



주식 알파고

(Stock Alphago)

*Deep-Reinforcement-Learning*을 통한 주식 모의투자



이승환



이주윤



최지웅



백건우



박세연

S tock Alphago

목 차





목 차



01 프로젝트 계획



03 데이터 수집



05 강화학습 적용



07 웹 구현



02 GCP 환경 구축



04 딥러닝 모델 생성



06 대신증권 API 연결



08 결 론

01

프로젝트 계획



주제 선정



프로젝트 목표



대신증권 API

강화학습

머신러닝 & 딥러닝



1억원 모의투자 € 2% 수익률 달성

프로젝트 일정표



8월 말

9월 초

9월 중

9월 말

10월 초

10월 중

10월 말

11월 초

프로젝트 계획 수립

GCP 환경 구축

대신증권 일/분단위 데이터 수집

딥러닝 코드 작성 & 학습 개선
Transformer 적용

강화학습 코드 작성 및 개선

대신증권 API 연결
실시간 모의 투자

포트폴리오 작성
Web 구현

프로젝트 사용 Tool



운영 체제



Windows 10



CentOS

개발 툴



colab

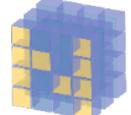


PuTTY

라이브러리



python™



NumPy



pandas

matplotlib



TensorFlow



Keras



MariaDB



HS



seaborn

참고문헌 (References)



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.



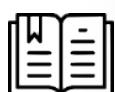
Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*. 2014.



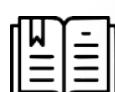
Xiong, Zhuoran, et al. "Practical deep reinforcement learning approach for stock trading." *arXiv preprint arXiv:1811.07522* (2018)



파이썬과 케라스를 이용한 딥러닝 / 강화학습 주식투자 – 위키북스 (퀀티랩, 2020)



Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International conference on machine learning*. PMLR, 2015.



Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).



Azhikodan, Akhil Raj, Anvitha GK Bhat, and Mamatha V. Jadhav. "Stock trading bot using deep reinforcement learning." *Innovations in Computer Science and Engineering*. Springer, Singapore, 2019. 41-49.

02

GCP 환경 구축





Google Cloud Platform



Google Cloud Platform

대용량 주식데이터 수집

저장 공간 필요



GCP 활용 : VM인스턴스



SSH 연결



수집 데이터 저장 : MariaDB



Jupyter 개발 환경 생성

VM 인스턴스 생성



VM 인스턴스 [인스턴스 만들기](#) [VM 가져오기](#) [새로고침](#) [시작/재개](#) [중지](#) [정지](#) [재설정](#) [삭제](#) [일정 만들기](#)

💡 'salphago' 인스턴스의 사용량이 높습니다. e2-custom(vCPU 2개, 8.5GB 메모리) 머신 유형으로 전환하면 월 \$20 정도 절감할 수 있습니다. [자세히 알아보기](#) [닫기](#)

[인스턴스](#) [인스턴스 일정](#)

VM 인스턴스는 Google 인프라에서 워크로드를 실행하기 위한 구성하기 쉬운 가상 머신입니다. [자세히 알아보기](#)

[필터](#) 속성 이름 또는 값 입력

<input checked="" type="checkbox"/> 상태	이름 ↑	영역	권장사항	다음에서 사용 중:	내부 IP	외부 IP	연결
<input checked="" type="checkbox"/>	salphago	asia-northeast3-a	💡 매월 \$20 절감 💡 매월 \$83 절감		10.178.0.3 (nic0)	34.64.200.74	SSH ⋮

VM 인스턴스

seyeon200@salphago:~ - Chrome

ssh.cloud.google.com/projects/salphago/zones/asia-northeast3-a/instances/salphago?authuser=0&hl=ko&projectN...

```
Connected, host fingerprint: ssh-rsa 0 90:58:48:62:AC:A3:5A:EF:53:50
:F6:BA:13:F4:AB:C0:A6:84:E1:CA:64:33:DE:01:39:4D:FC
Last login: Mon Nov  8 02:18:31 2021 from 35.235.242.81
(base) [seyeon200@salphago ~]$ jupyter notebook
```

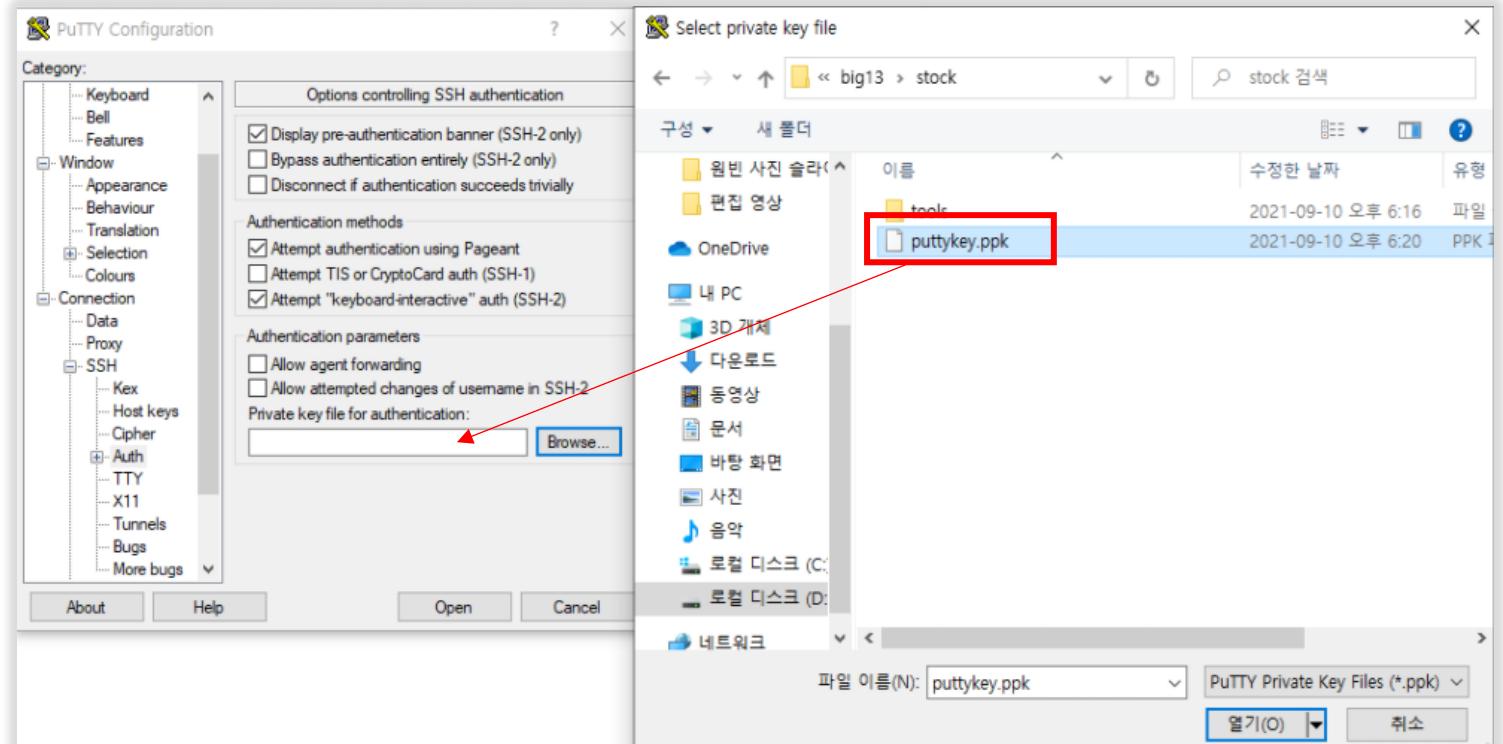
Putty 설치 & SSH 키 생성



Compute Engine
SSH 키

프로젝트의 VM 인스턴스에 연결하는 데 사용될 수 있는 SSH 키를 저장할 수 있습니다. 프로젝트 수준 키는 자체 SSH 키가 없는 모든 VM 인스턴스에 반영됩니다. 자세히 알아보기

SSH 키 추가



34.64.201.33 - PuTTY

```
Using username "nubes8711".
Authenticating with public key "nubes8711@gmail.com"
Passphrase for key "nubes8711@gmail.com":
```

MariaDB 설치



```
(base) [nubes8711@stockai ~]$ sudo vi /etc/yum.repos.d/MariaDB.repo
[nubes8711@stockai:~ - Chrome
 ssh.cloud.google.com/projects/stockai-325609/zones/asia-northeast3-a/instances/stockai?authuser=0&hl=k...
[mariadb]
name=MariaDB
baseurl=http://yum.mariadb.org/10.1/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
~
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 10.1.48-MariaDB MariaDB Server
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
+-----+
3 rows in set (0.00 sec)
```



03

데이터 수집



KOSPI200 Crawling - 한국증권거래소



- Selenium을 이용한 주식 데이터 크롤링

```

import numpy as np
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from urllib.request import urlopen
import time

day = pd.date_range(
    '20210817',
    periods=20,
    freq='B'
)

driver=webdriver.Chrome('D:/big13/python/data-anal/driver/chromedriver.exe')
driver.get('http://data.krx.co.kr/contents/MDC/MDI/mDILoader/index.cmd?menuId=MDC0201')

time.sleep(1)

a='''//*[@id="jsMdiMenu"]/div[4]/ul/li[1]/ul/li[2]/div/div[1]/ul/li[2]/a'''
b='''//*[@id="jsMdiMenu"]/div[4]/ul/li[1]/ul/li[2]/div/div[1]/ul/li[2]/ul/li[1]/a'''
c='''//*[@id="jsMdiMenu"]/div[4]/ul/li[1]/ul/li[2]/div/div[1]/ul/li[2]/ul/li[1]/ul/li[1]/a'''

driver.find_element_by_xpath(a).click()
time.sleep(1)
driver.find_element_by_xpath(b).click()
time.sleep(1)
driver.find_element_by_xpath(c).click()
time.sleep(15)

kpath = driver.find_element_by_id("mktId_0_0")
kpath.send_keys(Keys.ARROW_RIGHT)

time.sleep(1)

```

```

for today in day:

    days = today.strftime('%Y%m%d')
    date_input=driver.find_element_by_name('trdDd') # //*[@id="trdDd"]
    date_input.send_keys(Keys.CONTROL+'a'+Keys.BACK_SPACE)

    #날짜 설정
    date_input.send_keys(days)
    time.sleep(1)

    #조회버튼
    ipath = """//*[@id="jsSearchButton"]"""
    driver.find_element_by_xpath(ipath).click()
    time.sleep(2)
    #의뢰수정
    ypath = """//*[@id="MDCSTAT015_FORM"]/div[2]/div/p[2]/button[2]/img"""
    driver.find_element_by_xpath(ypath).click()
    time.sleep(1)

    zpath = "div > div:nth-child(2) > a"
    driver.find_element_by_css_selector(zpath).click()
    time.sleep(5)

driver.close()

from glob import glob
alist = glob('C:/Users/TJ/downloads/data_*.csv')

tmp_raw=[]
j=0
for i in alist:
    aa = pd.read_csv(i,encoding='euc-kr')
    aal['날짜']=day[j].strftime('%Y%m%d')
    j=j+1
    aa.sort_values(by='시가총액',inplace=True, ascending=False)
    aa = aa[:200]
    aa = aa.reset_index()
    aa.drop('index',inplace=True, axis=1)
    tmp_raw.append(aa)

tmp_raw

```



KOSPI200 Crawling - 대신증권

```
# 코스피 불러오기
KOSPI = []

g_objCodeMgr = win32com.client.Dispatch('CpUtil.CpCodeMgr')
codelist = g_objCodeMgr.GetGroupCodeList(2)
for code in codelist :
    print(code, g_objCodeMgr.CodeToName(code))
    KOSPI.append(code)
print(len(KOSPI))
```

```
# 차트 객체 로드
objStockChart = win32com.client.Dispatch("CpSysDib.StockChart")
```

```
# 주식 데이터 로드
for stockcode in tqdm(KOSPI):
    objStockChart.SetInputValue(0, stockcode) # 종목 코드
    objStockChart.SetInputValue(1, ord('1')) # 요청구분 (1=기간단위 조회)
    objStockChart.SetInputValue(3, '20160101') # 요청 시작일
    objStockChart.SetInputValue(4, -1) # 1000000개 = 모든 구간
    objStockChart.SetInputValue(5, [0, 1, 2, 3, 4, 5, 8]) # 요청항목 - 날짜, 시간, 시가, 고가, 저가, 종가, 거래량
    objStockChart.SetInputValue(6, ord('m')) # 차트구분
    objStockChart.SetInputValue(9, ord('1')) # 수정주가
```

```
# 저장공간 초기화
data = {
    'date' : [], # 0:날짜
    'open' : [], # 2:시가
    'high' : [], # 3:고가
    'low' : [], # 4:저가
    'close' : [], # 5:종가
    'volume' : [], # 8:거래량
}
```

```
# 데이터 요청
while True:
    checkandWait(1)
    objStockChart.BlockRequest()
    cnt = objStockChart.GetHeaderValue(3) # 3 : 수신개수
    for i in range(cnt): # 수신된 데이터 개수만큼 for문 실행

        # 날짜, 시간 데이터 합치기
        data['date'].append(objStockChart.GetDataValue(0, i)*10000 + objStockChart.GetDataValue(1, i))

        # 데이터 저장
        for j in range(1,len(data.keys())):
            key=list(data.keys())[j]
            data[key].append(objStockChart.GetDataValue(j+1, i))

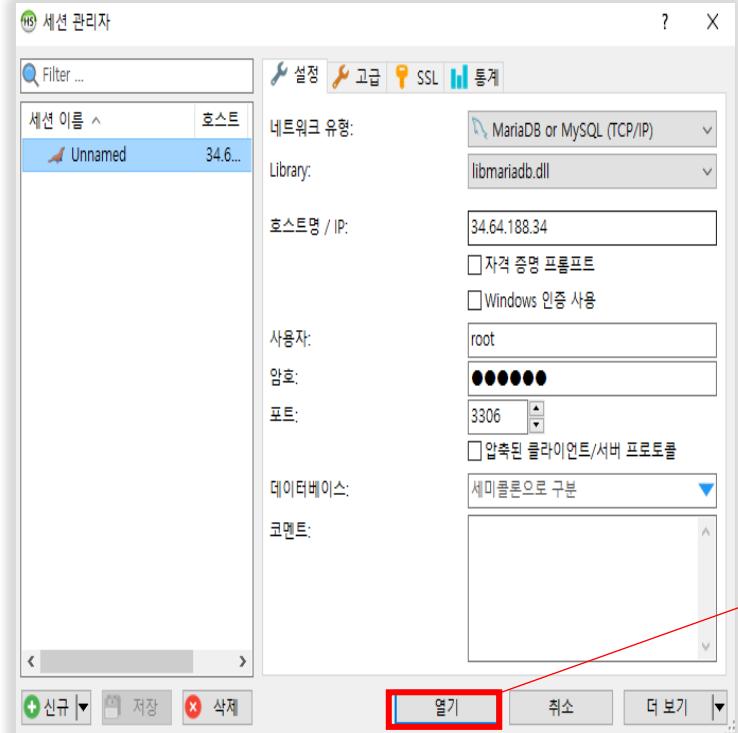
    # 더 요청할 데이터가 없으면 break, 있으면 계속 요청
    if objStockChart.Continue == False :
        break

# 딕셔너리 데이터프레임 변환
df = pd.DataFrame(data)
df['date'] = pd.to_datetime(df['date'], format='%Y%m%d%H%M')

df.sort_values(by='date', ascending=True, inplace=True) # 내림차순 정렬

# DataFrame을 DB에 바로 입력
df.to_sql(
    name=stockcode,
    con=engine,
    index=False,
    chunksize=1000,
    if_exists='replace'
)
```

KOSPI200 주식 데이터 수집 - 일단위



Untitled@mystock# - HeidiSQL 11.3.0.6295

파일 편집 검색 쿼리 도구 이동 도움말

데이터베이스 필터 테이블 필터 ★

호스트: 34.64.188.34 데이터베이스: mystock 쿼리

이름	행	크기	생성됨	업데이트됨	엔진	코멘트	유형
A000020	157,920	23.6 MIB	2021-10-04 09:1...		InnoDB		Table
A000040	141,330	21.5 MIB	2021-10-04 09:1...		InnoDB		Table
A000050	86,415	13.5 MIB	2021-10-04 09:1...		InnoDB		Table
A000060	185,640	28.6 MIB	2021-10-04 09:1...		InnoDB		Table
A000070	160,650	24.6 MIB	2021-10-04 09:2...		InnoDB		Table
A000075	15,120	2.5 MIB	2021-10-04 09:2...		InnoDB		Table
A000080	188,580	28.6 MIB	2021-10-04 09:2...		InnoDB		Table
A000087	43,010	6.5 MIB	2021-10-04 09:2...		InnoDB		Table
A000100	186,900	28.6 MIB	2021-10-04 09:2...		InnoDB		Table
A000105	57,540	9.5 MIB	2021-10-04 09:2...		InnoDB		Table
A000120	187,425	28.6 MIB	2021-10-04 09:2...		InnoDB		Table
A000140	137,959	20.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000145	22,260	3.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000150	179,970	27.6 MIB	2021-10-04 09:3...		InnoDB		Table
A000155	105,735	16.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000157	57,855	9.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000180	110,355	16.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000210	181,735	26.6 MIB	2021-10-04 09:3...		InnoDB		Table
A000215	90,705	13.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000220	141,960	21.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000225	79,485	12.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000227	12,390	2.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000230	67,653	10.5 MIB	2021-10-04 09:3...		InnoDB		Table
A000240	178,185	27.6 MIB	2021-10-04 09:3...		InnoDB		Table
A000270	188,580	28.6 MIB	2021-10-04 09:3...		InnoDB		Table
A000300	130,830	19.5 MIB	2021-10-04 09:4...		InnoDB		Table
A000320	88,620	13.5 MIB	2021-10-04 09:5...		InnoDB		Table
A000325	11,130	2.5 MIB	2021-10-04 09:5...		InnoDB		Table

KOSPI200 주식 데이터 수집 - 분단위



Session Manager

Filter ...

설정 고급 SSL 통계

세션 이름 ^ 호스트
Unnamed 34.6...

네트워크 유형: MariaDB or MySQL (TCP/IP)
Library: libmariadb.dll

호스트명 / IP: 34.64.188.34
 자격 증명 프롬프트
 Windows 인증 사용

사용자: root

암호: *****

포트: 3306
 압축된 클라이언트/서버 프로토콜

데이터베이스: 세미클론으로 구분

코멘트:

신규 저장 삭제 열기 취소 더 보기

Unnamed#stock100_min_all#A000100# - HeidiSQL 11.3.0.6295

파일 편집 검색 쿼리 도구 이동 도움말

데이터베이스 펌터 테이블 펌터 ★ 호스트: 34.64.188.34 데이터베이스: stock100_min_all 테이블: A000100 데이터 쿼리

stock100_min_all.A000100: 187,409 행 (총) (대략적), 제한 수: 1,000

date	open	high	low	close	vol	amount	money	sell	buy	stock	price	foreign	gigwan	vol2
2019-10-10 09:01:00	901	41,407	41,407	41,407	41,407	0	1,730	78,190,000	1	345	0	0	0	0
2019-10-10 09:02:00	902	41,407	41,407	41,407	41,407	0	505	22,830,000	102	345	0	0	0	0
2019-10-10 09:03:00	903	41,407	41,407	41,315	41,315	0	210	9,470,000	144	345	0	0	0	0
2019-10-10 09:04:00	904	41,315	41,315	41,132	41,132	0	1,245	56,030,000	382	356	0	0	0	0
2019-10-10 09:05:00	905	41,132	41,132	41,040	41,040	0	1,295	58,100,000	475	522	0	0	0	0
2019-10-10 09:06:00	906	41,040	41,223	41,040	41,223	0	1,930	86,620,000	486	897	0	0	0	0
2019-10-10 09:07:00	907	41,040	41,132	41,040	41,132	0	380	17,060,000	525	934	0	0	0	0
2019-10-10 09:08:00	908	41,132	41,132	41,040	41,040	0	50	2,240,000	531	938	0	0	0	0
2019-10-10 09:09:00	909	41,132	41,132	41,132	41,132	0	5	220,000	531	939	0	0	0	0
2019-10-10 09:10:00	910	41,132	41,223	41,132	41,223	0	1,070	48,050,000	531	1,153	0	0	0	0
2019-10-10 09:11:00	911	41,223	41,223	41,040	41,040	0	1,750	78,570,000	877	1,157	0	0	0	0
2019-10-10 09:12:00	912	41,132	41,132	41,132	41,132	0	45	2,020,000	877	1,166	0	0	0	0
2019-10-10 09:13:00	913	41,132	41,132	41,040	41,040	0	160	7,170,000	904	1,171	0	0	0	0
2019-10-10 09:14:00	914	41,040	41,132	41,040	41,040	0	55	2,470,000	914	1,172	0	0	0	0
2019-10-10 09:15:00	915	41,132	41,132	41,132	41,132	0	685	30,750,000	914	1,309	0	0	0	0
2019-10-10 09:16:00	916	41,040	41,223	41,040	41,223	0	835	37,460,000	990	1,400	0	0	0	0
2019-10-10 09:17:00	917	41,132	41,132	41,132	41,132	0	805	36,150,000	1,151	1,400	0	0	0	0
2019-10-10 09:18:00	918	41,132	41,132	41,132	41,132	0	505	22,670,000	1,211	1,441	0	0	0	0
2019-10-10 09:19:00	919	41,132	41,132	41,040	41,040	0	60	2,690,000	1,218	1,446	0	0	0	0
2019-10-10 09:20:00	920	41,132	41,132	41,040	41,040	0	40	1,800,000	1,221	1,451	0	0	0	0
2019-10-10 09:21:00	921	41,040	41,132	41,040	41,132	0	45	2,010,000	1,228	1,453	0	0	0	0
2019-10-10 09:22:00	922	41,040	41,132	40,949	40,949	0	1,730	77,500,000	1,556	1,471	0	0	0	0
2019-10-10 09:23:00	923	40,949	40,949	40,949	40,949	0	30	1,340,000	1,562	1,471	0	0	0	0
2019-10-10 09:24:00	924	40,949	40,949	40,949	40,949	0	35	1,570,000	1,569	1,471	0	0	0	0

종목 선정



- 삼성전자

코스피 1위 글로벌 전자 기업
(반도체, 가전, 모바일 등)

삼성전자

카카오

- 네이버

국내 1위 포털 서비스
(광고, 쇼핑, 디지털 간편 결제, IT
인프라 및 기업형 솔루션 제공)



네이버

현대모비스

- 카카오

국내 시장 점유율 1위 카카오톡
(커머스, 모빌리티, 페이, 게임,
뮤직, 콘텐츠 등)

- 현대모비스

폴리우레탄 수지 및 전자, 자동차,
다양한 정밀화학제품을 생산·판매

미래 산업 핵심 분야 종목 선택

04

딥러닝 모델 생성



GPU 설치



< CUDA, cuDNN 설치 >

NVIDIA GeForce GTX 1050

CUDA Toolkit

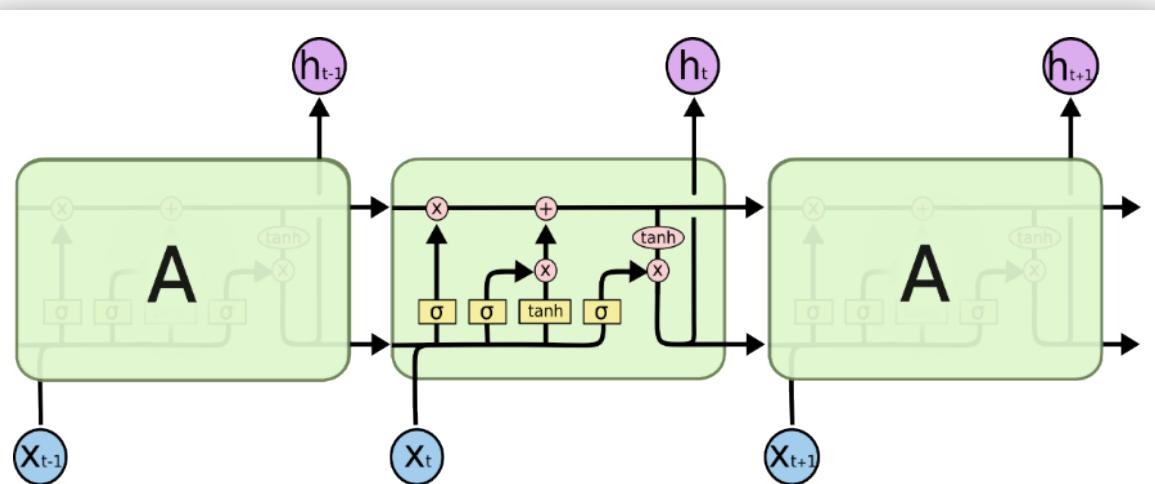
[Download cuDNN >](#)

< GPU 실행 확인 >

```
1 from tensorflow.python.client import device_lib  
2 device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"  
device_type: "CPU"  
memory_limit: 268435456  
locality {  
}  
incarnation: 9064427933454960484,  
name: "/device:GPU:0"  
device_type: "GPU"  
memory_limit: 1406107239  
locality {  
    bus_id: 1  
    links {  
    }  
}  
incarnation: 18008221933648879209  
physical_device_desc: "device: 0, name: GeForce GTX 1050, pci bus id: 0000:01:00.0,  
compute capability: 6.1"]
```

LSTM(Long Short Term Memory) Model



- LSTM(**Long Short-Term Memory**)는 순환 신경망(RNN) 기법의 하나로 셀, 입력 게이트, 출력 게이트, 망각 게이트를 이용해 기존 순환 신경망(RNN)의 문제인 **기울기 소멸 문제(vanishing gradient problem)**을 방지하도록 개발
- LSTM도 RNN과 똑같이 체인과 같은 구조를 가지고 있지만, 각 반복 모듈은 다른 구조를 갖고 있음
- 단순한 neural network layer 한 층 대신에, 4개의 layer가 서로 정보를 주고 받도록 되어 있음

LSTM(Long Short Term Memory) – 카카오(1)



```
Model: "sequential"
Layer (type)      Output Shape       Param #
=====
lstm (LSTM)      (None, 1, 512)     1087488
lstm_1 (LSTM)    (None, 256)        787456
dropout (Dropout) (None, 256)        0
dense (Dense)    (None, 1)          257
=====
Total params: 1,875,201
Trainable params: 1,875,201
Non-trainable params: 0
```

```
1 from tensorflow import keras
2
3 # Compiling
4 model.compile(optimizer=Adam(learning_rate=0.01, decay=1e-7), loss='mean_squared_error')
5
6 checkpoint_cb = keras.callbacks.ModelCheckpoint('005930_best_min_model.h5')
7 early_stopping_cb = keras.callbacks.EarlyStopping(
8     patience=10,
9     restore_best_weights=True
10)
11
12 # Fitting to the training set
13 hist = model.fit(
14     X_train
15     , y_train
16     , epochs=50
17     , batch_size=256
18     , validation_data=(X_val, y_val)
19     , verbose=1
20     , callbacks=[checkpoint_cb, early_stopping_cb]
21 )
```



- 정확도가 낮아 층을 추가하여 성능 향상을 시도



LSTM(Long Short Term Memory) – 카카오(2)

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 1, 512)	1087488
lstm_4 (LSTM)	(None, 1, 256)	787456
lstm_5 (LSTM)	(None, 128)	197120
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 2,072,193
 Trainable params: 2,072,193
 Non-trainable params: 0

```

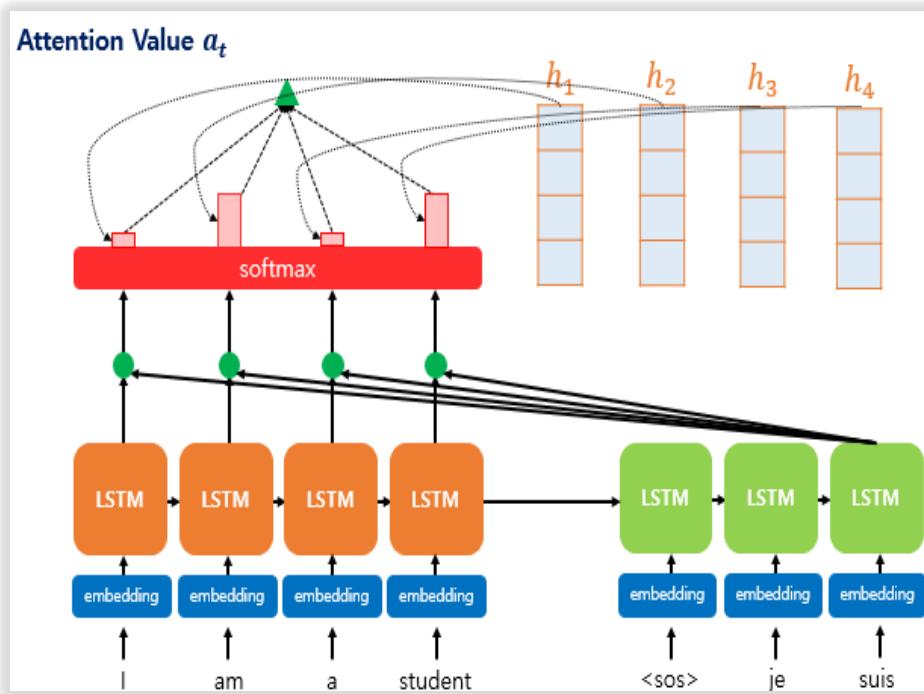
1 from tensorflow import keras
2
3 # Compiling
4 model.compile(optimizer=Adam(learning_rate=0.01, decay=1e-7), loss='mean_squared_error')
5
6 checkpoint_cb = keras.callbacks.ModelCheckpoint('005930_best_min_model.h5')
7 early_stopping_cb = keras.callbacks.EarlyStopping(
8     patience=10,
9     restore_best_weights=True
10 )
11
12 # Fitting to the training set
13 hist = model.fit(
14     X_train
15     , y_train
16     , epochs=50
17     , batch_size=256
18     , validation_data=(X_val, y_val)
19     , verbose=1
20     , callbacks=[checkpoint_cb, early_stopping_cb]
21 )

```



- 층을 추가하고 **hyperparameter**를 조정하여 좀 더 나은 결과를 도출함

Attention Mechanism



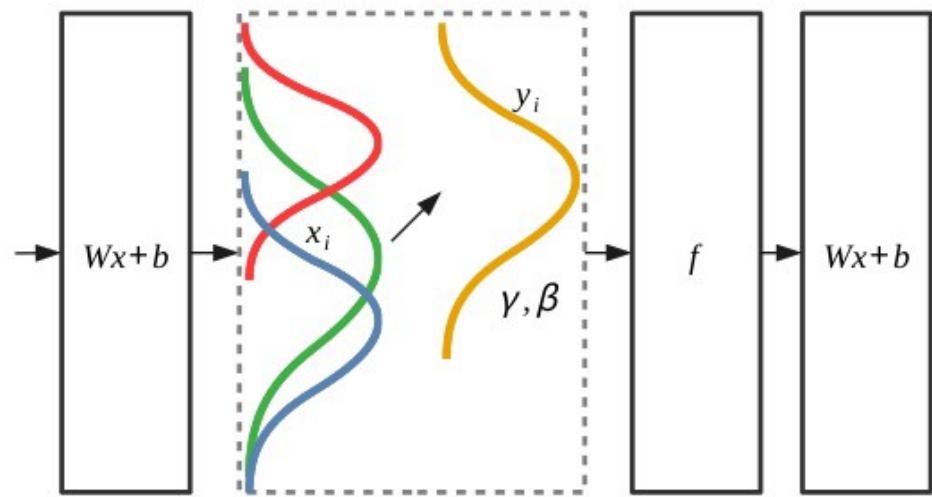
- seq2seq 모델은 인코더에서 입력 시퀀스를 **context vector**라는 하나의 고정된 크기의 벡터 표현으로 압축하고, 디코더는 이 context vector를 통해서 출력 시퀀스를 생성
- RNN에 기반한 seq2seq 모델은 **하나의 고정된 크기의 벡터에 모든 정보를 압축하려고 하니까 정보 손실이 발생하고 RNN의 고질적인 문제인 기울기 소실(Vanishing Gradient) 문제 발생**
- 어텐션은 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 전체 입력 문장을 다시 한 번 참고 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야 할 단어와 연관이 있는 입력 단어 부분을 좀 더 집중(attention)

Batch Normalization



Batch normalization

Ensure the output statistics of a layer are fixed.



- 배치 정규화는 학습 시 배치 단위의 평균과 분산들을 차례대로 받아 이동 평균과 이동 분산을 저장했다가 테스트 할 때는 해당 배치의 평균과 분산을 구하지 않고 구했던 평균과 분산으로 정규화를 수행
- 배치 정규화는 각 층에서 활성화 함수를 통과하기 전에 수행
- 입력에 대해 평균을 0으로 만들고, 정규화된 데이터에 대해서 scale과 shift를 수행 (이 때 매개변수 γ 는 스케일을 위해 사용하고, β 는 시프트에 사용하여 다음 레이어에 일정한 범위의 값들만 전달되게 함)
- 장점은 기울기 소실 문제 크게 개선, 가중치 초기화에 대해 덜 민감, 훨씬 큰 학습률 사용 가능 ☺ 학습 속도 개선
- 한계는 너무 작은 배치 크기에서는 잘 동작 안하기 때문에 미니 배치 크기에 의존적, RNN에 적용 어려움

Attention & BatchNormalization – 카카오(3)



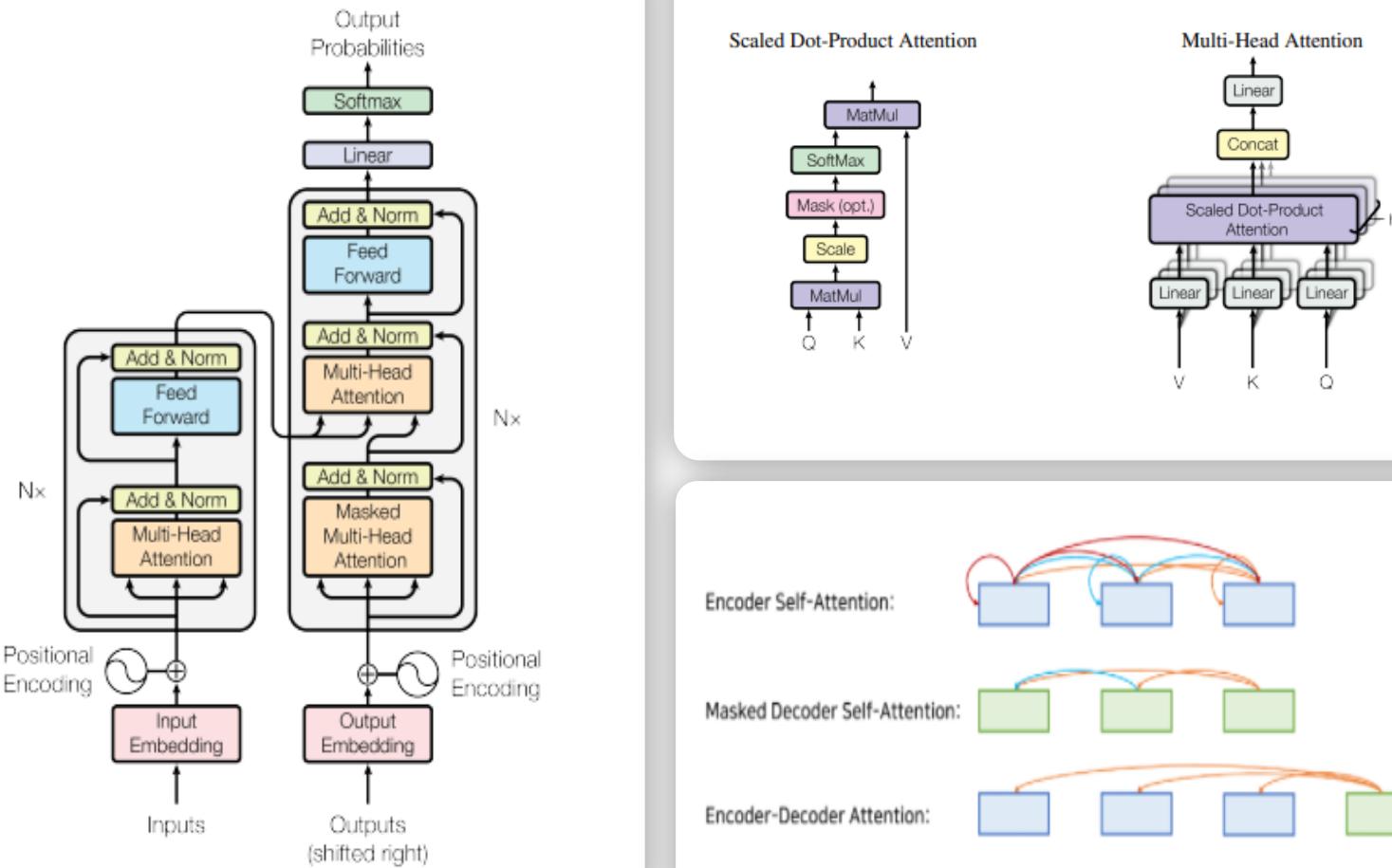
```
Model: "sequential"
Layer (type)      Output Shape       Param #
seq_self_attention (SeqSelfA (None, None, 1)    129
lstm (LSTM)        (None, 16)          1152
batch_normalization (BatchNo (None, 16)    64
dense (Dense)      (None, 2)           34
=====
Total params: 1,379
Trainable params: 1,347
Non-trainable params: 32
```

```
1 model.compile(
2     loss='mse'
3     , optimizer='adam'
4     , metrics=['mae', 'mape']
5 )
6 hist = model.fit(
7     train_X
8     , train_y
9     , epochs=training_cnt
10    , batch_size=16
11    , validation_data=(val_X, val_y)
12    , callbacks=[checkpoint_cb, earlystopping_cb]
13 )
```



- **Attention과 BatchNormalization**을 적용했을 때 기존 LSTM에 비해 성능이 향상되었음

Transformer Architecture



- Seq2seq와 유사한 **Transformer** 구조 사용
- 제안하는 Scaled Dot-Product Attention과 이를 병렬로 연결한 Multi-Head Attention 블록이 알고리즘의 핵심
- 입력위치를 표현하기 위해 Positional Encoding 사용
- RNN와 BPTT와 같은 과정이 없으므로 병렬 계산 가능
- 기존의 LSTM 속도 문제 해결하기 위해 **attention**을 적용한 Transformer를 주식에 활용

Transformer 구현 - 현대모비스



```

def layer_norm(inputs, epsilon=1e-8):
    mean, variance = tf.nn.moments(inputs, [-1], keep_dims=True)
    normalized = (inputs - mean) / (tf.sqrt(variance + epsilon))

    params_shape = inputs.get_shape()[-1:]
    gamma = tf.get_variable('gamma', params_shape, tf.float32, tf.ones_initializer())
    beta = tf.get_variable('beta', params_shape, tf.float32, tf.zeros_initializer())

    outputs = gamma * normalized + beta
    return outputs

def multihead_attn(queries, keys, q_masks, k_masks, future_binding, num_units, num_heads):
    T_q = tf.shape(queries)[1]
    T_k = tf.shape(keys)[1]

    Q = tf.layers.dense(queries, num_units, name='Q')
    K_V = tf.layers.dense(keys, 2*num_units, name='K_V')
    K, V = tf.split(K_V, 2, -1)

    Q_ = tf.concat(tf.split(Q, num_heads, axis=2), axis=0)
    K_ = tf.concat(tf.split(K, num_heads, axis=2), axis=0)
    V_ = tf.concat(tf.split(V, num_heads, axis=2), axis=0)

    align = tf.matmul(Q_, tf.transpose(K_, [0,2,1]))
    align = align / np.sqrt(K_.get_shape().as_list()[-1])

    paddings = tf.fill(tf.shape(align), float('-inf'))

    key_masks = k_masks
    key_masks = tf.tile(key_masks, [num_heads, 1])
    key_masks = tf.tile(tf.expand_dims(key_masks, 1), [1, T_q, 1])
    align = tf.where(tf.equal(key_masks, 0), paddings, align)

    if future_binding:
        lower_tri = tf.ones([T_q, T_k])
        lower_tri = tf.linalg.LinearOperatorLowerTriangular(lower_tri).to_dense()
        masks = tf.tile(tf.expand_dims(lower_tri, 0), [tf.shape(align)[0], 1, 1])
        align = tf.where(tf.equal(masks, 0), paddings, align)

    align = tf.nn.softmax(align)
    query_masks = tf.to_float(q_masks)
    query_masks = tf.tile(query_masks, [num_heads, 1])
    query_masks = tf.tile(tf.expand_dims(query_masks, -1), [1, 1, T_k])
    align *= query_masks

    outputs = tf.matmul(align, V_)
    outputs = tf.concat(tf.split(outputs, num_heads, axis=0), axis=2)
    outputs += queries
    outputs = layer_norm(outputs)
    return outputs

def pointwise_feedforward(inputs, hidden_units, activation=None):
    outputs = tf.layers.dense(inputs, 4*hidden_units, activation=activation)
    outputs = tf.layers.dense(outputs, hidden_units, activation=None)
    outputs += inputs
    outputs = layer_norm(outputs)
    return outputs

```

```

def learned_position_encoding(inputs, mask, embed_dim):
    T = tf.shape(inputs)[1]
    outputs = tf.range(tf.shape(inputs)[1]) # (T_q)
    outputs = tf.expand_dims(outputs, 0) # (1, T_q)
    outputs = tf.tile(outputs, [tf.shape(inputs)[0], 1]) # (N, T_q)
    outputs = embed_seq(outputs, T, embed_dim, zero_pad=False, scale=False)
    return tf.expand_dims(tf.to_float(mask), -1) * outputs

def sinusoidal_position_encoding(inputs, mask, repr_dim):
    T = tf.shape(inputs)[1]
    pos = tf.reshape(tf.range(0.0, tf.to_float(T), dtype=tf.float32), [-1, 1])
    i = np.arange(0, repr_dim, 2, np.float32)
    denom = np.reshape(np.power(10000.0, i / repr_dim), [1, -1])
    enc = tf.expand_dims(tf.concat([tf.sin(pos / denom), tf.cos(pos / denom)], 1), 0)
    return tf.tile(enc, [tf.shape(inputs)[0], 1, 1]) * tf.expand_dims(tf.to_float(mask), -1)

def label_smoothing(inputs, epsilon=0.1):
    C = inputs.get_shape().as_list()[-1]
    return ((1 - epsilon) * inputs) + (epsilon / C)

class Attention:
    def __init__(self, size_layer, embedded_size, learning_rate, size, output_size,
                 num_blocks=2, num_heads=8, min_freq=50):
        self.X = tf.placeholder(tf.float32, (None, None, size))
        self.Y = tf.placeholder(tf.float32, (None, output_size))

        encoder_embedded = tf.layers.dense(self.X, embedded_size)
        encoder_embedded = tf.nn.dropout(encoder_embedded, keep_prob=0.8)
        x_mean = tf.reduce_mean(self.X, axis=2)
        en_masks = tf.sign(x_mean)
        encoder_embedded += sinusoidal_position_encoding(self.X, en_masks, embedded_size)

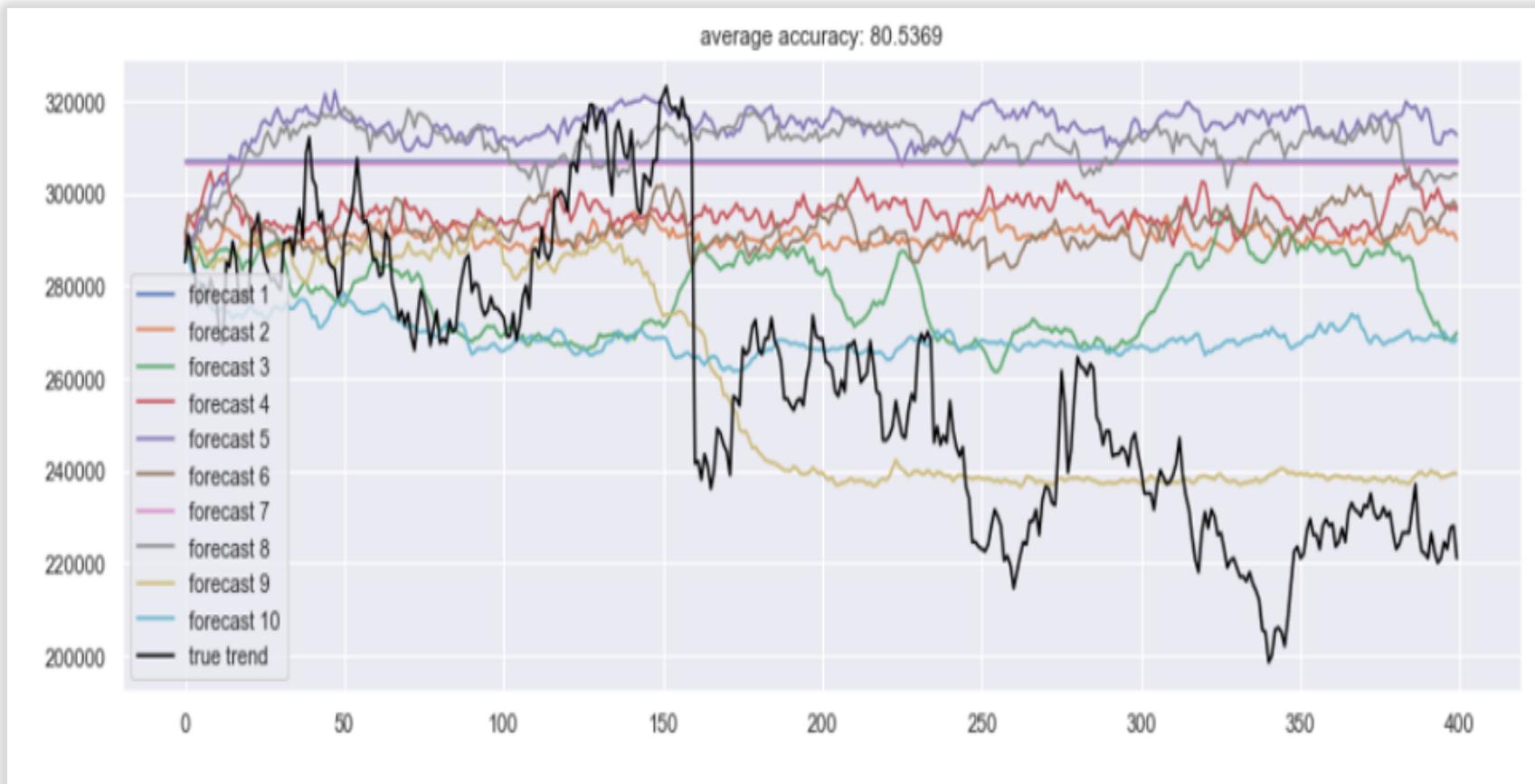
        for i in range(num_blocks):
            with tf.variable_scope('encoder_self_attn_%d' % i, reuse=tf.AUTO_REUSE):
                encoder_embedded = multihead_attn(queries=encoder_embedded,
                                                  keys=encoder_embedded,
                                                  q_masks=en_masks,
                                                  k_masks=en_masks,
                                                  future_binding=False,
                                                  num_units=size_layer,
                                                  num_heads=num_heads)

            with tf.variable_scope('encoder_feedforward_%d' % i, reuse=tf.AUTO_REUSE):
                encoder_embedded = pointwise_feedforward(encoder_embedded,
                                                          embedded_size,
                                                          activation=tf.nn.relu)

        self.logits = tf.layers.dense(encoder_embedded[-1], output_size)
        self.cost = tf.reduce_mean(tf.square(self.Y - self.logits))
        self.optimizer = tf.train.AdamOptimizer(learning_rate).minimize(
            self.cost
        )

```

Transformer 결과 - 현대모비스



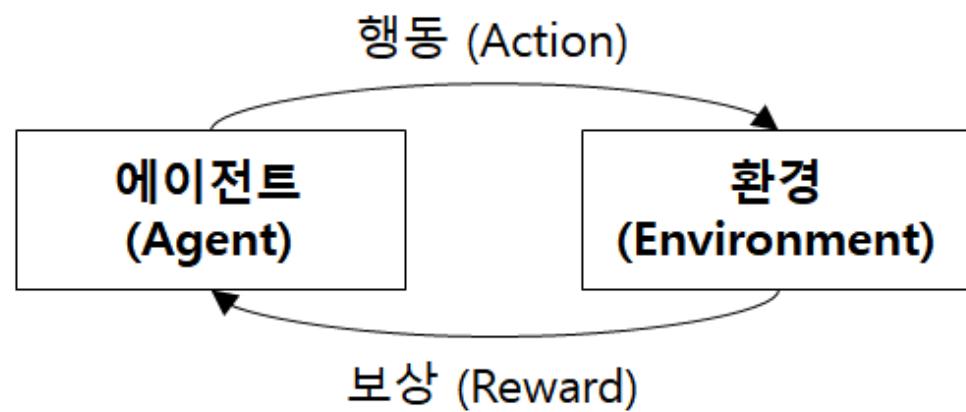
- Test data의 accuracy가 80% 밖에 되지 않아 강화학습에 미적용

05

강화학습 적용



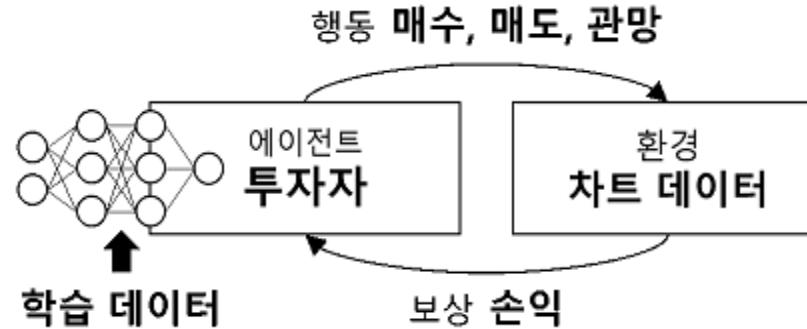
강화학습(Reinforcement Learning) 개요



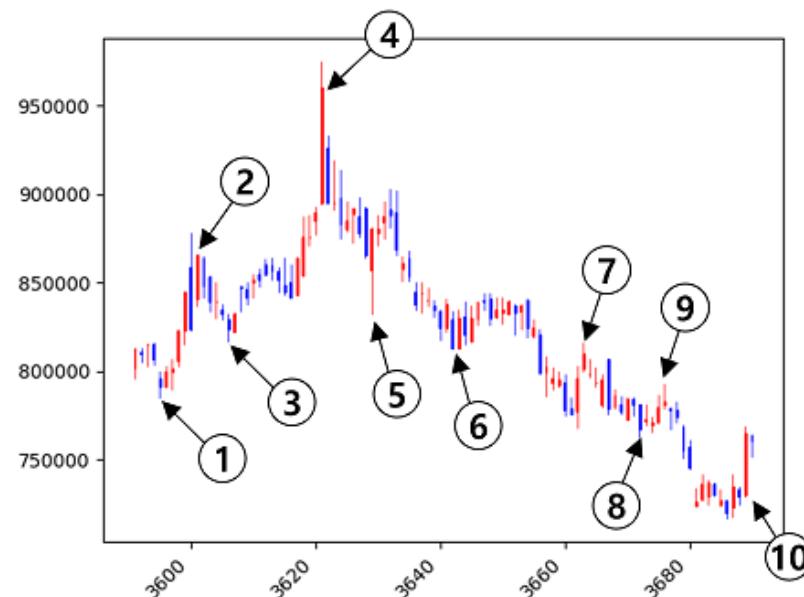
- 강화학습(**Reinforcement Learning**)은 머신러닝의 한 종류로 어떠한 환경에서 어떠한 행동을 했을 때 그것이 잘 된 행동인지 잘못된 행동인지를 나중에 판단하고 보상(또는 벌칙)을 줌으로써 반복을 통해 스스로 학습하게 하는 분야
- 구성 요소 ⓤ **환경(environment)**, **에이전트(Agent)**
- Agent는 특정 환경에서 행동(action)을 결정하고 환경은 그 결정에 대한 보상을 내림
- Agent가 행동을 결정하고 환경이 주는 보상으로 스스로 학습할 때 주로 딥러닝에서 다른 인공 신경망을 사용 환경과 에이전트의 상태 등을 입력값으로 신경망이 행동을 결정하고 보상이 있으면 이전의 입력값과 행동들을 긍정적으로 학습



강화학습 주식투자 구조



< 일봉차트 예 >

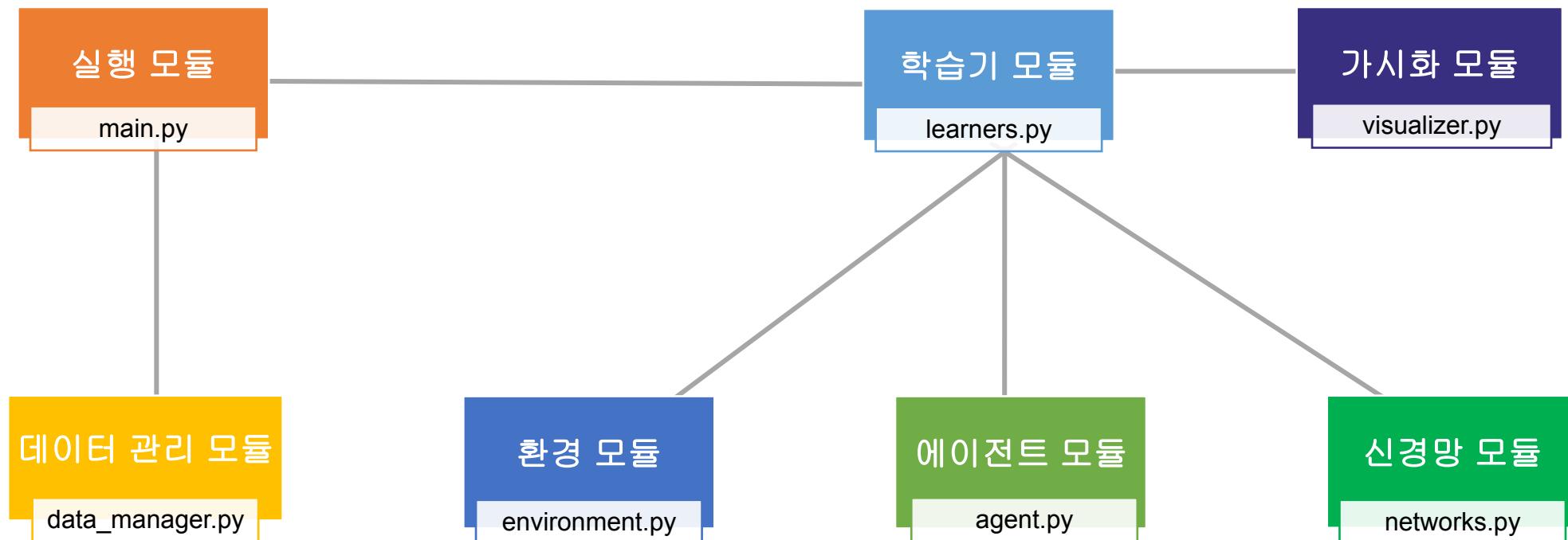


- 주식투자를 어떠한 환경에서 **매수(buy)**, **매도(sell)**, **관망(hold)**할지 강화학습에 적용
- 주식투자 강화학습에서 에이전트는 행동을 수행하는 주체인 **투자자** 역할을 함
- Action은 매수, 매도, 관망 등이 있을 수 있음
- 행동은 신경망으로 결정하고 신경망은 투자를 진행하면서 발생하는 보상과 학습 데이터로 학습
- 주식투자 강화학습에서의 환경은 다양하게 정할 수 있지만, 한 종목의 차트 데이터를 환경으로 고려
- Agent가 수행하는 행동의 결과로 발생하는 수익 또는 손실을 보상으로 하여 신경망을 학습

강화학습 모듈 구조



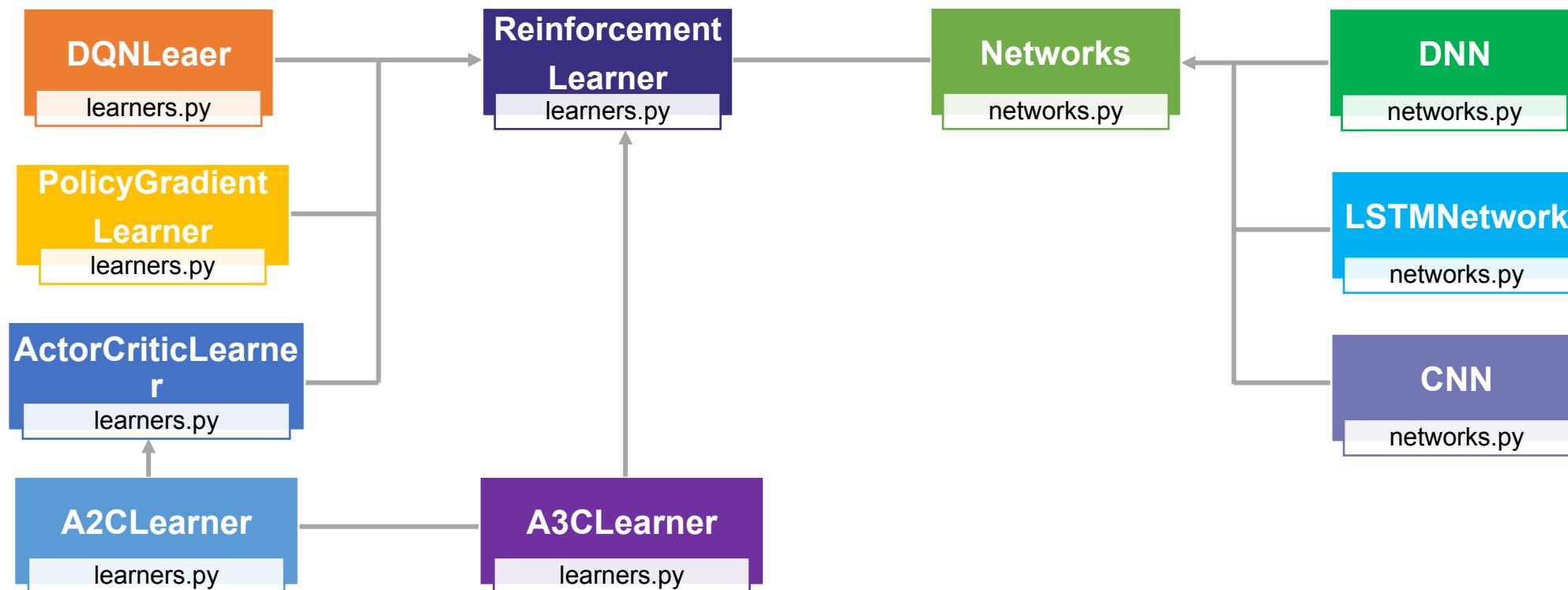
- 모듈마다 역할을 담당 / 다른 모듈과 서로 연관됨
- 몸통이 되는 모듈은 정책 학습기 모듈로 강화학습의 참여 요소인 환경, 에이전트, 신경망과 연관 관계를 맺음
- 학습 과정 기록을 위한 가시화 모듈 사용



강화학습 클래스 다이어그램



- learners 모듈의 기본 클래스인 ReinforcementLearner 클래스는 networks 모듈의 기본 클래스인 Network 클래스를 가치 신경망과 정책 신경망 학습을 위해 활용



환경 모듈 (environment.py)



- 환경 모듈(environment.py)은 투자할 종목의 차트 데이터를 관리하는 모듈

```
class Environment:  
    PRICE_IDX = 4 # 종가의 위치  
  
    def __init__(self, chart_data=None):  
        self.chart_data = chart_data  
        self.observation = None  
        self.idx = -1  
  
    def reset(self):  
        self.observation = None  
        self.idx = -1  
  
    def observe(self):  
        if len(self.chart_data) > self.idx + 1:  
            self.idx += 1  
            self.observation = self.chart_data.iloc[self.idx]  
            return self.observation  
        return None  
  
    def get_price(self):  
        if self.observation is not None:  
            return self.observation[self.PRICE_IDX]  
        return None  
  
    def set_chart_data(self, chart_data):  
        self.chart_data = chart_data
```

속성

- chart_data : 주식 종목의 차트 데이터
- Observation : 현재 관측치
- idx: 차트 데이터에서의 현재 위치

함수

- reset(): idx와 observation을 초기화
- observe(): idx를 다음 위치로 이동하고 observation을 업데이트
- get_price(): 현재 observation에서 종가를 획득

데이터 관리 모듈 (data_manager.py)



- 데이터 관리 모듈은 차트 데이터와 학습 데이터를 생성하는 모듈로 자질 벡터를 정의하고 데이터를 전처리

자질 벡터 정의

```
import pandas as pd
import numpy as np

COLUMNS_CHART_DATA = ['date', 'open', 'high', 'low', 'close', 'volume']

COLUMNS_TRAINING_DATA_V1 = [
    'open_lastclose_ratio', 'high_close_ratio', 'low_close_ratio',
    'close_lastclose_ratio', 'volume_lastvolume_ratio',
    'close_ma5_ratio', 'volume_ma5_ratio',
    'close_ma10_ratio', 'volume_ma10_ratio',
    'close_ma20_ratio', 'volume_ma20_ratio',
    'close_ma60_ratio', 'volume_ma60_ratio',
    'close_ma120_ratio', 'volume_ma120_ratio',
]

COLUMNS_TRAINING_DATA_V2 = [
    'per', 'pbr', 'roe',
    'open_lastclose_ratio', 'high_close_ratio', 'low_close_ratio',
    'close_lastclose_ratio', 'volume_lastvolume_ratio',
    'close_ma5_ratio', 'volume_ma5_ratio',
    'close_ma10_ratio', 'volume_ma10_ratio',
    'close_ma20_ratio', 'volume_ma20_ratio',
    'close_ma60_ratio', 'volume_ma60_ratio',
    'close_ma120_ratio', 'volume_ma120_ratio',
    'market_kospi_ma5_ratio', 'market_kospi_ma20_ratio',
    'market_kospi_ma60_ratio', 'market_kospi_ma120_ratio',
    'bond_k3y_ma5_ratio', 'bond_k3y_ma20_ratio',
    'bond_k3y_ma60_ratio', 'bond_k3y_ma120_ratio'
]
```

데이터 전처리 함수

```
def preprocess(data):
    windows = [5, 10, 20, 60, 120]
    for window in windows:
        data['close_ma{}'.format(window)] = \
            data['close'].rolling(window).mean()
        data['volume_ma{}'.format(window)] = \
            data['volume'].rolling(window).mean()
        data['close_ma%d_ratio' % window] = \
            (data['close'] - data['close_ma%d' % window]) \
            / data['close_ma%d' % window]
        data['volume_ma%d_ratio' % window] = \
            (data['volume'] - data['volume_ma%d' % window]) \
            / data['volume_ma%d' % window]

    data['open_lastclose_ratio'] = np.zeros(len(data))
    data.loc[1:, 'open_lastclose_ratio'] = \
        (data['open'][1:].values - data['close'][:-1].values) \
        / data['close'][:-1].values
    data['high_close_ratio'] = \
        (data['high'].values - data['close'].values) \
        / data['close'].values
    data['low_close_ratio'] = \
        (data['low'].values - data['close'].values) \
        / data['close'].values
    data['close_lastclose_ratio'] = np.zeros(len(data))
    data.loc[1:, 'close_lastclose_ratio'] = \
        (data['close'][1:].values - data['close'][:-1].values) \
        / data['close'][:-1].values
    data['volume_lastvolume_ratio'] = np.zeros(len(data))
    data.loc[1:, 'volume_lastvolume_ratio'] = \
        (data['volume'][1:].values - data['volume'][:-1].values) \
        / data['volume'][:-1]
        .replace(to_replace=0, method='ffill') \
        .replace(to_replace=0, method='bfill').values

    return data
```

데이터 전처리 전

	date	open	high	low	close	volume
0	1985-01-04	6511	6511	6473	6473	2232
1	1985-01-05	6434	6465	6427	6427	2167
2	1985-01-07	6434	6511	6427	6442	15417
3	1985-01-08	6434	6434	6358	6358	16879
4	1985-01-09	6312	6312	6090	6136	6488



데이터 전처리 후

close_ma5_ratio	volume_ma5_ratio	close_ma10_ratio	volume_ma10_ratio	close_ma20_ratio
0.023511	-0.235766	0.015041	-0.265695	0.005221
0.027632	-0.077210	0.024938	-0.093719	0.016262
0.005353	0.171601	0.014458	0.018455	0.005888
0.014798	-0.025572	0.033963	-0.156014	0.025500
0.008687	-0.127576	0.029650	-0.242264	0.021522
volume_ma20_ratio	close_ma60_ratio	volume_ma60_ratio	close_ma120_ratio	volume_ma120_ratio
-0.278464	-0.043202	-0.229901	-0.000607	-0.243799
-0.134493	-0.032058	-0.087239	0.010479	-0.106027
0.024924	-0.042124	0.068045	-0.000584	0.043533
-0.169687	-0.022414	-0.119372	0.019057	-0.154887
-0.257657	-0.023490	-0.214907	0.016545	-0.251375

학습기 모듈 (learners.py)



- 학습기 모듈은 ReinforcementLearner 클래스를 기본으로 다양한 강화학습 방법을 구현한 클래스

가치 신경망 함수

```
def init_value_network(self, shared_network=None,
                      activation='linear', loss='mse'):
    if self.net == 'dnn':
        self.value_network = DNN(
            input_dim=self.num_features,
            output_dim=self.agent.NUM_ACTIONS,
            lr=self.lr, shared_network=shared_network,
            activation=activation, loss=loss)
    elif self.net == 'lstm':
        self.value_network = LSTMNetwork(
            input_dim=self.num_features,
            output_dim=self.agent.NUM_ACTIONS,
            lr=self.lr, num_steps=self.num_steps,
            shared_network=shared_network,
            activation=activation, loss=loss)
    elif self.net == 'cnn':
        self.value_network = CNN(
            input_dim=self.num_features,
            output_dim=self.agent.NUM_ACTIONS,
            lr=self.lr, num_steps=self.num_steps,
            shared_network=shared_network,
            activation=activation, loss=loss)
    if self.reuse_models and \
       os.path.exists(self.value_network_path):
        self.value_network.load_model(
            model_path=self.value_network_path)
```

정책 신경망 함수

```
def init_policy_network(self, shared_network=None,
                       activation='sigmoid', loss='mse'):
    if self.net == 'dnn':
        self.policy_network = DNN(
            input_dim=self.num_features,
            output_dim=self.agent.NUM_ACTIONS,
            lr=self.lr, shared_network=shared_network,
            activation=activation, loss=loss)
    elif self.net == 'lstm':
        self.policy_network = LSTMNetwork(
            input_dim=self.num_features,
            output_dim=self.agent.NUM_ACTIONS,
            lr=self.lr, num_steps=self.num_steps,
            shared_network=shared_network,
            activation=activation, loss=loss)
    elif self.net == 'cnn':
        self.policy_network = CNN(
            input_dim=self.num_features,
            output_dim=self.agent.NUM_ACTIONS,
            lr=self.lr, num_steps=self.num_steps,
            shared_network=shared_network,
            activation=activation, loss=loss)
    if self.reuse_models and \
       os.path.exists(self.policy_network_path):
        self.policy_network.load_model(
            model_path=self.policy_network_path)
```

A2C Learner

Class

```
class A2CLearner(ActorCriticLearner):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    def get_batch(self, batch_size, delayed_reward, discount_factor):
        memory = zip(
            reversed(self.memory_sample[-batch_size:]),
            reversed(self.memory_action[-batch_size:]),
            reversed(self.memory_value[-batch_size:]),
            reversed(self.memory_policy[-batch_size:]),
            reversed(self.memory_reward[-batch_size:]))
        x = np.zeros((batch_size, self.num_steps, self.num_features))
        y_value = np.zeros((batch_size, self.agent.NUM_ACTIONS))
        y_policy = np.full((batch_size, self.agent.NUM_ACTIONS), .5)
        value_max_next = 0
        reward_next = self.memory_reward[-1]
        for i, (sample, action, value, policy, reward) \
            in enumerate(memory):
            x[i] = sample
            r = (delayed_reward + reward_next - reward * 2) * 100
            y_value[i, action] = r + discount_factor * value_max_next
            advantage = value[action] - value.mean()
            y_policy[i, action] = sigmoid(advantage)
            value_max_next = value.max()
            reward_next = reward
        return x, y_value, y_policy
```

신경망 모듈 (networks.py)



- 신경망 모듈(networks.py)은 가치 신경망과 정책 신경망으로 사용하기 위한 다양한 신경망 클래스

LSTM 클래스 : 생성자

```
class LSTMNetwork(Network):
    def __init__(self, *args, num_steps=1, **kwargs):
        super().__init__(*args, **kwargs)
        with graph.as_default():
            if sess is not None:
                set_session(sess)
            self.num_steps = num_steps
            inp = None
            output = None
            if self.shared_network is None:
                inp = Input((self.num_steps, self.input_dim))
                output = self.get_network_head(inp).output
            else:
                inp = self.shared_network.input
                output = self.shared_network.output
            output = Dense(
                self.output_dim, activation=self.activation,
                kernel_initializer='random_normal')(output)
            self.model = Model(inp, output)
            self.model.compile(
                optimizer=SGD(lr=self.lr), loss=self.loss)
```

LSTM 클래스 : 신경망 구조 설정

```
@staticmethod
def get_network_head(inp):
    output = LSTM(256, dropout=0.1,
                  return_sequences=True, stateful=False,
                  kernel_initializer='random_normal')(inp)
    output = BatchNormalization()(output)
    output = LSTM(128, dropout=0.1,
                  return_sequences=True, stateful=False,
                  kernel_initializer='random_normal')(output)
    output = BatchNormalization()(output)
    output = LSTM(64, dropout=0.1,
                  return_sequences=True, stateful=False,
                  kernel_initializer='random_normal')(output)
    output = BatchNormalization()(output)
    output = LSTM(32, dropout=0.1,
                  stateful=False,
                  kernel_initializer='random_normal')(output)
    output = BatchNormalization()(output)
    return Model(inp, output)
```

속성

- activation**: 신경망의 출력 레이어 활성화 함수 이름
- loss**: 신경망의 손실 함수
- lr**: 신경망의 학습 속도
- model**: Keras 라이브러리로 구성한 최종 신경망 모델

함수

- predict()**: 신경망을 통해 투자 행동별 가치나 확률 계산
- train_on_batch()**: 배치 학습을 위한 데이터 생성
- save_model()**: 학습한 신경망을 파일로 저장
- load_model()**: 파일로 저장한 신경망을 로드
- get_shared_network()**: 신경망의 상단부를 생성하는 클래스 함수

가시화 모듈 (visualizer.py)

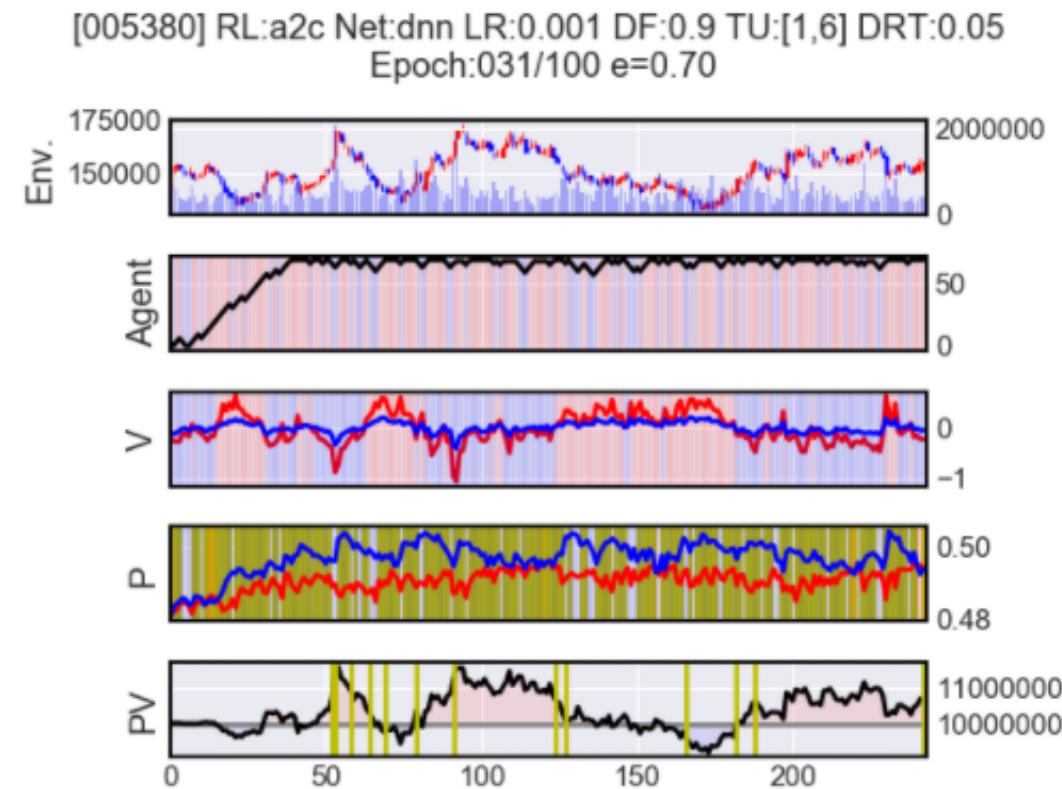


- 가시화 모듈(visualizer.py)은 신경망을 학습하는 과정에서 에이전트의 보유 주식 수, 가치 신경망 출력, 정책 신경망 출력, 투자 행동, 포트폴리오 가치 등을 시간에 따라 연속으로 보여주기 위한 시각화 기능을 담당

Visualizer 클래스 : 가시화 함수

```
# 차트 4. 정책 신경망
# 탐험을 노란색 배경으로 그리기
for exp_idx in exps:
    self.axes[3].axvline(exp_idx, color='y')
# 행동을 배경으로 그리기
_outvals = outvals_policy if len(outvals_policy) > 0 \
    else outvals_value
for idx, outval in zip(x, _outvals):
    color = 'white'
    if np.isnan(outval.max()):
        continue
    if outval.argmax() == Agent.ACTION_BUY:
        color = 'r' # 매수 빨간색
    elif outval.argmax() == Agent.ACTION_SELL:
        color = 'b' # 매도 파란색
    self.axes[3].axvline(idx, color=color, alpha=0.1)
# 정책 신경망의 출력 그리기
if len(outvals_policy) > 0:
    for action, color in zip(action_list, self.COLORS):
        self.axes[3].plot(
            x, outvals_policy[:, action],
            color=color, linestyle='-' )
```

가시화 모듈에서 생성한 가시화 결과



실행 모듈 (main.py)



- 실행 모듈(main.py)은 다양한 조건으로 강화학습을 수행할 수 있게 프로그램 인자를 구성하여 입력받은 인자에 따라 학습기 클래스를 이용해 강화학습을 수행하고 학습한 신경망들을 저장하는 메인 모듈

프로그램 인자설정

```

import os
import sys
import logging
import argparse
import json

import settings
import utils
import data_manager

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--stock_code', nargs='+')
    parser.add_argument('--ver', choices=['v1', 'v2'], default='v2')
    parser.add_argument('--rl_method',
        choices=['dqn', 'pg', 'ac', 'a2c', 'a3c'])
    parser.add_argument('--net',
        choices=['dnn', 'lstm', 'cnn'], default='dnn')
    parser.add_argument('--num_steps', type=int, default=1)
    parser.add_argument('--lr', type=float, default=0.01)
    parser.add_argument('--discount_factor', type=float, default=0.9)
    parser.add_argument('--start_epsilon', type=float, default=0)
    parser.add_argument('--balance', type=int, default=10000000)
    parser.add_argument('--num_epochs', type=int, default=100)
    parser.add_argument('--delayed_reward_threshold',
        type=float, default=0.05)
    parser.add_argument('--backend',
        choices=['tensorflow', 'plaidml'], default='tensorflow')
    parser.add_argument('--output_name', default=utils.get_time_str())
    parser.add_argument('--value_network_name')
    parser.add_argument('--policy_network_name')
    parser.add_argument('--reuse_models', action='store_true')
    parser.add_argument('--learning', action='store_true')
    parser.add_argument('--start_date', default='20170101')
    parser.add_argument('--end_date', default='20171231')
    args = parser.parse_args()

```

강화학습 실행

```

common_params = {}
list_stock_code = []
list_chart_data = []
list_training_data = []
list_min_trading_unit = []
list_max_trading_unit = []

for stock_code in args.stock_code:
    # 차트 데이터, 학습 데이터 준비
    chart_data, training_data = data_manager.load_data(
        os.path.join(settings.BASE_DIR,
        'data/{}/{}.csv'.format(args.ver, stock_code)),
        args.start_date, args.end_date, ver=args.ver)

    # 최소/최대 투자 단위 설정
    min_trading_unit = max(int(100000 / chart_data.iloc[-1]['close']), 1)
    max_trading_unit = max(int(1000000 / chart_data.iloc[-1]['close']), 1)

    # 공통 파라미터 설정
    common_params = {'rl_method': args.rl_method,
        'delayed_reward_threshold': args.delayed_reward_threshold,
        'net': args.net, 'num_steps': args.num_steps, 'lr': args.lr,
        'output_path': output_path, 'reuse_models': args.reuse_models}

```

강화학습 실행 (2)

```

# 강화학습 시작
learner = None
if args.rl_method != 'a3c':
    common_params.update({'stock_code': stock_code,
        'chart_data': chart_data,
        'training_data': training_data,
        'min_trading_unit': min_trading_unit,
        'max_trading_unit': max_trading_unit})
if args.rl_method == 'dqn':
    learner = DQNLearner(**{**common_params,
        'value_network_path': value_network_path})
elif args.rl_method == 'pg':
    learner = PolicyGradientLearner(**{**common_params,
        'policy_network_path': policy_network_path})
elif args.rl_method == 'ac':
    learner = ActorCriticLearner(**{**common_params,
        'value_network_path': value_network_path,
        'policy_network_path': policy_network_path})
elif args.rl_method == 'a2c':
    learner = A2CLearner(**{**common_params,
        'value_network_path': value_network_path,
        'policy_network_path': policy_network_path})
if learner is not None:
    learner.run(balance=args.balance,
        num_epochs=args.num_epochs,
        discount_factor=args.discount_factor,
        start_epsilon=args.start_epsilon,
        learning=args.learning)
    learner.save_models()
else:
    list_stock_code.append(stock_code)
    list_chart_data.append(chart_data)
    list_training_data.append(training_data)
    list_min_trading_unit.append(min_trading_unit)
    list_max_trading_unit.append(max_trading_unit)

```

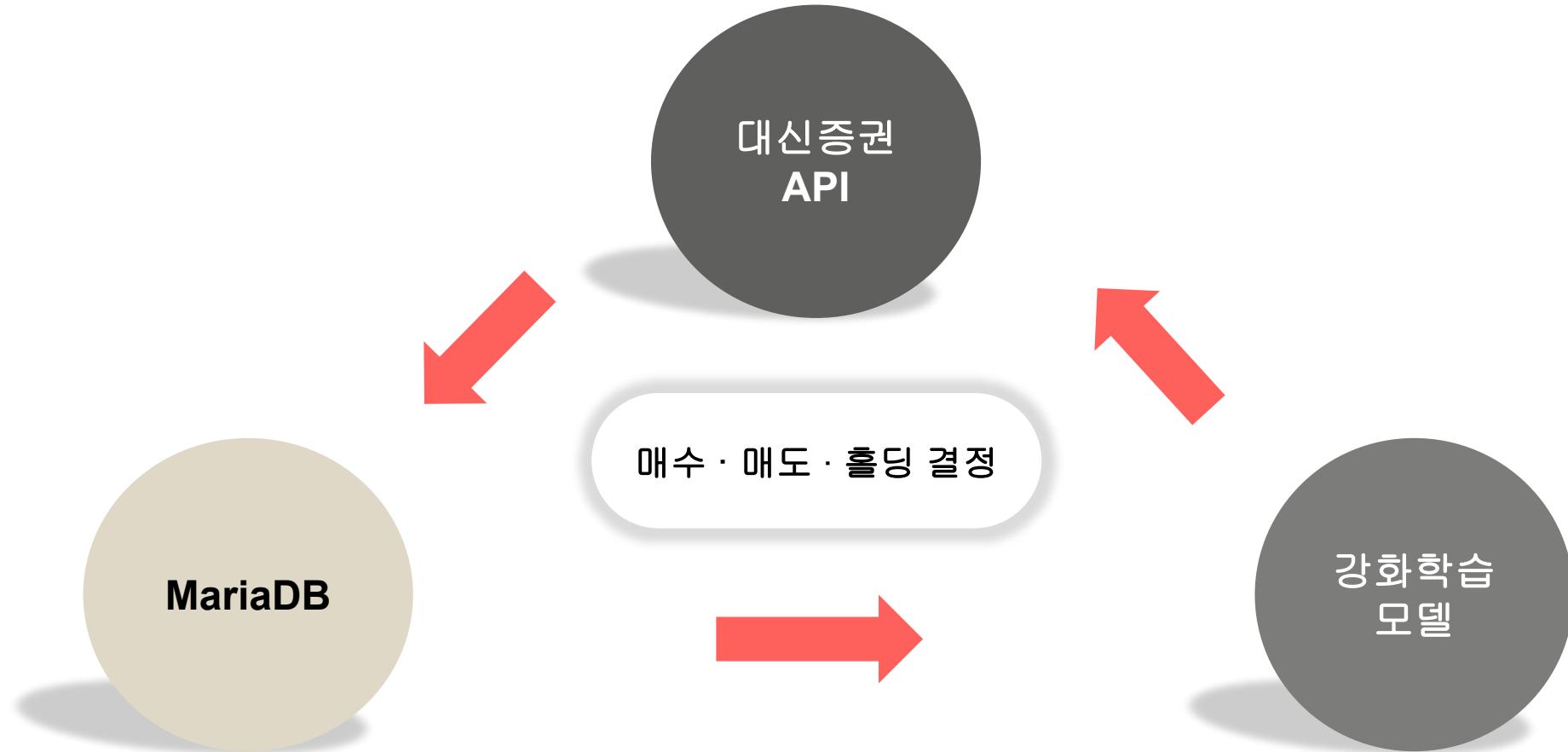
06

대신증권 API 연결





강화학습 모델 - API 연결 모식도



06 대신증권 API 연결

대신증권 Creon HTS



06 대신증권 API 연결

실시간 거래 - 매수(Buy) / 매도(Sell) / 홀딩(Hold)

The image shows a Windows desktop environment with several open windows:

- Jupyter Notebook:** A window titled "jupyter 실시간 행동(11.08)" containing Python code for trading. The code uses the "stock.new.mst" module to handle stock orders. It includes logic for buying (buy), selling (sell), and holding (hold) stocks like 카카오, 기아, 현대모비스, SK하이닉스, and NH투자증권.
- Bandicam Recording Window:** A window titled "BANDICAM UNREGISTERED" showing a recording interface with various icons for video, audio, and system controls. It also displays a preview of the recorded video at "1920x1080 - (0, 0), (1920, 1080) - 디스플레이 1".
- Code Editor:** A window titled "File Edit View Navigate Code Preferences Run Tools Help Window creo" showing Python code for "predict.py". The code involves predicting stock prices and executing trades based on the prediction. It includes a loop that checks the time and sleeps if it's past 15:30.
- Terminal:** A terminal window showing command-line output related to TensorFlow and CUDA.
- Taskbar:** The bottom of the screen shows the Windows taskbar with icons for Start, Search, Task View, File Explorer, Edge, Google Chrome, FileZilla, File Manager, and a few others.

실시간 거래 체결 내역



기간 300 일 주 월 년 분 틱 1 2 5 15 30



주문#	원#	종 목 명	대출일자	신용	거래	매체구분	정정	매매구분	유형	주문수량	주문단가	체결수량	체결단가	미체결량	대비가	현재가
4105		현대모비스			매도	DL		보통		8	244,500	8	245,000	0	-3500	245,000 정상
4083		현대모비스			매수	DL		보통		10	244,500	10	244,500	0	-3500	245,000 정상
4079		현대모비스			매도	DL		보통		8	245,000	8	245,000	0	-3500	245,000 정상
4067		현대모비스			매수	DL		보통		10	245,000	10	244,650	0	-3500	245,000 정상
4061		현대모비스			매수	DL		보통		10	245,000	10	245,000	0	-3500	245,000 정상
4053		현대모비스			매수	DL		보통		10	245,000	10	244,850	0	-3500	245,000 정상
4043		현대모비스			매수	DL		보통		50	244,500	50	244,500	0	-3500	245,000 정상
4036		현대모비스			매도	DL		보통		14	245,000	14	245,000	0	-3500	245,000 정상
4026		현대모비스			매도	DL		보통		2	244,500	2	244,500	0	-3500	245,000 정상
4015		현대모비스			매수	DL		보통		10	245,000	10	245,000	0	-3500	245,000 정상
4005		현대모비스			매도	DL		보통		30	244,500	30	244,866	0	-3500	245,000 정상
3998		현대모비스			매수	DL		보통		10	244,500	10	244,500	0	-3500	245,000 정상
3990		현대모비스			매도	DL		보통		3	245,000	3	245,000	0	-3500	245,000 정상
3984		현대모비스			매도	DL		보통		3	245,000	3	245,000	0	-3500	245,000 정상
3975		현대모비스			매도	DL		보통		88	245,000	88	245,000	0	-3500	245,000 정상
3969		현대모비스			매도	DL		보통		14	244,500	14	244,500	0	-3500	245,000 정상
3952		현대모비스			매도	DL		보통		15	244,500	15	244,833	0	-3500	245,000 정상
3912		현대모비스			매수	DL		보통		10	245,000	10	244,650	0	-3500	245,000 정상
3904		현대모비스			매수	DL		보통		10	244,500	10	244,500	0	-3500	245,000 정상
3889		현대모비스			매수	DL		보통		10	245,500	10	245,500	0	-3500	245,000 정상
3878		현대모비스			매수	DL		보통		10	245,000	10	245,000	0	-3500	245,000 정상
3870		현대모비스			매수	DL		보통		10	245,500	10	245,300	0	-3500	245,000 정상
3857		현대모비스			매수	DL		보통		10	245,000	10	245,000	0	-3500	245,000 정상
3851		현대모비스			매수	DL		보통		50	245,500	50	245,060	0	-3500	245,000 정상
3844		현대모비스			매수	DL		보통		10	245,500	10	245,400	0	-3500	245,000 정상
3829		현대모비스			매수	DL		보통		10	245,000	10	245,000	0	-3500	245,000 정상
3821		현대모비스			매도	DL		보통		2	245,500	2	245,500	0	-3500	245,000 정상
3804		현대모비스			매도	DL		보통		3	245,500	3	245,500	0	-3500	245,000 정상
3791		현대모비스			매수	DL		보통		10	245,000	10	245,000	0	-3500	245,000 정상
3778		현대모비스			매도	DL		보통		46	245,000	46	245,130	0	-3500	245,000 정상
3770		현대모비스			매도	DL		보통		7	245,500	7	245,500	0	-3500	245,000 정상
3764		현대모비스			매도	DL		보통		8	245,000	8	245,000	0	-3500	245,000 정상
3757		현대모비스			매수	DL		보통		10	245,000	10	245,000	0	-3500	245,000 정상
3753		현대모비스			매도	DL		보통		167	246,000	26	246,000	141	-3500	245,000 정상
3742		현대모비스			매수	DL		보통		10	246,000	10	245,550	0	-3500	245,000 정상
3720		현대모비스			매도	DL		보통		25	245,000	25	245,660	0	-3500	245,000 정상
3714		현대모비스			매도	DL		보통		28	245,000	28	245,482	0	-3500	245,000 정상
3703		현대모비스			매도	DL		보통		31	245,000	31	245,225	0	-3500	245,000 정상
3691		현대모비스			매수	DL		보통		10	245,500	10	245,000	0	-3500	245,000 정상
3681		현대모비스			매수	DL		보통		10	245,000	10	245,150	0	-3500	245,000 정상

07

웹 구현





Web (Notion)

Stock Alphago

주식 알파고 (STOCK ALPHAGO)

Stock Alphago

Stock Alphago는 주식자동매매 시스템을 만드는 것을 목적으로 구성된 프로젝트 팀입니다

보드 · 상태별 ▾

속성 그룹화 하위 그룹화 NEW 필터 정렬 ⌂ 검색 ⌂ 새로 만들기 ▾

팀원 5

- 이승환
- 백건우
- 박세연
- 최지웅
- 이주윤

+ 새로 만들기

프로젝트 소개 2

프로젝트의 목표는 Deep-Reinforcement-Learning을 통한 모의 투자를 실시하여 수익 2% 창출을 하는 것입니다

PPT

+ 새로 만들기

딥러닝 신경망 모델 3

- LSTM
- Attention & BatchNormalization
- Transformer

+ 새로 만들기

강화학습 & 대신증권 API 2

- 강화학습 모델 시작화
- API 모의투자 자동매매

+ 새로 만들기

Web 1

- 웹페이지

+ 새로 만들기

+ 습관 그룹

Status 없음

< URL : <https://honey-bramble-be2.notion.site/f057ef8928e949738fee778dee135aa0?v=084be3874d0d4ad8a56646a55256833a> >



Web (HTML)



**Stock
Alphago**

- INTRO ■
- ABOUT
- WORK
- TOOLS

f t G in

Stock Alphago

INTRODUCING OUR TEAM

Salphago는 Stock Alphago의 준말로 주식 알파고가 되겠다는 포부를 가지고 있다

프로젝트의 목표는 Deep-Reinforcement-Learning을 통한 모의투자를 실시하여 수익 2% 창출을 하는 것이다.

08

결론



KOSPI 현황 (2020-21)

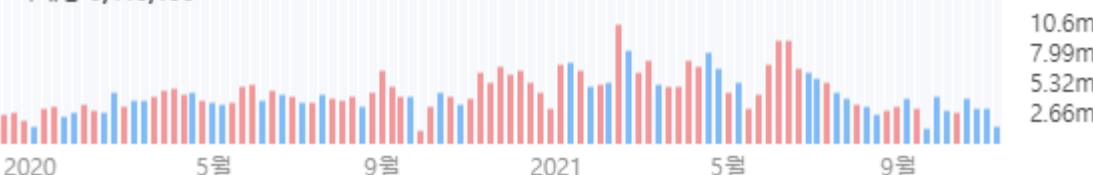


KOSPI 시 3,161 고 3,215 저 3,130 종 3,201 ▲ 67 +2.14% 거 3,410,153

이동평균 5 20 60 120



거래량 3,410,153



- 외국인 투자자 부채와 기관매도로 인한 국내 증시 약세
- COVID-19 Pandemic으로 인한 주가 과열 우려로 코스피 하락
- 恒大(헝다)그룹의 파산 우려로 인한 글로벌 금융 위기 위험
- 22년 3월 대선으로 대선 수혜주에 영향

= 전반적인 KOSPI 지수 하락세



KOSPI

개선 사항



종목 평가



SAMSUNG

HYUNDAI
MOBIS

NAVER

kakao

수익률

0.70 (%)

종목명	가장고	전일종가	현재가	전일평가	금일평가	당일순익
삼성전자	2,410	69,900	70,600	168,014	169,701	1,794
현대모비스	114	241,500	244,000		27,743	21
NAVER	224	407,000	408,500		91,265	901
카카오	2,330	125,500	127,000	291,642	295,137	3,966
합계	5,078			459,657	583,848	6,682

결과

1억의
투자금액으로
0.7% 수익을
창출

현대모비스 학습모델 비교

HYUNDAI
MOBIS

개월

전일	97,666	금일	97,383	당일손익	-284	수익률	-0.14 (%)	(천원)
종목명	가장고	전일종가	현재가	전일평가	금일평가	당일손익		
현대모비스	61	245,000	242,500	69,644	14,753	-141		

HYUNDAI
MOBIS

개월

전일	97,857	금일	98,305	당일손익	447	수익률	0.58 (%)	(천원)
종목명	가장고	전일종가	현재가	전일평가	금일평가	당일손익		
현대모비스	10	241,500	244,000	17,101	2,433	566		

HYUNDAI
MOBIS

개월

전일	374,002	금일	377,953	당일손익	3,950	수익률	1.09 (%)	(천원)
종목명	가장고	전일종가	현재가	전일평가	금일평가	당일손익		
현대모비스	264	241,500	244,000	30,107	64,247	480		

결과

학습기간이
늘어날수록
성능이 좋아짐

향후 개선사항



- ConvLSTM를 이용한 다변량 모델 생성
- 거래량으로 종목 선택 반영 필요
- Transformer Model 정확도 성능 개선 필요
- 국내외 증시 판단을 위한 뉴스 자연어 처리 필요
- 저항선과 지지선에 대한 모델 판단 적용 필요
- 10호가로 매매 가격 판단 필요



종합



목표했던 투자금액의 2%수익은
달성하지 못하였지만
모델을 더 많이 학습을시키고
거래량이 많은 종목을 사용하면
목표했던 수익률을
낼 수 있을 것이라 예상된다



감사합니다