생체 신호를 기반으로 한 감정 분류 및 회화 재생성 시스템 연구

CUAI 3기 Deep Learning Track B팀
배병헌(제약학), 서혜련(기계공학), 이나혁(소프트웨어공학), 이보림(소프트웨어공학), 이하윤(소프트웨어공학), 이환진(에너지시스템공학)

생체 신호 중 뇌파는 의도적으로 조작이 불가능 하기 때문에 객관적인 데이터로 널리 사용되며 요즘 이러한 연구가 많이 진행되고 있다. 우리는 이런 뇌파의 특징을 이해하고 신호 데이터를 분석하여 감정을 추출한다. 이후, 사용자가 original data를 넣으면 그림 생성 모델을 이용하여 감정에 따라 새롭게 변형시킨 output을 만들어낸다.

1. 서 론

감정을 분석할 때 가장 객관적인 지표는 생체 신호이다. 생체 신호는 행동이나 표정과는 다르게 조작이 불가능 하기 때문이다.
뇌파(腦波, Electroencephalogram; EEG)는 뇌의 신경활동에 의해 발생하는 전기적 신호를 대뇌피질 또는 두피에서 기록한 것으로 뇌전도(腦電圖)라고도 한다. 뇌파는 기본적으로 주파수에 따라 8가지 대역으로 분류된다.

우리는 뇌파의 전기적 신호를 측정하여 FFT(Fast Fourier Transform)을 통해 주파수 대역별로 분류한 후, (Lightbgm model)을 이용하여 감정을 분류할 것이다. 이후 유클리디안 거리 계산을 통해 감정에 따라 분류된 명화 추천하고, 추천된 명화를 토대로 변형된 이미지를 생성할 것이다.

2. 본 론

1) 뇌파 신호 전처리

뇌파 신호의 측정은 SICHIRAY TGAM Starter Kit Brainwave Sensor를 이용하여 Sampling period가 0.01s인 전기 신호(Sample frequency = 100 Hz)를 측정했다. 위와 같이 정한 이유는 Fourier transform의 Nyquist Frequency 와 연관이 있다. 신호 처리에서 Sampling period의 절반 이상의 주파수는 측정할 수 없기 때문이다. 즉 우리는 뇌파 신호를 0.1~50Hz 내의 영역에서 분석할 것이다.
40초 동안 긍정 혹은 부정적인 감정을 유발하는 영상을 시청하며 뇌파 신호를 측정한다. 40초 중 감정이 고조된 5~25초의 뇌파 신호를 이용하여 분석을 진행했다.
뇌파 신호 분석 기법 중 주파수 대역별로 뇌파를 분류하는 Band power와 전력 스펙트럼 밀도(Power Spectral Density)를 이용하여 주파수 범위에 따른 뇌파의 활성화 정도를 측정한다.
뇌파의 시간에 따른 전기적 신호(Time domain)를 numpy에서 제공하는 FFT(Discrete Fast Fourier Transform)을 이용해 크기(Amplitude)와 주파수(Frequency)로 구성된 Frequency Domain 으로 변환한다. 변환된 신호의 PSD를 구한 후 뇌파 신호의 주파수 대역별로 크기를 합하여 기존에 정의된 뇌파의 종류로 재분류 한다. 이때 FFT의 결과는 좌우대칭이므로 양수의 값만 계산한다.

뇌파 종류와 주파수 범위[Hz] 정의 및 뇌파 별 분류

- Delta frequency : 0.1 - 4
- Theta frequency : 4 - 8
- AlphaLow frequency : 8 - 10
- AlphaHigh frequency : 10 - 13
- BetaLow frequency : 13 - 18
- BetaHigh frequency : 18 - 30
- GammaLow frequency : 30 - 40
- GammaMid frequency : 40 - 50

그림 1 주파수 범위에 따른 뇌파 분류

```
def get_data(data):
    data = data['eegRawValueVolts']
    data = data.iloc[range(100,2500,1)]
    length = len(data)
    fft = np.fft.fft(data)
    fft_spectrum = np.square(fft)
    #fft_shift = np.fft.fftshift(fft_magnitude)[1200:2400]
    fft_spec = fft_spectrum[1200:2400]
    return fft_spec
```

그림 2 뇌파 신호 전처리 코드

```python
def get_result_df(emotion, num_data, target):
    temp_df = pd.DataFrame()
    brain_wave = ['Delta', 'Theta', 'AlphaLow', 'AlphaHigh', 'BetaLow', 'BetaHigh',
                  'GammaLow', 'GammaMid']

    for n in range(num_data+1):
        name = '/Users/ihwanjin/data_set/'+ emotion + str(n) +'.csv'
        data = pd.read_csv(name, error_bad_lines=False)
        fft_spec = get_data(data)
        # 뇌파 분류 및 비율 계산
        Delta = fft_spec[start_cri:DT_cri].sum() / len(fft_spec[start_cri:DT_cri])
        Theta = fft_spec[DT_cri:TAl_cri].sum() / len(fft_spec[DT_cri:TAl_cri])
        AlphaLow = fft_spec[TAl_cri:AlAh_cri].sum() / len(fft_spec[TAl_cri:AlAh_cri])
        AlphaHigh = fft_spec[AlAh_cri:AhBl_cri].sum() / len(fft_spec[AlAh_cri:AhBl_cri])
        BetaLow = fft_spec[AhBl_cri:BlBh_cri].sum() / len(fft_spec[AhBl_cri:BlBh_cri])
        BetaHigh = fft_spec[BlBh_cri:BhGl_cri].sum() / len(fft_spec[BlBh_cri:BhGl_cri])
        GammaLow = fft_spec[BhGl_cri:GlGm_cri].sum() / len(fft_spec[BhGl_cri:GlGm_cri])
        GammaMid = fft_spec[GlGm_cri:].sum() / len(fft_spec[GlGm_cri:])
        # DataFrame 쌓기
        new_row = {'Delta':Delta, 'Theta':Theta, 'AlphaLow':AlphaLow,
           'AlphaHigh':AlphaHigh, 'BetaLow':BetaLow, 'BetaHigh':BetaHigh,
           'GammaLow':GammaLow, 'GammaMid':GammaMid}
        temp_df = temp_df.append(pd.Series(new_row), ignore_index=True)

    temp_df = temp_df[brain_wave]
    temp_df['sum'] = temp_df.sum(axis=1)
    for wave in brain_wave:
        temp_df[wave] = temp_df[wave] / temp_df['sum']
    temp_df.drop('sum', axis=1, inplace=True)
    temp_df['target'] = target

    return temp_df
```

그림 3 데이터 프레임 생성 코드

2) 감정 분류

감정을 분류하는 모델에는 이산적 모델과 연속적 모델이 있다. 이번 프로젝트에서는 감정을 단순히 화남, 슬픔 등으로 분류하는게 아니라, 세밀한 감정 분석을 위해 연속적인 모델 중 J.Posner 가 제안한 Valence-Arousal 모델을 채택하여 감정의 긍정성과 부정성을 수치로 나타낼 것이다. Valence는 감정이 얼마나 긍정적인지를 의미하고, Arousal은 그 감정의 정도가 얼마나 강렬한지를 나타낸다. 본 프로젝트에서는 감정의 Valence 만 고려하여 정규화 하였다.

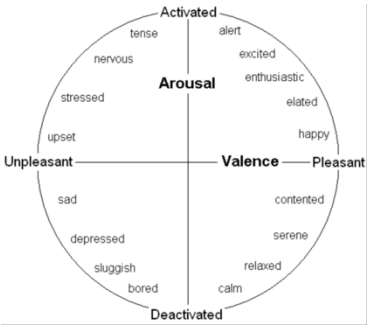


그림 4 Valence-Arousal model

3) 데이터 학습 및 감정 분류

긍정적인 영상과 부정적인 영상을 각각 Target 0, 1로 기록하고, 분류된 뇌파의 비율을 기록하며 데이터를 쌓는다. 이후 데이터를 LIGHTbgm 모델에 학습시킨다. 그 결과는 다음과 같다.

	Delta	Theta	AlphaLow	AlphaHigh	BetaLow	BetaHigh	GammaLow	GammaMid	target
0	0.041864	0.052574	0.063733	0.040696	0.073423	0.092863	0.193681	0.441166	0
1	0.085977	0.104100	0.084520	0.123948	0.180413	0.163605	0.121590	0.135846	0
2	0.114804	0.075066	0.070262	0.089976	0.099025	0.084266	0.161153	0.305447	0
3	0.092744	0.082229	0.086221	0.108802	0.134967	0.164811	0.110780	0.219446	0
4	0.058400	0.072453	0.066846	0.060899	0.076832	0.119684	0.200035	0.344850	0

그림 5 감정분류 모델에 들어갈 데이터

```
kfold = StratifiedKFold(n_splits=3)
dtree=LGBMClassifier()

parameters = {'learning_rate':[0.01,0.03,0.05,0.07,0.09],'max_depth':[5,7,9,11,13,15],'num_leaves':[5,6,7,8,9,10],
              'objective':['binary'],'boosting_type':['gbdt', 'dart', 'goss']}
gcv = GridSearchCV(dtree, param_grid=parameters, cv=kfold,scoring='accuracy', refit=True) #  scoring = f1
gcv.fit(X_train, y_train)
print('final params', gcv.best_params_)   # 최적의 파라미터 값 출력
print('best score', gcv.best_score_)      # 최고의 점수


model=gcv.best_estimator_  # 최적의 파라미터로 모델 생성
y_pred=model.predict(X_test)
accuracy_score(y_test, y_pred)
```

그림 6 하이퍼 파라미터 최적화 및 정확도 측정

Kfold 교차검증을 이용하여 모델의 신뢰성을 높였고, Grid search를 이용하여 모델의 성능을 높였다. 그 결과 최적의 파라미터와 정확도는 아래 그림과 같다.

```
final params {'boosting_type': 'gbdt', 'learning_rate': 0.09, 'max_depth': 5, 'num_leaves': 6, 'objective': 'binary'}
best score 0.7452795261014439
```

그림 7 최적 파라미터와 정확도

4) 감정에 따른 명화 추천

Happy, sad에 대한 비율을 input으로 유클리디안 거리 계산법으로 SinGAN에서 학습할 명화를 선택한다. 사용자가 넣을 이미지를 흑백이라 가정하고 진행하기 때문에 학습시킬 명화도 흑백으로 학습시킬 것이다. 그렇기에 색을 제외한 붓터치나 그림에 쓰인 소재 등이 주는 분위기 위주로 9개의 그림을 선택하였다.

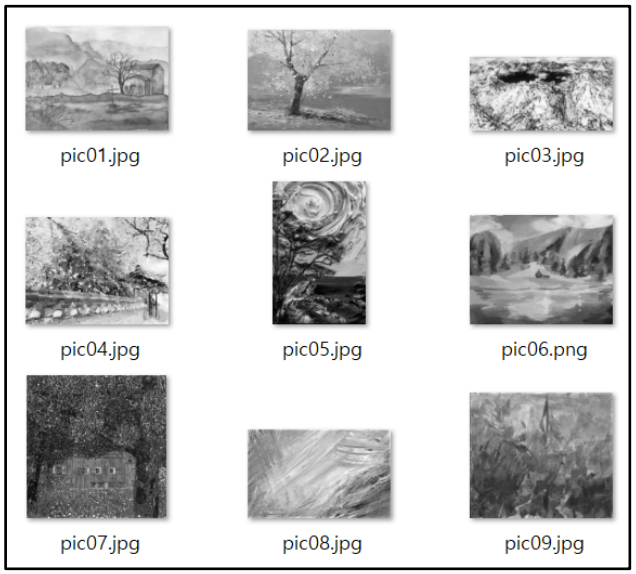


그림 6~14 그림생성모델에 사용될 9개의 그림

9개의 명화에 대한 사람들이 느끼는 감정 비율을 사전 설문을 진행하여 통계를 냈다. 진행된 설문에 대한 이미지 별 감정비율의 결과는 아래와 같다.

	A	B	C
1	pic_num	happy	sad
2	1	0.41	0.59
3	2	0.71	0.29
4	3	0.31	0.69
5	4	0.75	0.25
6	5	0.4	0.6
7	6	0.44	0.56
8	7	0.45	0.55
9	8	0.2	0.8
10	9	0.24	0.76

그림 15 그림당 happy와 sad의 감정비율(1~9에 해당되는 그림은 그림 6의 순서와 동일)

위의 csv파일을 기반으로 sklearn의 euclidean_distances를 이용하여 간단하게 거리를 계산하였다.

```
# 유클리디안 거리 계산
def l1_normalize(v):
    norm = np.sum(v)
    return v / norm

df_norm_l1 = l1_normalize(df[["happy","sad"]])
dist_cal = euclidean_distances(input_ratio, df_norm_l1[:])
df['dist'] = pd.DataFrame(dist_cal.reshape(9,1))

df_sorted = df.sort_values(by='dist' ,ascending=True)  #거리 가까운 순으로 정렬
```

	pic_num	happy	sad	dist
7	8	0.20	0.80	0.645311
8	9	0.24	0.76	0.650328
2	3	0.31	0.69	0.659624
4	5	0.40	0.60	0.672501
0	1	0.41	0.59	0.673994

그림 16,17 유클리디안 거리 계산 코드와 감정비율이 (happy,sad)=(0.22,0.78)이라 가정했을 때 해당 결과

유클리디안 거리 계산을 통해 가장 가까운 그림이

다음 이미지 생성 모델에 학습시킬 input으로 들어간다.

5) 감정에 따른 새로운 그림 생성

새로운 이미지 생성을 위해 GAN(Generative Adversarial Network)를 사용하였다. GAN은 Generator(생성자)모델로 새로운 가짜 그림을 만들어내면 Discriminator(구분자)모델이 Real Data인지 Fake Data인지 잡아내면서 이 두가지 모델을 가지고 경쟁적으로 학습시키는 것을 말한다. 우리는 이 중에서도 SinGAN(1)이라는 모델을 사용하였다. SinGAN의 가장 큰 특징은 Single image만으로 학습시킨다는 것이다. SinGAN의 전체적인 구조는 아래와 같다.

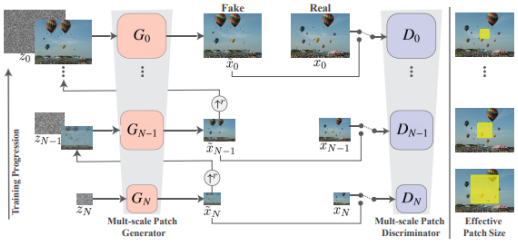


그림 18 SinGAN의 전체적이 구조

4)에서 추천된 그림을 SinGAN에 학습시킨 모델에 사용자가 넣은 이미지를 input으로 Paint to image를 사용한다. Paint to image는 SinGAN의 여러 application 중 하나인 style transfer이다.

6) 결과 예시

① 사용자 감정:

행복하다고 느낌(happy:80%, sad:20%)

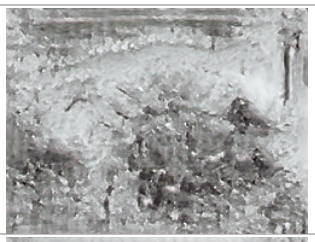
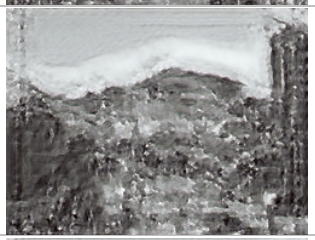
② 추천된 그림:

Pic04

③ 사용자가 바꾸고자 하는 이미지:

④ 감정이 반영된 이미지 결과물:

Start_scale	Output

1	
2	
3	
4	
5	
6	
7	

3. 결 론

본 프로젝트를 통해서 생체 신호를 기반으로 사용자의 감정을 추출한 후, 사용자의 이미지에 사용자의 감정에 따른 STYLE로 변형시켜 새로운 그림을 만들어 냈다.

직접 뇌파를 측정하여 감정 별 분류모델을 만들어 실제 감정변화에 따른 뇌파변화를 확인할 수 있었다.

본 프로젝트에서는 2가지 감정, 9개의 그림만을 사용하였지만, 차후 더 많은 감정을 추출하고, 더 많은 그림을 학습시킨다면 현 시점을 시각적으로 포착하여 유지시킨다는 사진의 보편적인 의미를 넘어서 시각적인 것뿐만 아니라 현 시점의 감정도 함께 포착하여 유지시킬 수 있다는 것에 본 프로젝트의 의의가 있다고 본다.

참고 문헌

1) Tamar Rott Shaham, Tali Dekel, Tomer Michaeli "SinGAN: Learning a Generative Model from a Single Natural Image" ICCV 2019