케라수요일 3시
프로젝트 발표

케라수요일3시

프로세스

- (완료) 데이터 구하기
- (완료) 데이터 전처리하기 (32, 32, 3)
- GAN 모델링
- 웹페이지 구현 (시간이 남을 경우에)

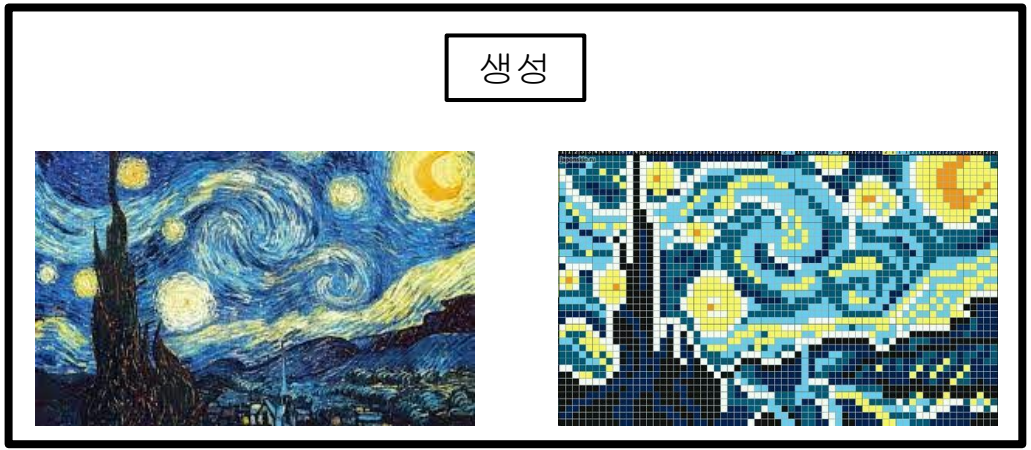
프로젝트에 대한 간략한 설명

StyleGAN 모델 통해 이미지를 픽셀풍으로 변환하기

1. 데이터 수집
   픽셀풍으로 변환된 사진 수집

2. 알려진 StyleGAN 모델을 이용해 결과물 만들어 내기
   만족할 만한 결과물을 얻을 때까지 조정

3. 웹페이지 구현
   생성 버튼을 누르면 새로운 픽셀 사진 생성해주는 웹페이지 구현



데이터 구하기

http://pixeljoint.com/ , https://opengameart.org/에서 크롤링하여 데이터 수집

```
import urllib.request
from bs4 import BeautifulSoup

print("크롤링 시작")
cnt = 0
keyword = input()
n = int(input())
cnt = 1
if n < 24:
    url = "http://pixeljoint.com/pixels/new_icons.asp?search="+str(keyword)+ "&pg=1"

    fp = urllib.request.urlopen(url)

    source = fp.read();
    fp.close()
    soup = BeautifulSoup(source, 'html.parser')

    if len(soup.find_all("div", class_="imgbox")) > 0:
        for s in soup.find_all("div", class_="imgbox"):
            if n == 0:
                break
            s = s.find("img")["src"]
            saveURL = "D:₩₩jupyter_notebook₩₩TeamProject₩₩image₩₩" + str(cnt) + ".jpg"
            realURL = "http://pixeljoint.com/" + s
            print(realURL)
            urllib.request.urlretrieve(realURL,saveURL)
            cnt += 1
            n-=1

else:

    page = int(n/24) + 1
    for i in range(1,page+1):
        url = "http://pixeljoint.com/pixels/new_icons.asp?search="+str(keyword)+ "&pg=" +str(i)
        fp = urllib.request.urlopen(url)
        source = fp.read();
        fp.close()
        soup = BeautifulSoup(source, 'html.parser')
        if len(soup.find_all("div", class_="imgbox")) > 0:
            for s in soup.find_all("div", class_="imgbox"):
                if n == 0:
                    break
                s = s.find("img")["src"]
                saveURL = "D:₩₩jupyter_notebook₩₩TeamProject₩₩image₩₩" + str(cnt) + ".jpg"
                realURL = "http://pixeljoint.com/" + s
                urllib.request.urlretrieve(realURL,saveURL)
                cnt += 1
                n-=1
```

```
title=500
url_list = list()
html_list = list()

# 51개의 page
for i in range(51):
    url = url+'&page='+str(i)

    raw = requests.get(url)
    html = BeautifulSoup(raw.text,'html.parser')

    # 한 페이지에 24개의 content
    content_title = html.find_all(class_ = "art-preview-title") #각 content의 title

    content_url_list = list() # 각 content의 url
    basic_url = 'https://opengameart.org/'

    for j in content_title:
        href = j.find('a').attrs['href']

        content_url = basic_url+href
        raw = requests.get(content_url)
        soup = BeautifulSoup(raw.text,'html.parser')   # 각 content 페이지로 이동.

        ## 사진을 찾아냄
        content = soup.find(id='maincontent').find(class_='field field-name-field-art-preview field-type-file field-label-above')
        ## title은 url에서 가장 뒤에 있는 녀석으로
        # title = content_url.split('/')[-1]

        ## title은 501부터 1000까지
        title += 1
        ## img url을 찾아냄
        img = content.find('img')['src']
        ## jpg파일로 저장하기
        img_data = requests.get(img).content
        with open(str(title) + '.jpg', 'wb') as handler:
            handler.write(img_data)

    print(i)
```

크롤링을 통해 수집한 데이터

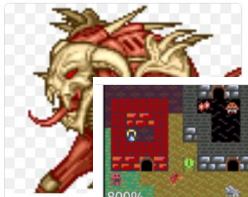
keyword - 사람

101.jpg

102.jpg

103.jpg

104.jpg



keyword - background

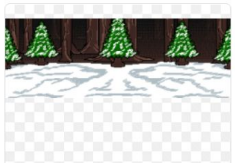
653.jpg


654.jpg

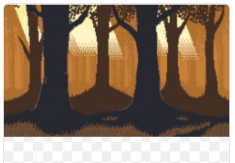

655.jpg


656.jpg


657.jpg

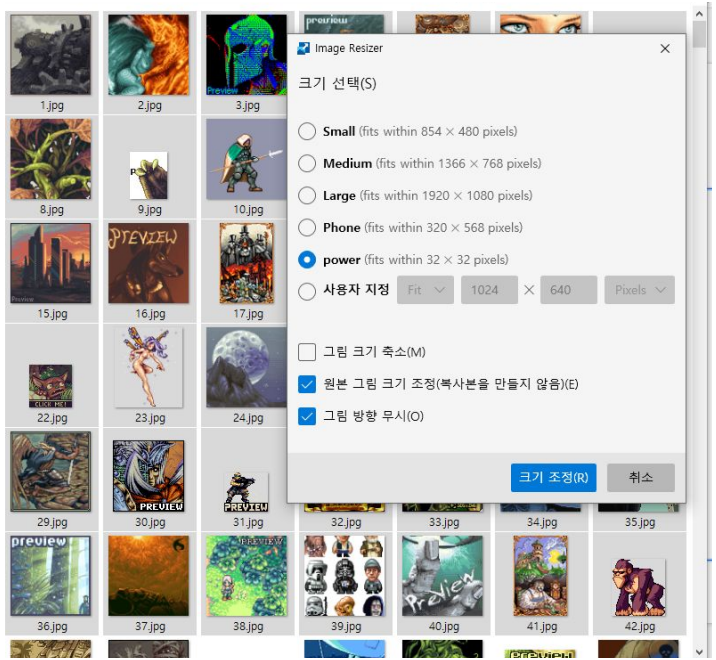

658.jpg

659.jpg


660.jpg

데이터 전처리

resize 후 RGB 채널로 변경




```
for i in lst:
    if i[-3:]=='jpg':
        img_lst.append(i)
```

```
for i in img_lst:
    x = Image.open(i)
    x.convert('RGB').save('./new/'+i,'JPEG',quality=100)
```

GAN 모델링

```
[ ]  from google.colab import drive
     drive.mount('/content/gdrive')
```

```
     Mounted at /content/gdrive
```

```
[ ]  from PIL import Image
     class Data():
       def __init__(self):
         img_data_list = []
         images = os.listdir("/content/gdrive/My Drive/datasets/images/testImg")
         for path in images:
           img = Image.open("/content/gdrive/My Drive/datasets/images/testImg/" + path)
           img_data_list.append([np.array(img).astype('float32')])
           self.x_train = np.vstack(img_data_list) / 255.0
           print(self.x_train.shape)

           # Load dataset.
     dataset = Data()
     x_train = dataset.x_train
     y_train = dataset.x_train
```

```python
import keras
from keras import layers
import numpy as np

latent_dim = 32
height = 32
width = 32
channels = 3

#생성자
generator_input = keras.Input(shape=(latent_dim,))

# 입력을 16 × 16 크기의 128개 채널을 가진 특성 맵으로 변환
x = layers.Dense(128 * 16 * 16)(generator_input)
x = layers.LeakyReLU()(x)
x = layers.Reshape((16, 16, 128))(x)

# 합성곱 층 추가
x = layers.Conv2D(256, 5, padding='same')(x)
x = layers.LeakyReLU()(x)

# 32 × 32 크기로 업샘플링
x = layers.Conv2DTranspose(256, 4, strides=2, padding='same')(x)
x = layers.LeakyReLU()(x)

# 합성곱 층을 더 추가
x = layers.Conv2D(256, 5, padding='same')(x)
x = layers.LeakyReLU()(x)
x = layers.Conv2D(256, 5, padding='same')(x)
x = layers.LeakyReLU()(x)

# 32 × 32 크기의 1개 채널을 가진 특성 맵 생성
x = layers.Conv2D(channels, 7, activation='tanh', padding='same')(x)
generator = keras.models.Model(generator_input, x)
generator.summary()
```

```python
#판별자
discriminator_input = layers.Input(shape=(height, width, channels))
x = layers.Conv2D(128, 3)(discriminator_input)
x = layers.LeakyReLU()(x)
x = layers.Conv2D(128, 4, strides=2)(x)
x = layers.LeakyReLU()(x)
x = layers.Conv2D(128, 4, strides=2)(x)
x = layers.LeakyReLU()(x)
x = layers.Conv2D(128, 4, strides=2)(x)
x = layers.LeakyReLU()(x)
x = layers.Flatten()(x)

# 드롭아웃 층을 넣는 것이 아주 중요함!
x = layers.Dropout(0.4)(x)

# 분류 층
x = layers.Dense(1, activation='sigmoid')(x)

discriminator = keras.models.Model(discriminator_input, x)
discriminator.summary()

# 옵티마이저에서 (값을 지정하여) 그래디언트 클리핑 사용
# 안정된 훈련을 위해서 학습률 감쇠를 사용
discriminator_optimizer = keras.optimizers.RMSprop(lr=0.0008, clipvalue=1.0, decay=1e-8)
discriminator.compile(optimizer=discriminator_optimizer, loss='binary_crossentropy')
```

```
#훈련 시작
import os
from keras.preprocessing import image

# 데이터를 정규화
x_train = x_train.reshape(
    (x_train.shape[0],) + (height, width, channels)).astype('float32') / 255.

iterations = 100
batch_size = 20
save_dir = '/content/gdrive/My Drive/datasets/gan_images/'
if not os.path.exists(save_dir):
    os.mkdir(save_dir)

# 훈련 반복 시작
start = 0
for step in range(iterations):
    # 잠재 공간에서 무작위로 포인트를 샘플링
    random_latent_vectors = np.random.normal(size=(batch_size, latent_dim))

    # 가짜 이미지를 디코딩함
    generated_images = generator.predict(random_latent_vectors)

    # 진짜 이미지와 연결
    stop = start + batch_size
    real_images = x_train[start: stop]
    combined_images = np.concatenate([generated_images, real_images])

    # 진짜와 가짜 이미지를 구분하여 레이블을 합침
    labels = np.concatenate([np.ones((batch_size, 1)),
                             np.zeros((batch_size, 1))])
    # 레이블에 랜덤 노이즈를 추가 아주 중요함!
    labels += 0.05 * np.random.random(labels.shape)

    # discriminator를 훈련
    d_loss = discriminator.train_on_batch(combined_images, labels)

    # 잠재 공간에서 무작위로 포인트를 샘플링
    random_latent_vectors = np.random.normal(size=(batch_size, latent_dim))

    # 모두 "진짜 이미지"라고 레이블을 만듦
    misleading_targets = np.zeros((batch_size, 1))

    # generator를 훈련(gan 모델에서 discriminator의 가중치는 동결)
    a_loss = gan.train_on_batch(random_latent_vectors, misleading_targets)
```

```
start += batch_size
if start > len(x_train) - batch_size:
  start = 0

# 중간 중간 저장하고 그래프를 그림
if step % 10 == 0:
    # 모델 가중치를 저장
    gan.save_weights('gan.h5')

    # 측정 지표를 출력
    print('스텝 %s에서 판별자 손실: %s' % (step, d_loss))
    print('스텝 %s에서 적대적 손실: %s' % (step, a_loss))

    # 생성된 이미지 하나를 저장
    img = image.array_to_img(generated_images[0] * 255., scale=False)
    img.save(os.path.join(save_dir, 'generated' + str(step) + '.png'))

    # 비교를 위해 진짜 이미지 하나를 저장
    img = image.array_to_img(real_images[0] * 255., scale=False)
    img.save(os.path.join(save_dir, 'real' + str(step) + '.png'))
```

real data

generated data

해야 할 일

- 다른 GAN 모델(style GAN, cycel GAN, …) 시도

- 웹 사이트 구현 (시간 남으면)