이 완료 되었습니다.')


######### 학습 모델 저장
saver = tf.train.Saver()
saver.save(sess, save_file)


print('모델저장이 완료 되었습니다.')
```

4 시연

- 결과와 데이터 위치


관리
result
파일
홈
공유
보기
사진 도구
내 PC › 바탕 화면 › omok_v1.0 › result
result 검색
2board_black.png
2board_pred.png
2board_white.png
4board_black.png
4board_pred.png
4board_white.png
6board_black.png
6board_pred.png
6board_white.png
8board_black.png
8board_pred.png
8board_white.png
10board_black.png
10board_pred.png
10board_white.png
11board_black.png
11board_pred.png
11board_white.png
12board_black.png
12board_pred.png
12board_white.png
13board_black.png
13board_pred.png
13board_white.png
255개 항목
1개 항목 선택함 93.1KB
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
Ln: 327 Col: 0
Ln: 130 Col: 0


img
txt
2020_11_26_9_55_33.png
2020_11_26_9_55_33.txt
2020_11_27_9_2_10.png
2020_11_27_9_2_10.txt
2020_11_27_9_10_17.png
2020_11_27_9_10_17.txt
2020_11_27_9_12_28.png
2020_11_27_9_12_28.txt
2020_11_27_9_13_43.png
2020_11_27_9_13_43.txt
2020_11_27_9_54_8.png
2020_11_27_9_54_8.txt
2020_11_28_10_38_21.png
2020_11_28_10_38_21.txt
2020_11_30_4_37_53.png
2020_11_30_4_37_53.txt
2020_11_30_4_39_10.png
2020_11_30_4_39_10.txt
2020_11_30_4_39_47.png
2020_11_30_4_39_47.txt
2020_11_30_4_41_46.png
2020_11_30_4_41_46.txt
2020_11_30_4_43_59.png
2020_11_30_4_44_13.png
2020_11_30_4_44_29.png
2020_11_30_4_4
2020_11_30_4_4
2020_11_30_4_4
2020_11_30_4_4
2020_11_30_4_4
2020_11_30_5_1
2020_11_30_5_1
2020_11_30_5_1
2020_11_30_5_1

Thanks!