# 얼굴 예측 서비스

## 2조 : 겐조

경기도 데이터 산업인력 양성사업
파이널 프로젝트

# 팀원 소개>>

허지훈

현세민

박세지

김연세

## Step. 1 / 문제 발견

장기 실종아동을
찾지 못하는 문제점

## Step. 2 / 주제 설정

실종아동의 오래된 사진을 통한
현재 얼굴 예측

## Step. 3 / 데이터 수집

사진 데이터셋 수집

## Step. 4 / 데이터 전처리

사진 데이터 가공

## Step. 5 / 모델 적용

GAN 모델

## Step. 6 / 최종 모델

HRFAE

# Step. 7 / 서비스 구현

## 리액트(node.js)를 활용한 서비스 구현과 시연

발표 시작합니다.

## Step. 1 / 문제 발견

장기 실종아동을
찾지 못하는 문제점

## Step. 2 / 주제 설정

실종아동의 오래된 사진을 통한
현재 얼굴 예측

## Step. 3 / 데이터 수집

사진 데이터셋 수집

# 1. 문제 발견


동영상 시청

# 1. 문제 발견


## 코퍼스(언어데이터)를 활용한 시각화

관심 주제인 **'실동아동'과 관련된 키워드**를 파악하기 위한 작업

'네이버 뉴스'에서 '실종아동' 키워드로 검색하는 크롤링 기술을 통해 코퍼스(언어데이터) 수집

-> word cloud 로 시각화

# 1. 문제 발견


## PART 2 .텍스트 마이닝

In [3]:
```
# <텍스트 마이닝 >
import requests # from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
from datetime import datetime
import time # 코드 쉬는 시간
import re # 정규표현식 : regular expression
import numpy as np
import nltk
```

### 1) 웹페이지 크롤링

In [2]:
```
# 1. 크롤링을 위한 함수 생성
def main_crawling(query, start_date, end_date, sort_type, max_page):

# 여백으로 작성시 디폴트 설정
    if query == '':
        query = '데이터 분석'
    if len(start_date) != 10:
```

In [17]:
```
# 내림차순 정렬된 단어 중 상위 50 단어
# ('해결', 15),
for word, count in sorted_word_dic[:50]:
    print("{0}({1})".format(word, count), end=" ")
```

기술(168) 개발(86) 가족(79) 얼굴(75) 몽타주(64) 찾다(61) 경찰(60) 나이(55) 경찰청(54) 확인(54) 모습(53) 장기(52) 정보(49) 변환(48) 활용(44) 변화(44) 현재(39) 예측(39) 등록(38) 만들다(37) 유전자(35) 과학기술(34) 추진(33) 분석(31) 복합(31) 배포(28) 부처(27) 치매(27) 따르다(26) 신원(26) CCTV(26) 시간(25) 발견(25) 지문(25) 같다(25) 앵커(25) 서울(24) 정부(24) 한국(23) 사건(23) 무단(22) 없다(22) 아들(22) 정통부(22) 이번(21) 찾기(21) 금지(21) 당시(21) 잃어버리다(21) 부모(20)

In [18]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from PIL import Image # "No module named 'PIL'" 에러가 발생하면 [ pip install Pillow==5.4.1 ] 로 라이브러리를 설치
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
# !pip install wordcloud 아래 wordcloud 실행이 안될시 설치
```

In [21]:
```
from wordcloud import ImageColorGenerator # Image 로부터 Color 를 생성(Generate)해내는 객체입니다.

youtube_coloring = np.array(Image.open("face.png"))
image_colors = ImageColorGenerator(youtube_coloring)
```

# 1. 문제 발견

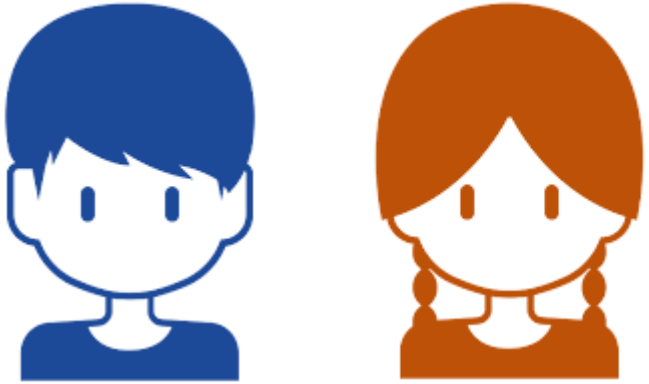

크롤링 후 시각화 결과 분석

-> **'몽타주', '얼굴', '변화'**

-> **'기술', '개발', '예측'**

과 같은 키워드가 많이 노출되고 있음을 파악

# 2. 주제 설정

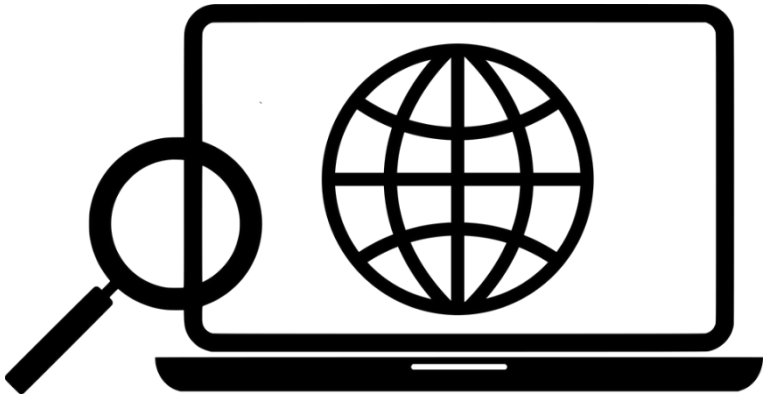


“ 어린 얼굴의 사진으로

나이 든 얼굴을 예측하자! ”

“ 누구나 사용할 수 있도록

서비스를 구축하자! ”



# 2. 주제 설정

## 인공지능 신경망을 통한

## 예측 서비스 구축

## Step. 1 / 문제 발견

장기 실종아동을
찾지 못하는 문제점

## Step. 2 / 주제 설정

실종아동의 오래된 사진을 통한
현재 얼굴 예측

## Step. 3 / 데이터 수집

사진 데이터셋 수집

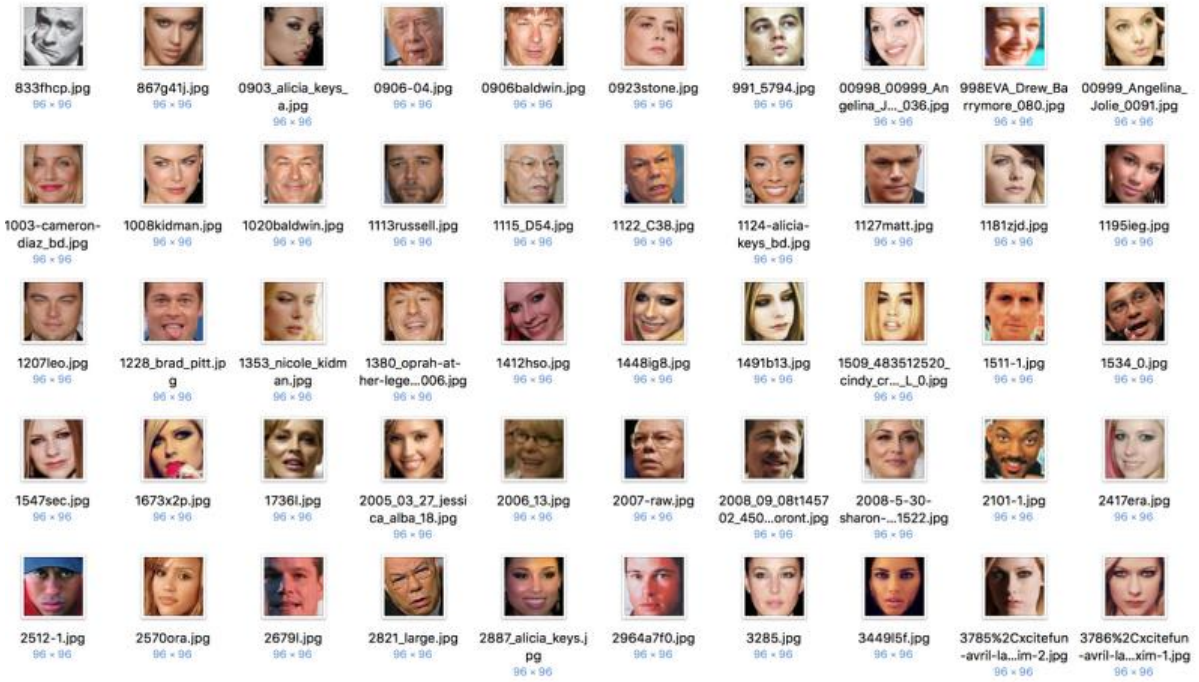
# 3. 데이터 수집

학습을 위한 얼굴 이미지 데이터 수집

# 3. 데이터 수집

Wikipedia

62,328 images




이미지의 메타 데이터(사진 찍은 년도, 생일, 경로)가 포함된 .mat 파일

이미지 데이터 출처 : wiki_crop 데이터셋 (Wikipedia)

## Step. 4 / 데이터 전처리

사진 데이터 가공

## Step. 5 / 모델 적용

GAN 모델

## Step. 6 / 최종 모델

HRFAE

# 4. 데이터 전처리

01 MatLab 형식의 파일을 Python 데이터 유형으로 변환

02 로우 데이터의 length(길이) 오류 해결

# 4. 데이터 전처리

## 01 MatLab 형식의 파일을 Python 데이터 유형으로 변환

### Scipy함수를 이용해 DF로 전환

In [41]:
```
df = pd.DataFrame(mat['wiki'][0])
df
```

Out [41]:

| | dob | photo_taken | full_path | gender | name | face_location | face_score | second_face_score |
|---|---|---|---|---|---|---|---|---|
| 0 | [[723671, 703186, 711677, 705061, 720044, 7161... | [[2009, 1964, 2008, 1961, 2012, 2012, 1971, 19... | [[[17/10000217_1981-05-05_2009.jpg], [48/10000... | [[1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0,... | [[[Sami Jauhojärvi], [Dettmar Cramer], [Marc O... | [[[[111.29109473 111.29109473 252.66993082 252... | [[4.3009623883308095, 2.6456394971903463, 4.32... | [[nan, 1.9492479052091165, nan, nan, nan, nan,... |

In [25]:
```
loaded_images = load_images(wiki_dir, images[10000:20000], (image_shape[0], image_shape[1]))
with open('load_images.pickle', 'wb') as f:
    pickle.dump(loaded_images, f)
```

# 4. 데이터 전처리

## 02 로우 데이터의 length(길이) 오류 해결

**Shape이 맞지 않는다는 오류 발생**

```
80
--> 550                 ctx=ctx)
    551         else:
    552           outputs = execute.execute_with_cancellation(

e:₩python₩lib₩site-packages₩tensorflow₩python₩eager₩execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ct
x, name)
     58     ctx.ensure_initialized()
     59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
---> 60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:
     62     if name is not None:

InternalError:  Blas GEMM launch failed : a.shape=(128, 110), b.shape=(110, 2048), m=128, n=2048, k=110
        [[node functional_3/dense_1/MatMul (defined at <ipython-input-21-b56c6151dee9>:25) ]] [Op:__inference_predict_function_1043]

Function call stack:
predict_function
```

# 4. 데이터 전처리

## 02 로우 데이터의 length(길이) 오류 해결

### Wiki_mat 파일명에 ':' 제거

In [114]:

```
df = pd.DataFrame(data=mat['dob'][0][0], columns=['dob'])
for i in mat.dtype.names:
    df[i] =mat[i][0][0]
df
```

Out [114]:

| | dob | photo_taken | full_path | gender | name | face_location | face_score | second_face_score |
|---|---|---|---|---|---|---|---|---|
| 0 | 723671 | 2009 | [17/10000217_1981-05-05_2009.jpg] | 1.0 | [Sami Jauhojärvi] | [[111.29109473290997, 111.29109473290997, 252.... | 4.300962 | NaN |
| 1 | 703186 | 1964 | [48/10000548_1925-04-04_1964.jpg] | 1.0 | [Dettmar Cramer] | [[252.48330229530742, 126.68165114765371, 354.... | 2.645639 | 1.949248 |
| 2 | 711677 | 2008 | [12/100012_1948-07-03_2008.jpg] | 1.0 | [Marc Okrand] | [[113.52, 169.83999999999997, 366.08, 422.4]] | 4.329329 | NaN |
| 3 | 705061 | 1961 | [65/10001965_1930-05-23_1961.jpg] | 1.0 | [Aleksandar Matanović] | [[1, 1, 634, 440]] | -inf | NaN |
| 4 | 720044 | 2012 | [16/10002116_1971-05-31_2012.jpg] | 0.0 | [Diana Damrau] | [[171.61031405173117, 75.57451239763239, 266.7... | 3.408442 | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 62323 | 707582 | 1963 | [49/9996949_1937-04-17_1963.jpg] | 1.0 | [Guus Haak] | [[128.92773553879837, 128.92773553879837, 320.... | 4.029268 | NaN |

# 4. 데이터 전처리

## 02 로우 데이터의 length(길이) 오류 해결

### Wiki_mat 파일명에 ':' 제거

```
from scipy.io import matlab
import pandas as pd
mat = matlab.loadmat('wiki_crop₩wiki.mat')['wiki'][0]
df_mat=pd.DataFrame(matlab.loadmat('wiki_crop₩wiki.mat')['wiki'][0])
```

```
face_location0=[mat['face_location'][0][0][i][0][0] for i in range(len(df))]
face_location1=[mat['face_location'][0][0][i][0][1] for i in range(len(df))]
face_location2=[mat['face_location'][0][0][i][0][2] for i in range(len(df))]
face_location3=[mat['face_location'][0][0][i][0][3] for i in range(len(df))]
```

```
df = pd.DataFrame(data=mat['dob'][0][0], columns=['dob'])
for i in mat.dtype.names:
    df[i] =mat[i][0][0]
df['face_location0']=face_location0
df['face_location1']=face_location1
df['face_location2']=face_location2
df['face_location3']=face_location3
```

| photo_taken | full_path | gender | name | face_location | face_score | second_face |
|---|---|---|---|---|---|---|
| 2009 | [17/10000217_1981-05-05_2009.jpg] | 1.0 | [Sami Jauhojärvi] | [[111.29109473290997, 111.29109473290997, 252.... | 4.300962 | |
| 1964 | [48/10000548_1925-04-04_1964.jpg] | 1.0 | [Dettmar Cramer] | [[252.48330229530742, 126.68165114765371, 354.... | 2.645639 | 1 |
| 2008 | [12/100012_1948-07-03_2008.jpg] | 1.0 | [Marc Okrand] | [[113.52, 169.83999999999997, 366.08, 422.4]] | 4.329329 | |
| 1961 | [65/10001965_1930-05-23_1961.jpg] | 1.0 | [Aleksandar Matanović] | [[1, 1, 634, 440]] | -inf | |
| 2012 | [16/10002116_1971-05-31_2012.jpg] | 0.0 | [Diana Damrau] | [[171.61031405173117, 75.57451239763239, 266.7... | 3.408442 | |

## Step. 4 / 데이터 전처리

사진 데이터 가공

## Step. 5 / 모델 적용

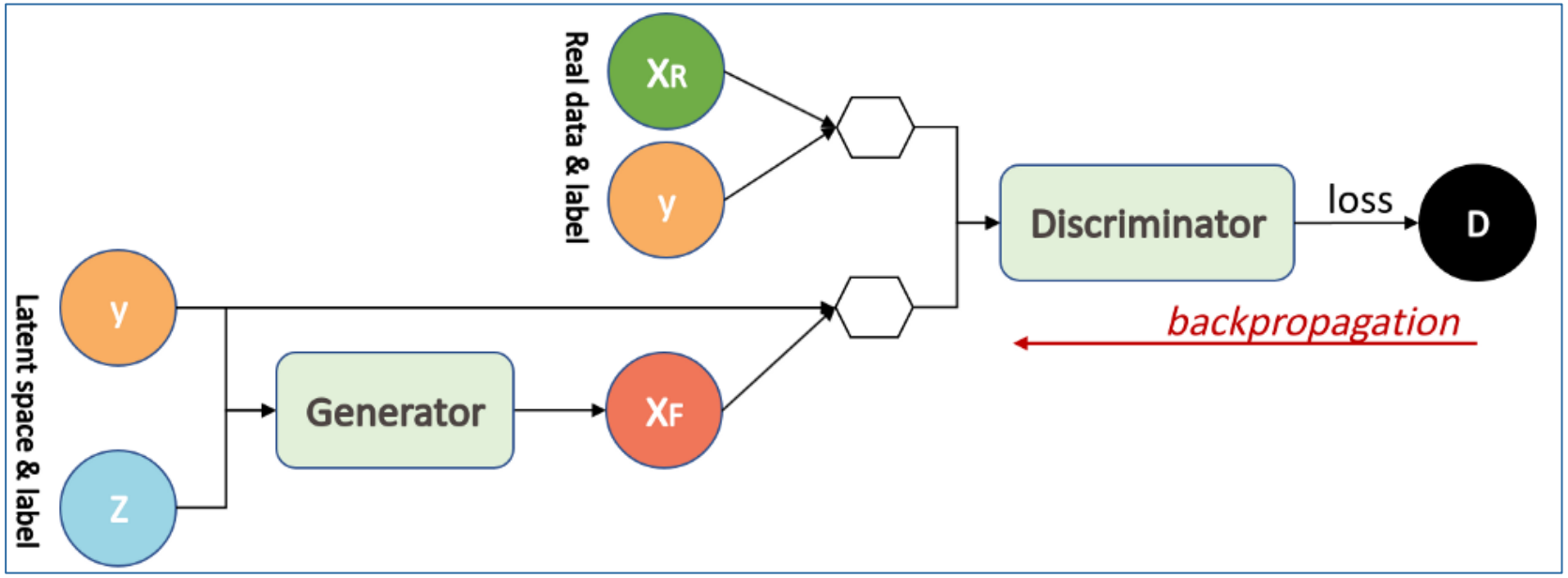
GAN 모델

## Step. 6 / 최종 모델

HRFAE

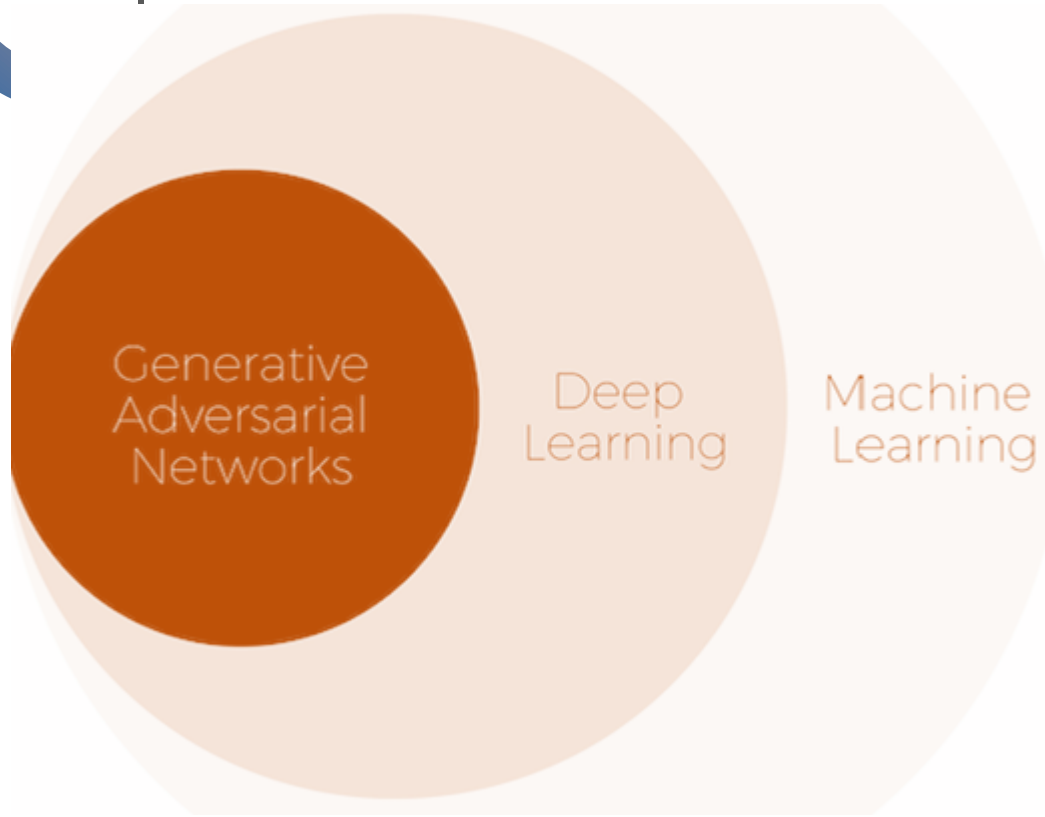
# 5. 모델 적용

## 01 CGAN(Conditional-Generative Adversarial Networks) 모델 연구



# 5. 모델 적용

## 01 cGAN(Conditional-Generative Adversarial Networks)

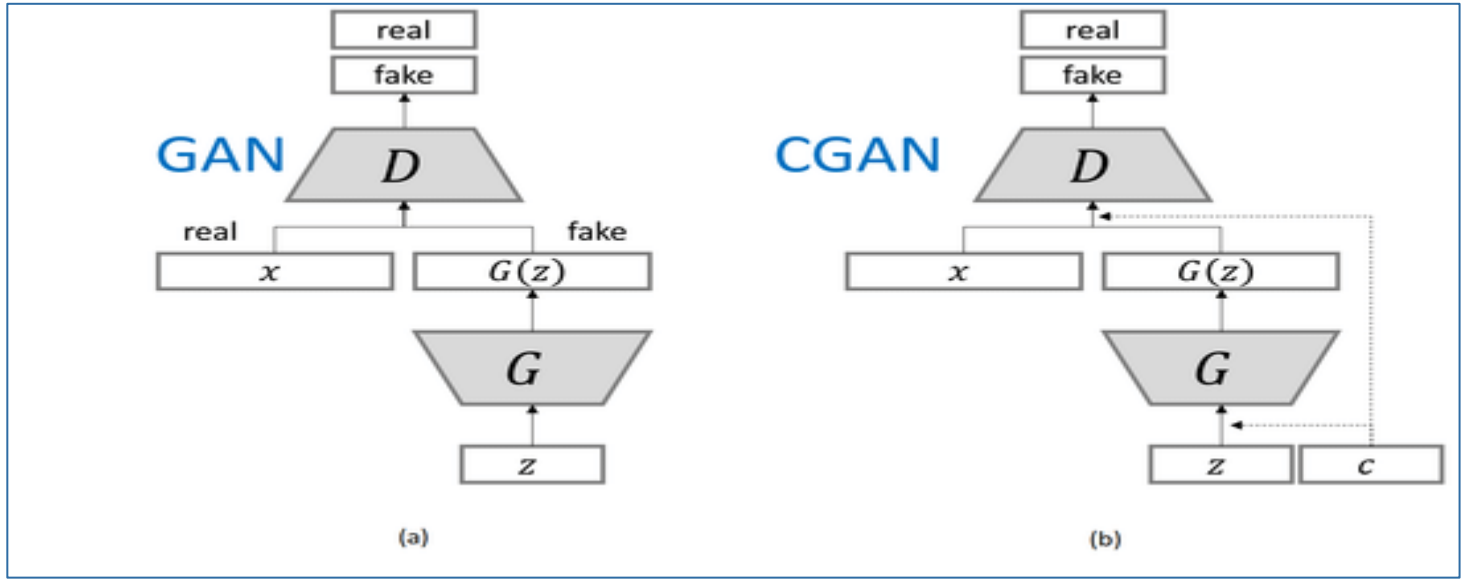



### “ 비지도 학습 ”

> 지페위조범(Generator)은 경찰을 최대한 열심히 속이려고 한다.
> 경찰(Discriminator)은 이렇게 위조된 지페를 진짜와 감별하려고(Classify) 노력한다.
>
> 이러한 경쟁 속에서 두 그룹 모두 속이고 구별하는 서로의 능력이 발전하고,
> 결과적으로는 진짜 지페와 위조 지페를 구별할 수 없을 정도(구별할 확률 pd=0.5)에 이르게 된다는 것!

# 5. 모델 적용

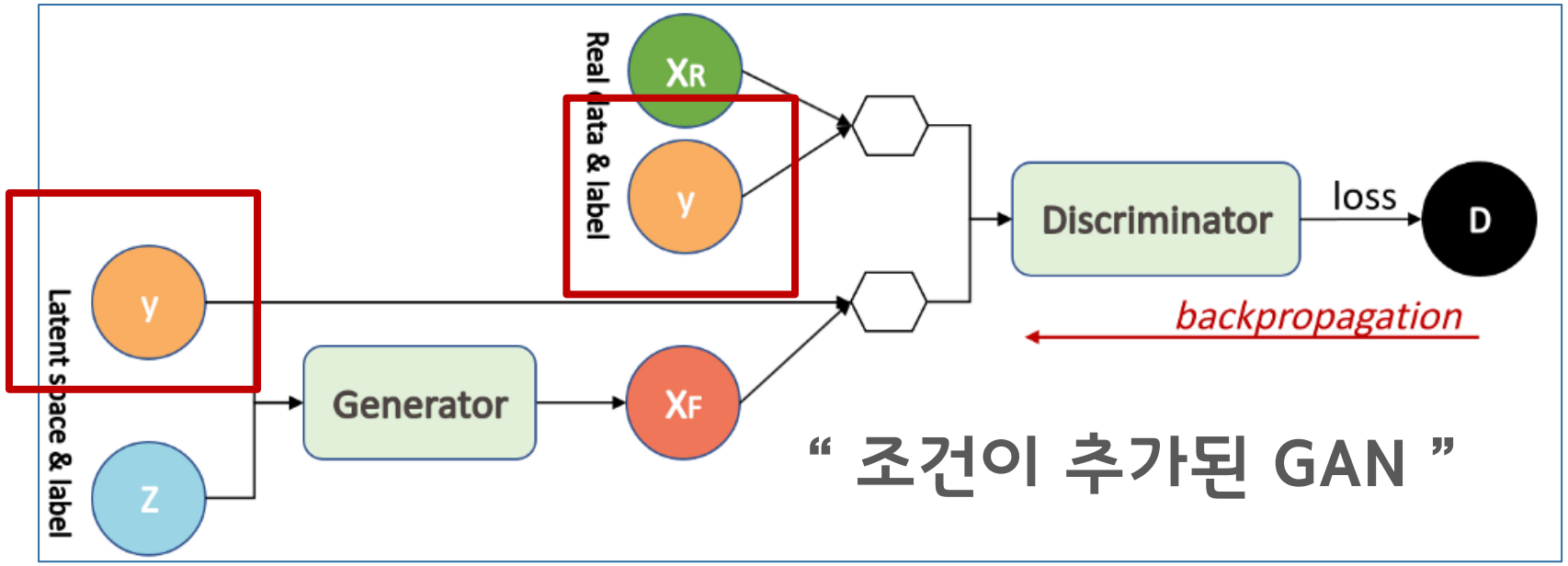
## 01 cGAN(Conditional-Generative Adversarial Networks)



# 5. 모델 적용

## 01 CGAN(Conditional-Generative Adversarial Networks)

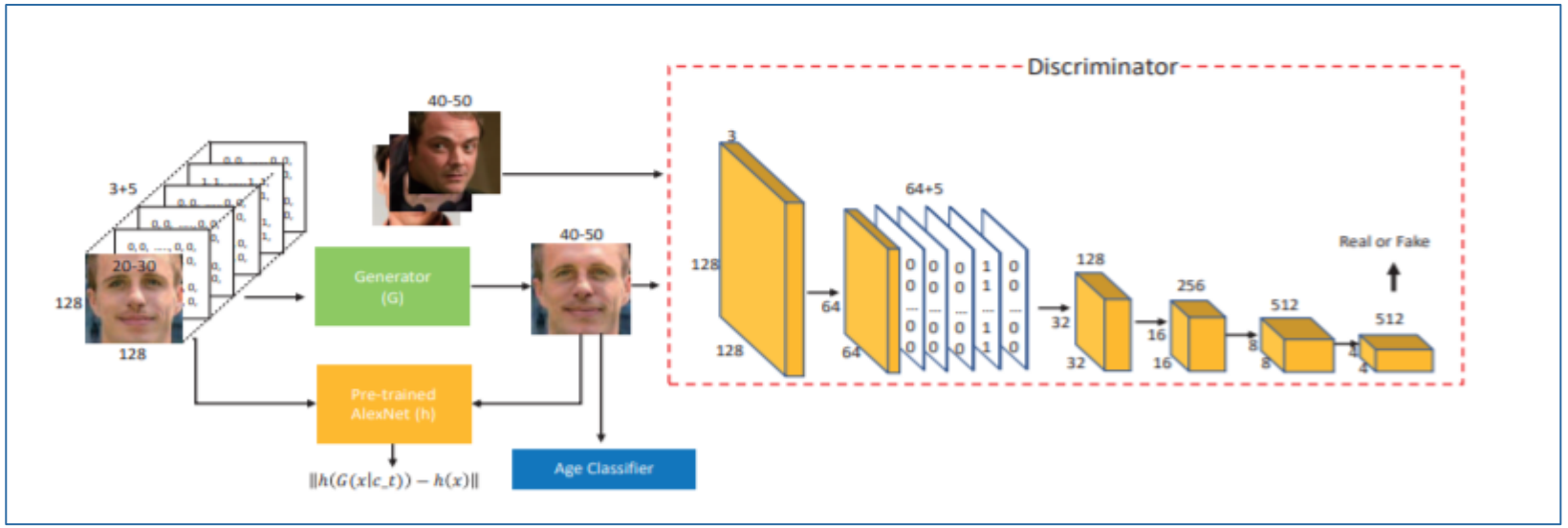
### 모델 연구



# 5. 모델 적용

02 CGAN(Conditional-Generative Adversarial Networks)

## 모델 적용




Epoch : 500 / batch-size : 128

# 5. 모델 적용

## 02 CGAN 모델 적용

| Layer | Discriminator |
|---|---|
| 1 | conv(filter = 64, kernel = 3) + leakyReLU |
| 2 | conv(filter = 128, kernel = 3) + BatchNorm + leakyReLU |
| 3 | conv(filter = 256, kernel = 3) + BatchNorm + leakyReLU |
| 4 | conv(filter = 512, kernel = 3) + BatchNorm + leakyReLU |
| 5 | conv (filter = 512, kernel = 3) + BatchNorm + leakyReLU |
| 6 | fully 1 + sigmoid |

# 5. 모델 적용

## 02 CGAN 모델 적용

| Layer | Generator |
|---|---|
| 1 | fully 2048 + leakyReLU + Dropout(0.2) |
| 2 | Fully 16384 + BatchNorm + leakyReLU + Dropout(0.2) |
| 3 | conv(filter = 128, kernel = 5) + BatchNorm + leakyReLU |
| 4 | conv(filter = 64, kernel = 5) + BatchNorm + leakyReLU |
| 5 | conv(filter = 3, kernel = 5) + BatchNorm + leakyReLU |

# 5. 모델 적용

## 02 CGAN 모델 적용

| Layer | Encoder |
|---|---|
| 1 | conv(filter = 32, kernel = 5) + leakyReLU |
| 2 | conv(filter = 64, kernel = 5) + BatchNorm + leakyReLU |
| 3 | conv(filter = 128, kernel = 5) + BatchNorm + leakyReLU |
| 4 | conv(filter = 32, kernel = 5) + BatchNorm + leakyReLU |
| 5 | Flatten |
| 6 | Fully 4096 + BatchNorm + leakyReLU |
| 7 | fully 100 |

# 5. 모델 적용

02

## CGAN 모델 적용

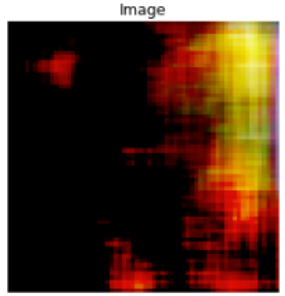
트레이닝

```
[ ]  y.shape
```

```
(62328, 10)
```

```
[ ]  batch_size=128
     for epoch in range(epochs):
         print("Epoch: {}".format(epoch))

         gen_losses = []
         dis_losses = []

         number_of_batches = int(len(loaded_images) / batch_size)
         print("Number of batches: ", number_of_batches)
     #     for index in range(10):
         for index in range(number_of_batches):
             print("Batch: {}".format(index + 1))

             images_batch = loaded_images[index * batch_size:(index + 1) * batch_size]
             images_batch = images_batch / 127.5 - 1.0
             images_batch = images_batch.astype(np.float32)

             y_batch = y[index * batch_size: (index + 1) * batch_size]
```

Image

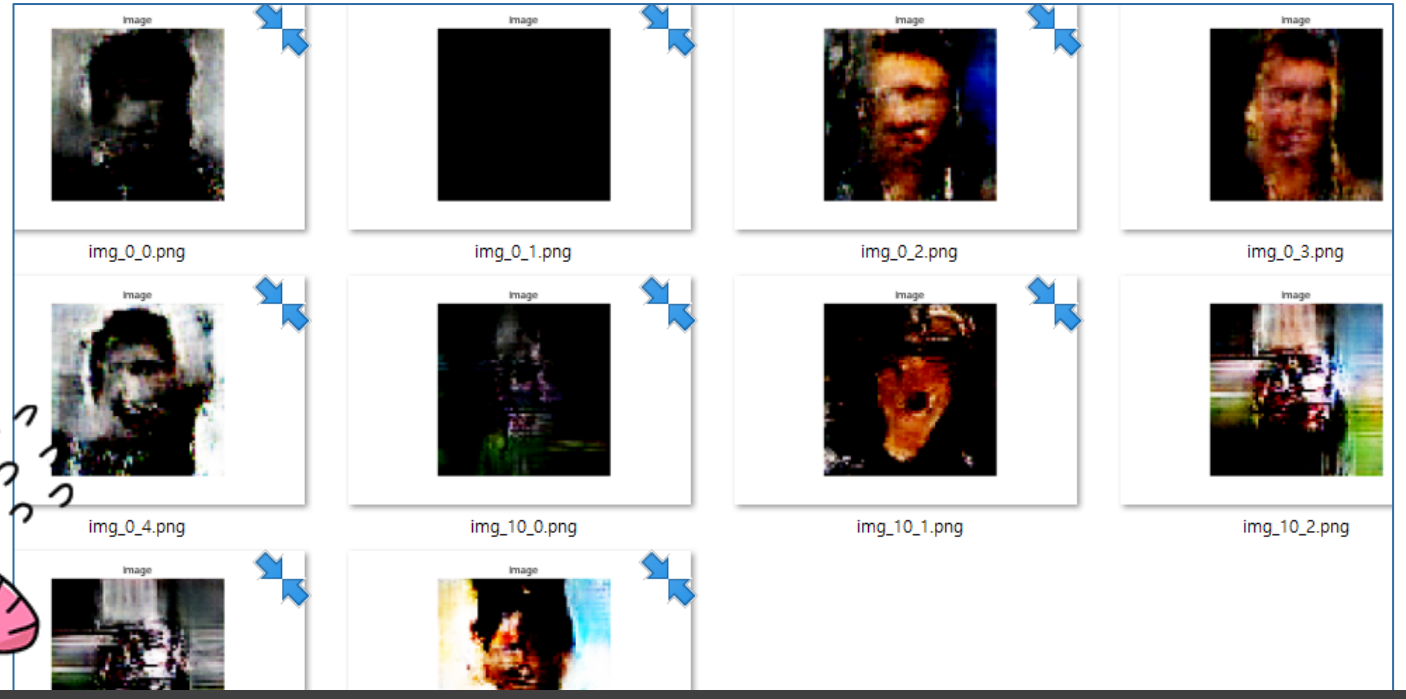


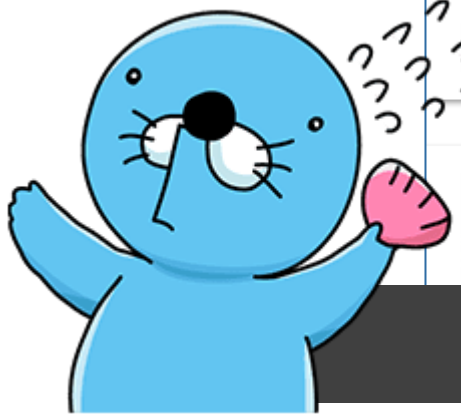
```
In [ ]: encoder = build_encoder()
        encoder.compile(loss = euclidean_distance_loss,
                        optimizer = 'adam')

        ## Load the generator network's weights
        try:
            generator.load_weights("generator.h5")
        except Exception as e:
            print("Error: ", e)

        z_i = np.random.normal(0, 1, size = (5000, z_shape))
        y = np.random.randint(low = 0, high = 1033, size = (5000, ),
                              dtype = np.int64)
        num_classes = len(set(y))
        y = np.reshape(np.array(y), [len(y), 1])
        y = to_categorical(y, num_classes = num_classes)
```

**Co-Lab에서의 GPU사용, 멀티 GPU와 같은 다양한 시도**

# 5. 모델 적용

## 02 CGAN 모델 적용



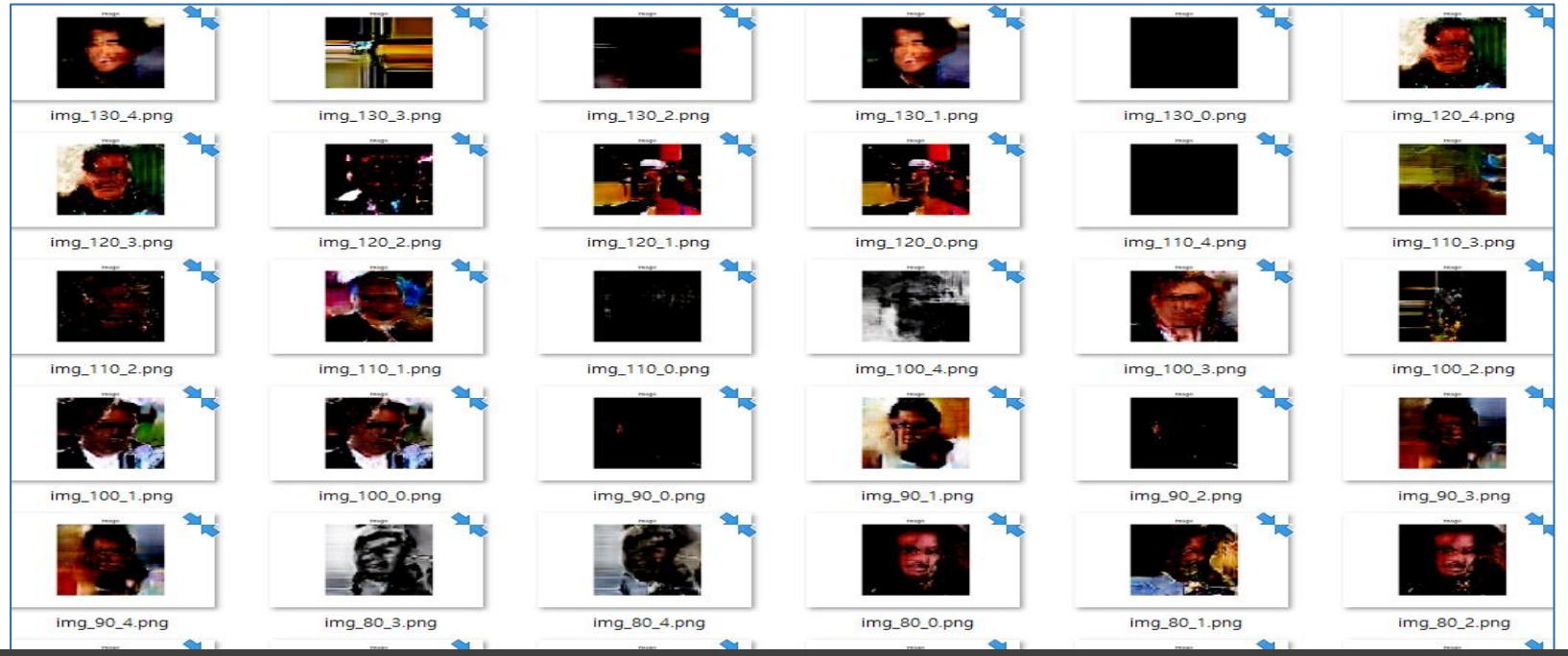
CGAN을 돌리면서 마주치던 난관들…

# 5. 모델 적용

## 02 CGAN 모델 적용

# 5. 모델 적용

## 02 CGAN 모델 적용




**포기하지 않는다!**

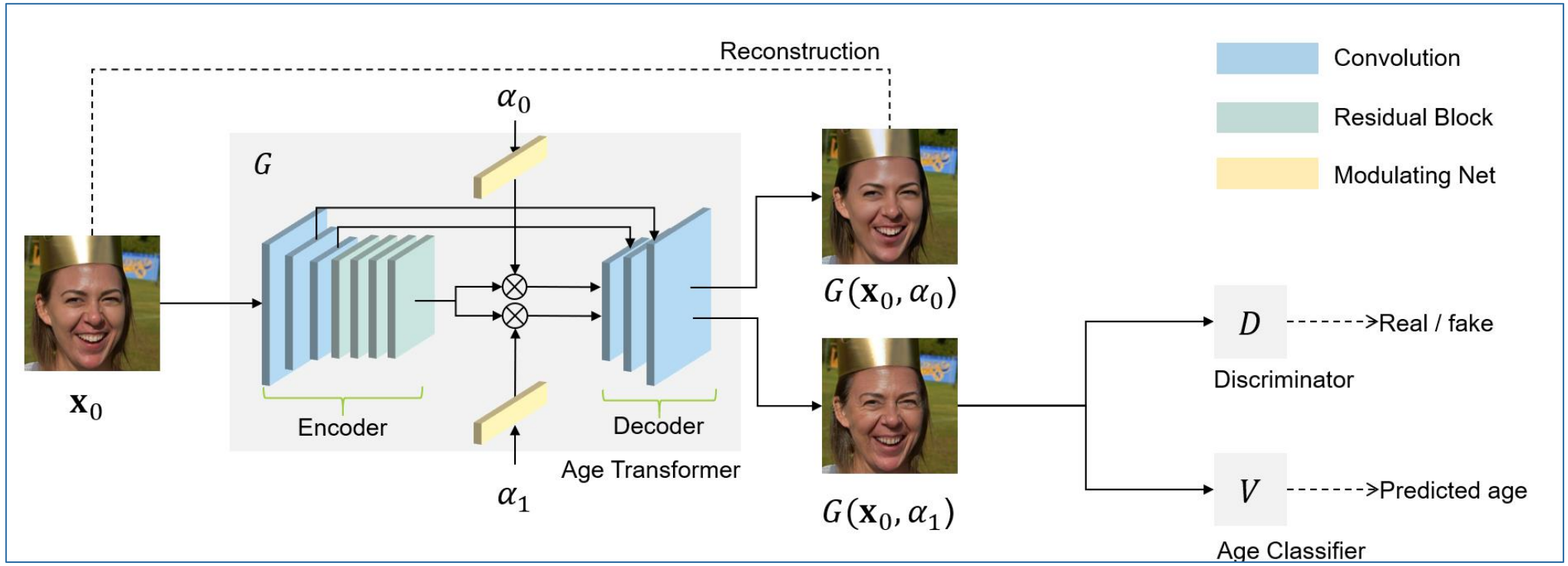
# 6. 최종 모델

HRFAE(High-Resolution Face Aging Editing)

모델 적용

# 6. 최종 모델

## HRFAE(High-Resolution Face Aging Editing) 모델



# Step. 7 / 서비스 구현

## 리액트(node.js)를 활용한 서비스 구현과 시연

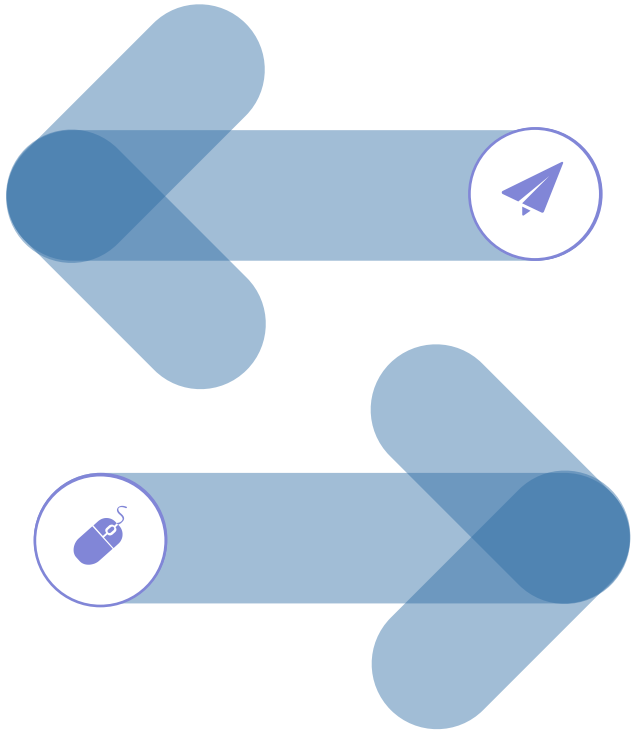
# 7. 서비스 구현

## 리액트 & 노드JS를 사용하여

## 웹사이트 구축

# 7. 서비스 구현

Front-End



Back-End

http://13.124.182.247/

# “DONE IS BETTER THAN PERPECT”

완벽하진 않지만 저희 팀원들은 완성을 위해 끝까지 노력했습니다.

# 감사합니다.

## 2조 : 겐조

경기도 데이터 산업인력 양성사업
파이널 프로젝트