

portfolio



All-Round Application Developer

☞ Android(Kotlin) 개발자가 되기 위해 노력하고 있습니다.

✚ Android Mid-Level / Senior Developer 라는 단기 목표를 이루고 난 후,
iOS Application(Swift, Objective C) & React Native 등 끊임없는 자기개발을 통해
Application의 모든 분야를 아우르는 All Round App-Developer를 꿈꾸고 있습니다.

About

Contact

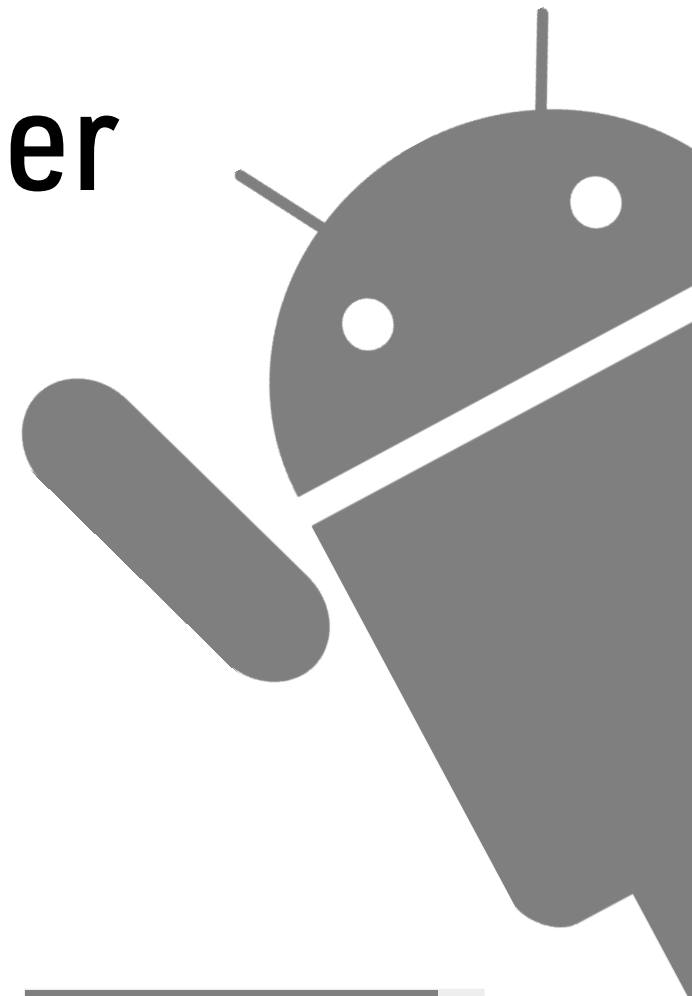
☎ 010-2832-8390
✉ jongillou@naver.com

Graduate

2014 효명고등학교 졸업
2014 인하대학교 정보통신공학과 입학
2021 인하대학교 정보통신공학과 졸업

Skill

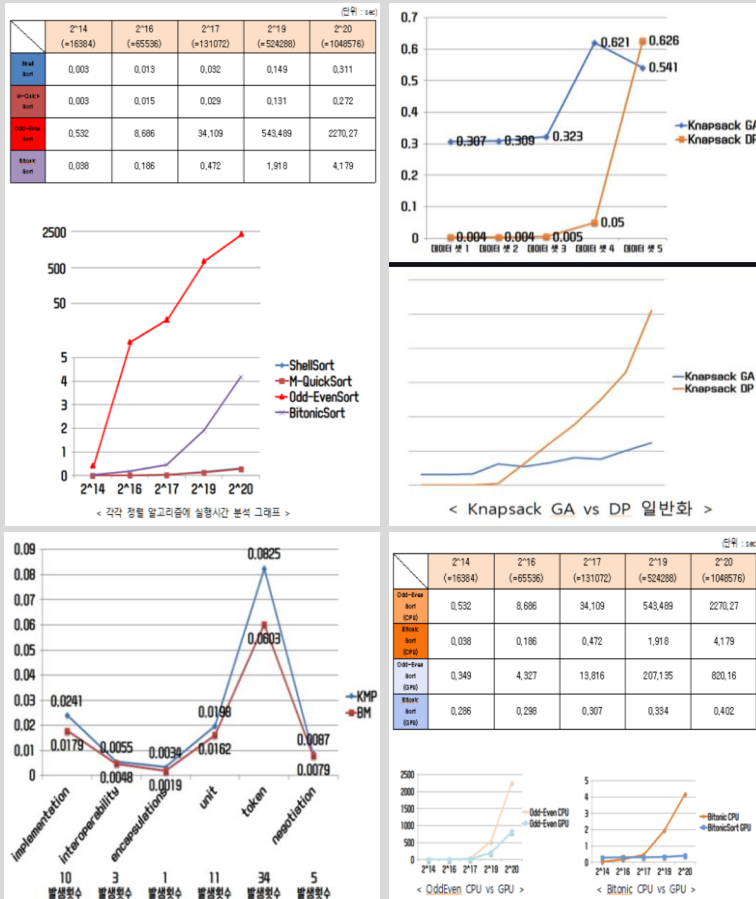
C++	<div><div></div></div>	90
Java, Kotlin	<div><div></div></div>	80
Python, C#	<div><div></div></div>	60
Machine Learning	<div><div></div></div>	75



ABOUT PROJECTS



PROJECT #1. ALGORITHM PROJECTS



사용한 기술스택 : C++, Visual Studio, CUDA (GPU)

프로젝트 인원 : 1명

프로젝트 기간 : 2개월

Visual Studio, C++을 이용하여 알고리즘을 구현하고 그 결과에 대해 분석하는 프로젝트를 진행하였습니다.

<상세 내용>

P1 : 정렬 알고리즘을 구현하고, 각 정렬 알고리즘 간의 성능을 분석

(Link) https://github.com/DCherish/Proj_Algorithm

P2 : String 탐색 알고리즘을 구현하고, 각 탐색 알고리즘 간의 성능을 분석

(Link) https://github.com/DCherish/Proj_Algorithm_2

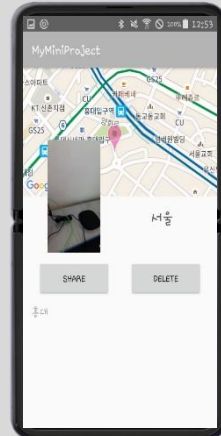
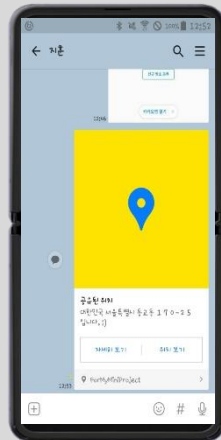
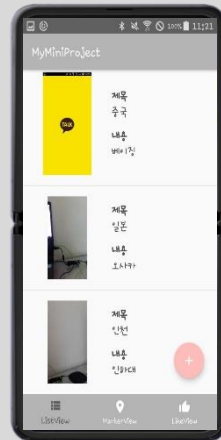
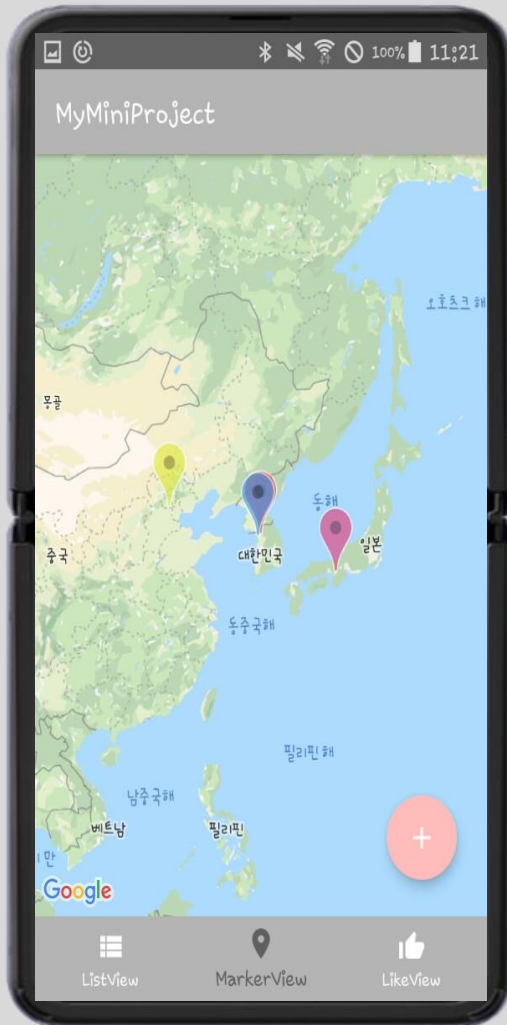
P3 : 그래프 알고리즘을 구현하고, 각 알고리즘을 분석

(Link) https://github.com/DCherish/Proj_Algorithm_3

P4 : 0-1 KnapSack 문제를 Dynamic Programming, Genetic Algorithm을 이용해 해결한 후 해당 결과를 분석

(Link) https://github.com/DCherish/Proj_Algorithm_Final

PROJECT #2-1. ANDROID PROJECTS



사용한 기술스택 : Java, Android Studio
프로젝트 인원 : 1명
프로젝트 기간 : 1개월

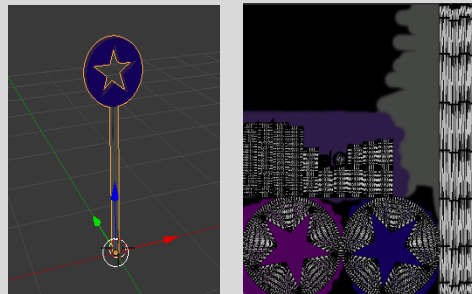
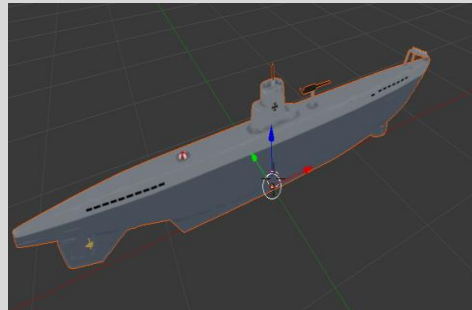
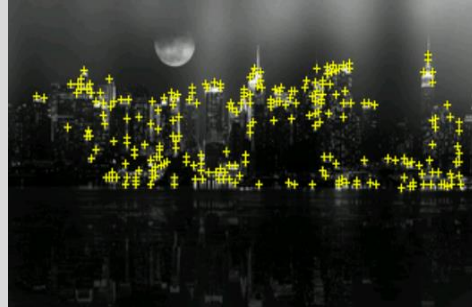
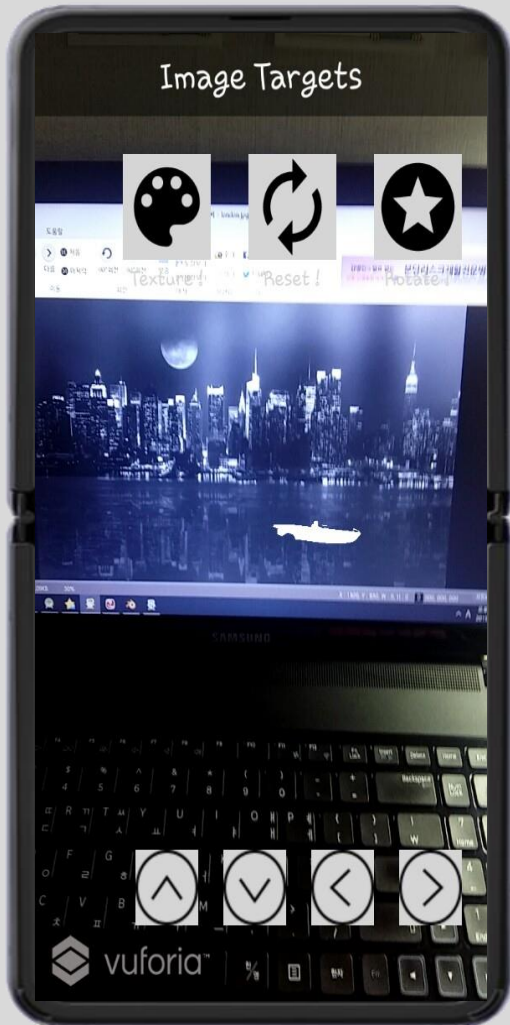
Android Studio, Java를 이용하여 여행 Application을 만들었습니다.

〈상세 내용〉

Bottom Navigation View에는 3개의 목록(ListView, Map, ListView)이 존재합니다. ListView의 각 항목을 누르거나 Map(GoogleMap)에 존재하는 Marker를 누르면 상세 페이지로 넘어갑니다. 상세 페이지에는 해당 항목 공유하기와 삭제하기 기능이 구현되어 있습니다. 공유하기 기능은 KakaoLink Open API를 이용하여 구현하였으며, Marker 위치의 위도/경도 값을 Geocoder를 이용하여 주소로 변경하여 공유하였습니다. 또한, SQLite와 SharedPreferences를 이용하여 항목들을 관리(저장, 삭제)할 수 있도록 구현하였습니다.

(Link) https://github.com/DCherish/Proj_Android

PROJECT #2-2. ANDROID PROJECTS



사용한 기술스택 : Java, Android Studio, Vuforia SDK
프로젝트 인원 : 1명
프로젝트 기간 : 1개월

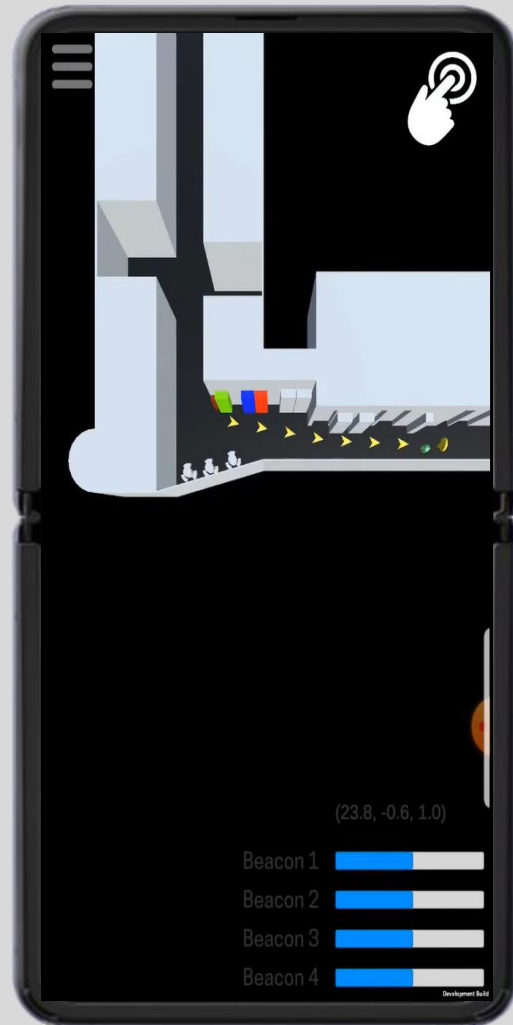
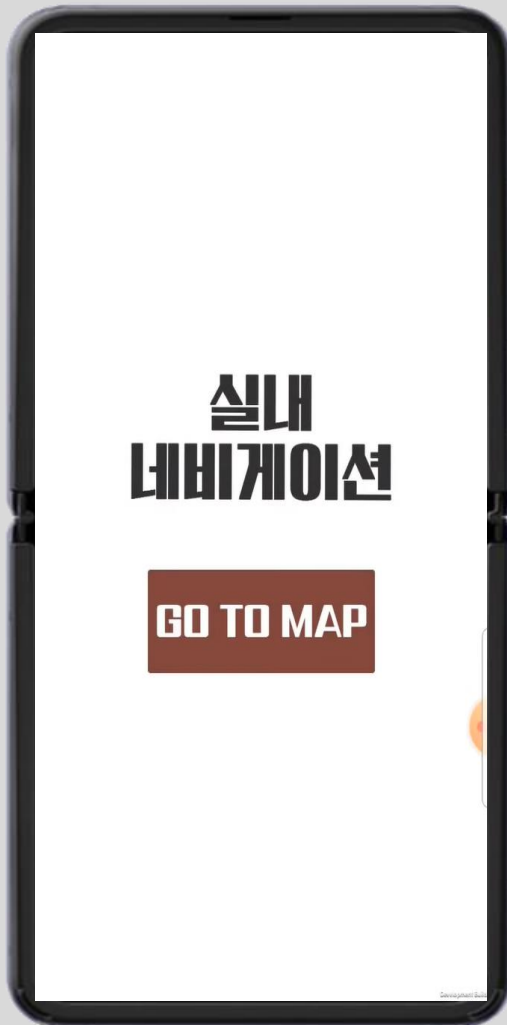
Android Studio, Java와 Vuforia SDK를 이용해 AR Application을 만들었습니다.

〈상세 내용〉

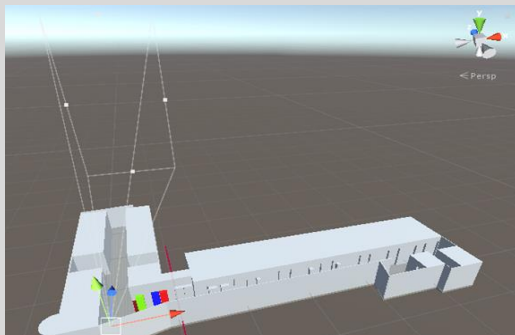
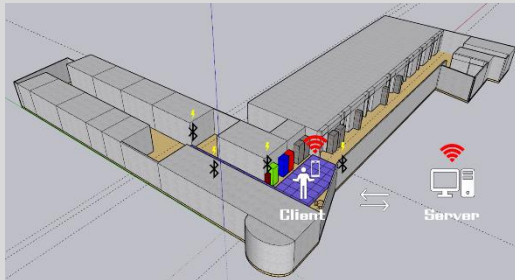
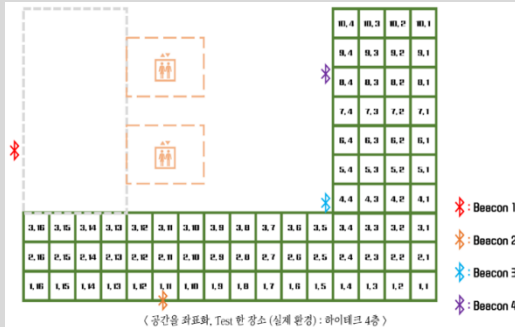
Camera에 Image Target(야경 사진)을 비추면 화면에 Cruise가 Augment 되어 마치 바다 위에 Cruise가 존재하는 것처럼 보이는데, Modeling Transform을 구현하여 상하좌우 방향으로 움직이거나 Cruise의 방향을 회전할 수 있도록 하였습니다. 또한, Coin 버튼을 누르면 바다 위의 Random한 위치에 Coin이 Augment 되어 Cruise가 움직이며 해당 Coin을 획득할 수 있는 Mini-Game까지 구현하였습니다.

(Link) https://github.com/DCherish/Proj_Android

PROJECT #3-1. AR NAVIGATION APPLICATION



PROJECT #3-2. AR NAVIGATION APPLICATION



BLE Beacon을 활용한 실내 AR Navigation

Argumented Reality Indoor Navigation using BLE Beacon

김지훈, 황성현
(Jihoon Kim and Sungghyun Hwang)

지도교수 : 유상조 (SangJo Yoo)

요약

본 논문에서는 블루투스 신호 강도를 이용한 실내 위치 측정 개선 및 증강현실을 이용한 내비게이션 서비스 제공 시스템을 구축하였다. 기존 측정 방식과 달리 기계학습을 통해 실내 측정을 할 수 있도록 했다. 또한 증강현실 기술을 접목해 사용자에게 보다 생동감 있는 모바일 앱을 제공함으로써 편리성을 제공할 수 있을 것으로 기대된다.

Abstract

In this paper, a system for improving indoor position measurement using Bluetooth signal strength and providing navigation service using augmented reality was established. Unlike the existing measurement method, we let them do indoor measurement through machine Learning. It is also expected that it will be able to provide convenience by providing more lively mobile apps to users by combining Argumented Reality technology.

Keywords: Machine Learning, Argumented Reality, Indoor Positioning, Application

1. 서론

최근 소비자들의 편리성을 증대시키기 위하여 실내 위치 측정에 관한 다양한 연구가 진행되고 있다. 단순히 무선 신호에 기반한 측위 시스템에서는



```
L17 = tf.nn.relu(tf.matmul(L16, W17))
L17 = tf.nn.dropout(L17, keep_prob)

L18 = tf.nn.relu(tf.matmul(L17, W18))
L18 = tf.nn.dropout(L18, keep_prob)

hypothesis = tf.matmul(L18, out)

h = tf.identity(hypothesis, name='h')

WARNING:tensorflow:From <python-input-448665201>:2: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use 'rate' instead of 'keep_prob'. Rate should be set to 'rate = 1 - keep_prob'.

cost = tf.reduce_mean(tf.square(hypothesis - y))

opt_inizier = tf.train.GradientDescentOptimizer(learning_rate=0.00001)
train_op = opt_inizier.minimize(cost)

init = tf.global_variables_initializer()
```

사용한 기술스택 : Python, C#, Java, TensorFlow, Jupyter Notebook, Unity 3D, Eclipse, Android Studio

프로젝트 인원 : 2명

프로젝트 기간 : 7개월

무선 신호를 이용하여 실내에서 길 찾기 서비스를 제공하는 Navigation Application을 제작하였습니다.

<상세 내용>

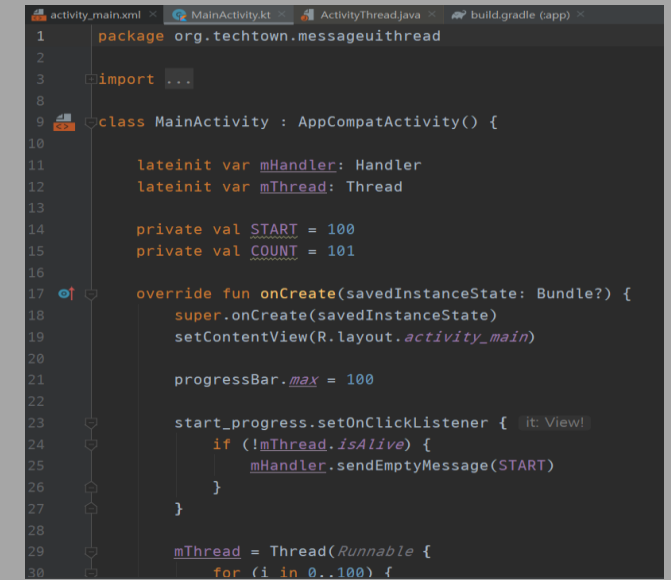
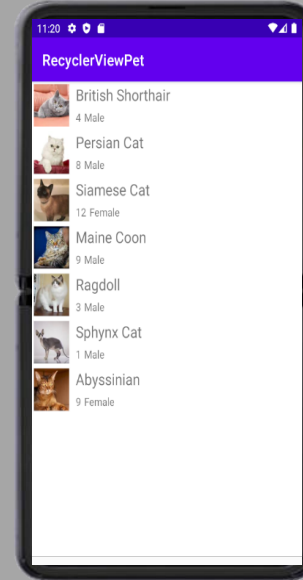
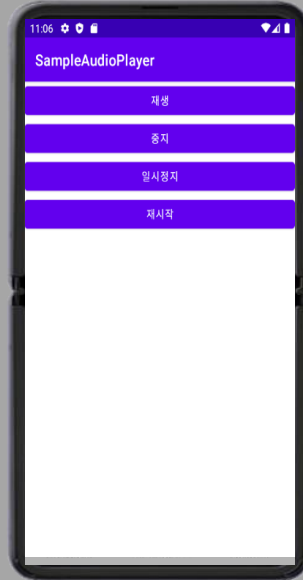
실내에서 정확하게 위치 측위를 하기 위해 Deep Learning의 DNN 모델을 이용하여 시스템을 설계하였습니다. 먼저, Unity 3D Tool을 이용하여 양질의 AR Application을 구현할 수 있었습니다. 또한, Eclipse Tool을 이용하여 Server와 Client 간의 Socket Programming을 구현하여 Bluetooth RSSI (신호 세기)를 송수신할 수 있었습니다. 또한, Android Plugin을 생성하여 이 파일을 통해 Unity 3D에서 Android 접근 권한을 설정 및 Bluetooth를 이용할 수 있도록 하였습니다.

(Link) <https://www.youtube.com/watch?v=NvaQYQSHGWI>

ABOUT STUDY



STUDY #1. ANDROID STUDY



Android에 대해 더욱 깊게 공부하고자 하여 다양한 자료(책, 인터넷, 동영상 등)를 통해 독학하였습니다. 이를 통해 강의에서 배우지 못했던 Android의 기본적인 원리와 다양한 내용에 대해 학습할 수 있었습니다. 나아가 Kotlin을 사용하는 Android 개발자가 되고 싶어 Kotlin을 공부하였습니다. Kotlin을 공부하며 배운 내용이었던 kotlin-android-extensions 및 Anko 라이브러리가 deprecated 되어 현재 새롭게 Jetpack과 관련된 내용을 공부하고 있습니다.

(Link) https://github.com/DCherish/Android_N_Kotlin

STUDY #2. MACHINE/DEEP LEARNING


Recurrent Neural Network

MULTIMEDIA NETWORK

- RNN computational graph
 - Re-use the same weight matrices at every time step

INHA University Prof. Sang-U Yoo siyoo@inha.ac.kr <http://multinet.inha.ac.kr> 55

MNIST DNN Case Study



```

cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=hypothesis, labels=y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int((mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(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('Learning Finished!')
```

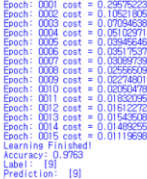
```

correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Accuracy: ', sess.run(accuracy, feed_dict={
    X: mnist.test.images, Y: mnist.test.labels}))

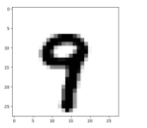
r = random.randint(0, mnist.test.num_examples - 1)
print('Label: ', sess.run(tf.argmax(mnist.test.labels[r:r+1], 1)))
print('Prediction: ', sess.run(
    tf.argmax(hypothesis, 1), feed_dict={X: mnist.test.images[r:r+1], Y:
mnist.test.labels[r:r+1]


plt.imshow(mnist.test.images[r:r+1].
          reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()

```



Epoch: 0001 cost = 0.295752
Epoch: 0002 cost = 0.19521059
Epoch: 0003 cost = 0.09456284
Epoch: 0004 cost = 0.051029714
Epoch: 0005 cost = 0.03945646
Epoch: 0006 cost = 0.03517578
Epoch: 0007 cost = 0.03069793
Epoch: 0008 cost = 0.02748106
Epoch: 0009 cost = 0.02248016
Epoch: 0010 cost = 0.02034766
Epoch: 0011 cost = 0.01845223
Epoch: 0012 cost = 0.01827253
Epoch: 0013 cost = 0.01845223
Epoch: 0014 cost = 0.01845223
Epoch: 0015 cost = 0.01196584
Learning Finished!
Accuracy: 0.9763





INHA University Prof. Sang-Jo Yoo sjyoo@inha.ac.kr

http://multinet.inha.ac.kr **62**

FileEditViewInsertCellKernelsHelp

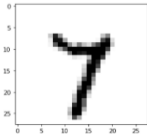
Python 3.0

RunCode

```
r = random.randint(0, mnist.test.num_examples - 1)
print('Label: ', sess.run(tf.argmax(mnist.test.labels[r : r + 1], 1)))
print(
    "Prediction: ",
    sess.run(tf.argmax(logits, 1), feed_dict={X: mnist.test.images[r : r + 1]})
)

plt.imshow(
    mnist.test.images[r : r + 1].reshape(28, 28),
    cmap='Greys',
    interpolation='nearest'
)
plt.show()

Learning started. It takes sometime.
Epoch: 0001 cost = 0.36309277
Epoch: 0002 cost = 0.09609546
Epoch: 0003 cost = 0.06975126
Epoch: 0004 cost = 0.07008167
Epoch: 0005 cost = 0.04725429
Epoch: 0006 cost = 0.04110843
Epoch: 0007 cost = 0.03678346
Epoch: 0008 cost = 0.03035547
Epoch: 0009 cost = 0.03232642
Epoch: 0010 cost = 0.02587789
Epoch: 0011 cost = 0.02253171
Epoch: 0012 cost = 0.03077974
Epoch: 0013 cost = 0.01760167
Epoch: 0014 cost = 0.01572606
Epoch: 0015 cost = 0.01401075
Learning finished
Accuracy: 0.9885
Label: [7]
Prediction: [7]
```



In []:

```
File Edit View Insert Cell Widgets Help
In [4]: env = gym.make('FrozenLake-v0')

In [5]: 0 = np.zeros([env.observation_space.n, env.action_space.n])

In [6]: learning_rate = .05
        dis = 0
        num_episodes = 2000

In [7]: rList = []
        for i in range(num_episodes):
            state = env.reset()
            rAI = 0
            done = False

            while not done:
                action = np.argmax(Q[state, :] + np.random.randn(1, env.action_space.n) / (i + 1))
                new_state, reward, done, _ = env.step(action)


                Q[state, action] = (1 - learning_rate) * Q[state, action] + learning_rate * (reward + dis * np.max(Q[new_state, :]))
                state = new_state

                rAI += reward

            rList.append(rAI)

        print("Score over time: " + str(sum(rList) / num_episodes))
        print("Final Q-Table Values")
        print(Q)
        plt.bar(range(len(rList)), rList, color='blue')
        plt.show()

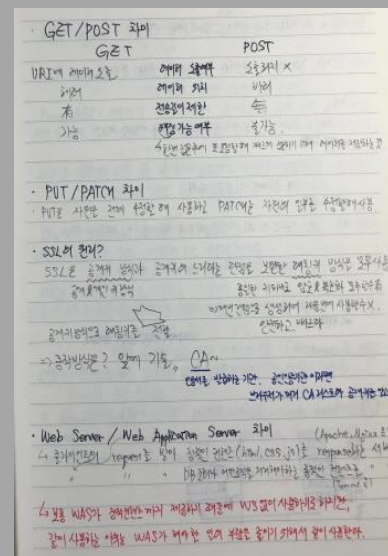
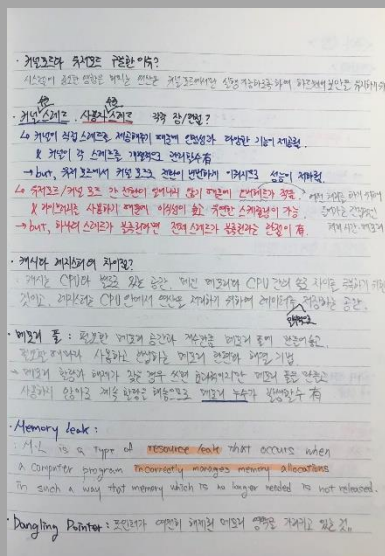
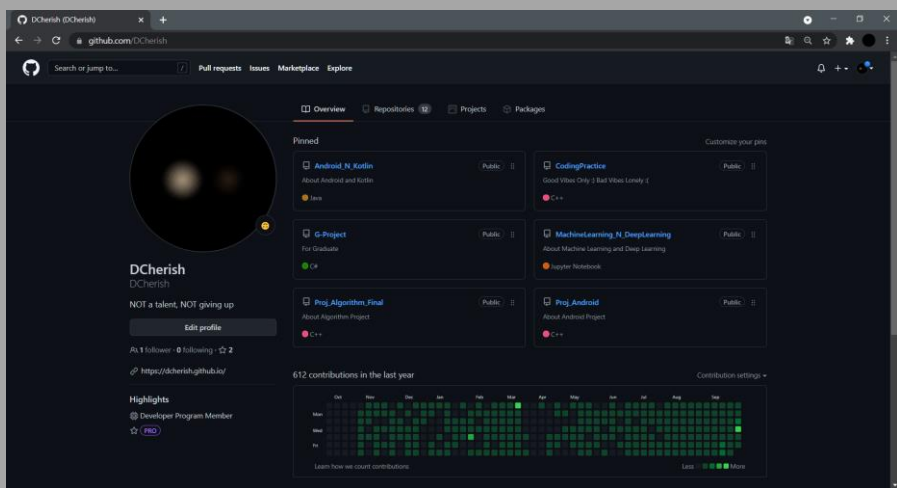
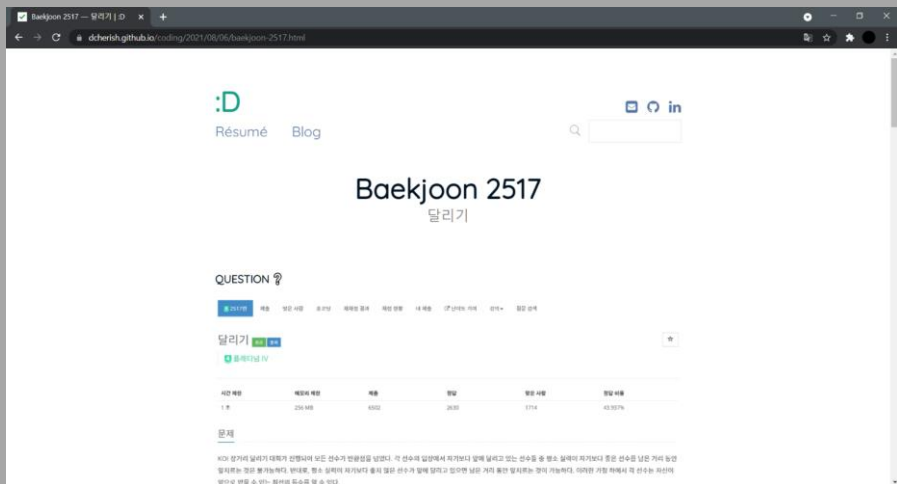
Score over time: 0.349
Final Q-Table Values
[[ 0.7581175e-05  5.35429563e-03  6.07323300e-01  1.00000211e-02]
 [ 2.8897177e-04  7.00314599e-04  4.25664118e-05  2.45444835e-05]
 [ 8.0076620e-04  1.23606870e-05  1.20644040e-05  1.70119843e-01]
 [ 9.80386179e-06  5.09284929e-04  1.97047930e-04  1.40897851e-01]
 [ 7.98124870e-01  9.13885770e-04  2.75729797e-04  4.62869284e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 1.37446000e-01  1.47411920e-02  2.01048100e-05  5.63253000e-05]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 2.12100500e-01  1.80777800e-03  3.49825500e-05  7.68476500e-05]
 [ 0.00000000e+00  8.21931300e-01  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  3.48267540e-01  0.00000000e+00  4.65376700e-05]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 3.08150270e-04  0.00000000e+00  9.79360190e-01  3.16197754e-04]
 [ 0.00000000e+00  0.00000000e+00  9.65313090e-01  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```



프로젝트를 마친 후 Machine Learning과 Deep Learning 분야에 대해 조금 더 깊게 공부하고자 지도 교수님에게 자문하였고, 교수님께서 세미나 목적으로 제작하셨던 강좌를 저에게 제공해주셨습니다. 프로젝트를 위해 이 분야에 대하여 독학할 당시, 이해가 되지 않았던 부분이 많이 존재했는데 이 강좌와 자료를 통해 많은 도움을 얻을 수 있었고 Machine/Deep Learning 분야에 대해 좀 더 넓고 깊게 개념들을 정리할 수 있었습니다.

(Link)  https://github.com/DCherish/MachineLearning_N_DeepLearning

STUDY #3. CODING/PROGRAMMING STUDY



(출처) <https://brunch.co.kr/@oemilk/113>

백준, 프로그래머스 사이트를 통해 다양한 문제를 풀며 알고리즘 지식과 문제 해결 능력을 쌓고 있습니다. 가끔 저 자신의 부족함을 느낄 때도 있지만 꾸준히 문제를 풀고 블로그에 정리하며 성장하기 위해 묵묵히 노력하고 있습니다.

(Link) <https://github.com/DCherish/CodingPractice>

또한, 컴퓨터 관련 CS 지식에 대해 정리한 노트필기 및 관련 자료를 블로그에 업로드할 예정이며, 업로드 이후부터는 지속적으로 최신화를 할 예정입니다.

(Link) <https://dcherish.github.io/blog>

thankyou

