코미디 빅리그 코너 분석

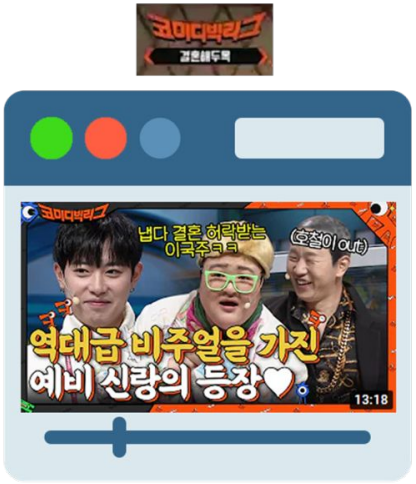
미디어 데이터 분석 및 AI 모델링

AI 수도권4반 13조

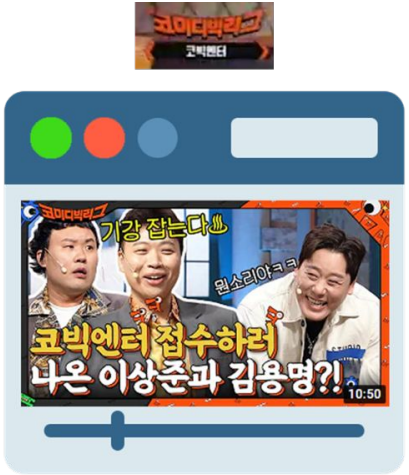
출처1 : https://www.entermedia.co.kr/news/articleView.html?idxno=20235
출처2 : https://www.hankookilbo.com/News/Read/201910060913744548

원본 데이터의 분석과 클래스 분류

두분 사망토론



결혼해두목



코빅엔터

사이코러스

-> 코미디 빅리그 로고 아래 해당 코너명이 있어서 클래스 분류의 기준이 됨

전처리 1_사이즈

-> 코너명은 동일한 위치에 존재하므로 코너명만 잘라서 이미지 속 규칙을 찾아냄

전처리 1_사이즈

전처리 2_패턴추출

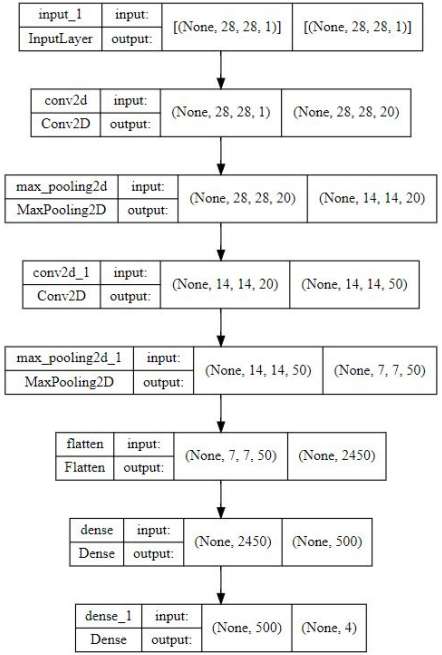
-> 글자라는 패턴을 중점으로 흑백사진 채널에 Threshold + Contour 연산을 통해 이미지의 특징을 강조(추출)

전처리 1_사이즈 | 전처리 2_패턴추출 | 전처리 3_Canny

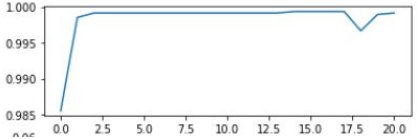
-> 전처리2와 유사한 방법이나, Threshold/Contour없이 Canny연산을 통하여 윤곽선을 뚜렷하게 강조

영훈 에이블러님 모델 1 비교 및 요약【구조】

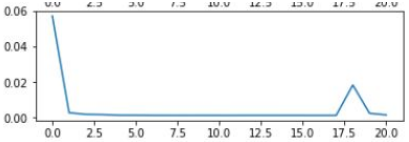
lenet



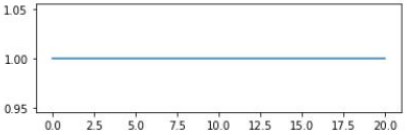
accuracy



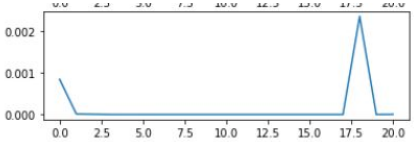
loss



val_accurac



val_loss



이미지 처리

전처리기법1 + threshold(cv2.THRESH_BINARY) + Contour

모델학습

LeNet사용

이미지 처리 이유

1) 좌상단의 코너명 및 부제목의 경우 노이즈가 전혀 없는 인위적이며 공통적인 데이터기에 학습에 사용하기 가장 안정적이라 생각.
2) 해당 부분만을 자르고 패턴특징을 좀 더 증폭시키며 명확한 결과를 얻기 위해 Contour기법 사용.

관련 자료 : https://d2.naver.com/helloworld/8344782

영훈 에이블러님 모델 **1 test** 분류 결과

LeNet

```
import numpy as np
answer = ['1043:1552', '0128:1037', '1534:2357', '2359:2945']
for [index, a] in enumerate(answer):
  start, end = a.split(':')
  image_list = np.linspace(int(start), int(end), int(end) - int(start) + 1)
  print(index+1)
  cor = 0
  wro = 0
  for i in image_list:
    i = int(i)
    temp = cv2.imread(TEST_PATH + '/image/220320' + str(i).zfill(6)+'.jpg', cv2.IMREAD_GRAYSCALE)
    temp = cv2.resize(temp, (HEIGHT,WIDTH))
    # print(model.predict(temp.reshape(1,HEIGHT,WIDTH,CHANNEL)))
    pre = model.predict(temp.reshape(1,HEIGHT,WIDTH,CHANNEL))[0].argmax(axis=0)
    if pre == index:
      cor += 1
    else: wro +=1
  print(cor/(cor+wro) * 100)
```

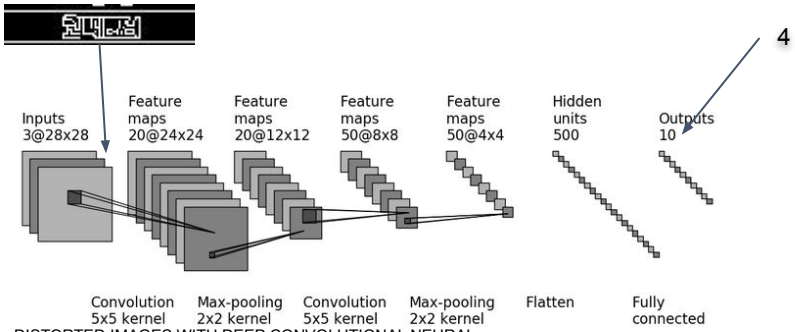
```
1
100.0
2
99.8901098901099
3
97.33009708737865
4
99.82964224872232
```

테스트데이터의 예측은 클래스4개 들어오면 시각적으로 분별이 힘듦.

모델 정확도 확인차원에서 초단위로 답을 제시하고 정확도를 테스트함.
ex)1043 = 17(분)*60 + 23(초)

각 프레임을 1초단위로 저장했기에 맞춘개수 / 전체개수 * 100으로 정확도 구현.
각 코너를 해당 코너로 예측하는 것은 정확함을 의미(acc)

영훈 에이블러닝 모델 2 전처리1 + Canny 적용후 LeNet모델을 사용한 학습



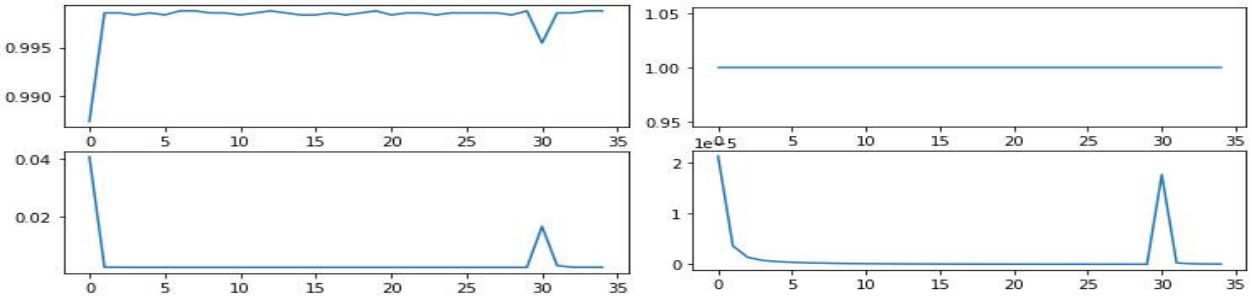
DISTORTED IMAGES WITH DEEP CONVOLUTIONAL NEURAL NETWORKS, Page 2-2
, Yiren Zhou, Sibo Song, Ngai-Man Cheung

검증

```
import numpy as np

answer = ['1043:1552', '0128:1037', '1534:2357', '2359:2945']
for [index, a] in enumerate(answer):
  start, end = a.split(':')
  image_list = np.linspace(int(start), int(end), int(end) - int(start) + 1)
  print(index+1)
  cor = 0
  wro = 0
  for i in image_list:
    i = int(i)
    temp = cv2.imread(ROOT_PATH + '/canny_test/image/220320' + str(i).zfill(6)+'.jpg', cv2.IMREAD_GRAYSCALE)
    temp = cv2.resize(temp, (HEIGHT,WIDTH))
    # print(model.predict(temp.reshape(1,HEIGHT,WIDTH,CHANNEL)))
    pre = model.predict(temp.reshape(1,HEIGHT,WIDTH,CHANNEL))[0].argmax(axis=0)
    if pre == index:
      cor += 1
    else: wro +=1
  print(cor/(cor+wro) * 100)
```

```
1
100.0
2
99.8901098901099
3
97.20873786407766
4
99.82964224872232
```

학습 과정(좌:acc/loss, 우:val_acc/val_loss



영훈 에이블러님 모델 1,2 결과

모델 예측 결과

	conner	start_time	end_time
0	enter	00:00:00	00:52:50
1	boss	00:00:06	00:53:00
2	samang	00:00:14	00:53:01
3	psycho	00:17:21	00:25:53

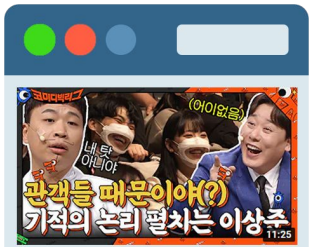
실제 값(정답)

	corner	start_time	end_time
0	enter	00:02:08	00:17:18
1	psycho	00:17:21	00:25:53
2	boss	00:25:57	00:39:17
3	samang	00:39:20	00:49:09

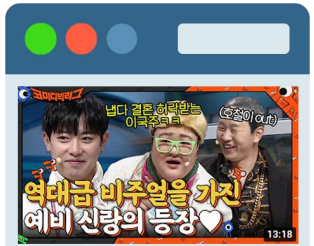
->사이코러스 코너만 정상적으로 예측, 나머지 코너들에 대해 시작 시간과 끝 시간을 예측하지 못함
->4개의 클래스외에 클래스가 없는 부분들이 혼동을 줌

원본 데이터의 분석과 클래스 분류 2

두분 사망토론



결혼해두목



코빅엔터

사이코러스

-> 코미디 빅리그 로고 아래 해당 코너명이 있어서 클래스 분류의 기준이 됨

클래스분류없음

-> 오프닝이나 코너 중간 사이 애니메이션 구간에는 코너명이 없으므로 따로 클래스 분류

전처리 1_사이즈

-> 코너명은 동일한 위치에 존재하므로 코너명만 잘라서 이미지 속 규칙을 찾아냄

전처리 1_사이즈 ×

전처리 2_흑백 처리 ×

+

×

cy

cy

cy

cy

cy

enter

enter

enter

enter

enter

marry

marry

marry

marry

marry

two

two

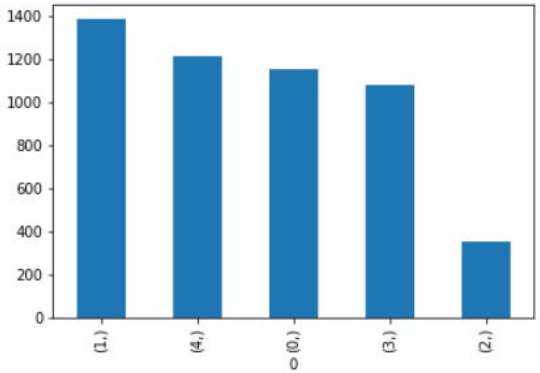
two

two

two

class imbalance

```
1   pd.DataFrame(train_generator.labels).value_counts().plot(kind='bar');
```



각 클래스 별 데이터 개수가 다름

따라서 개수가 적은 class에 더 집중할 수 있도록 class weight 설정

class weight 설정

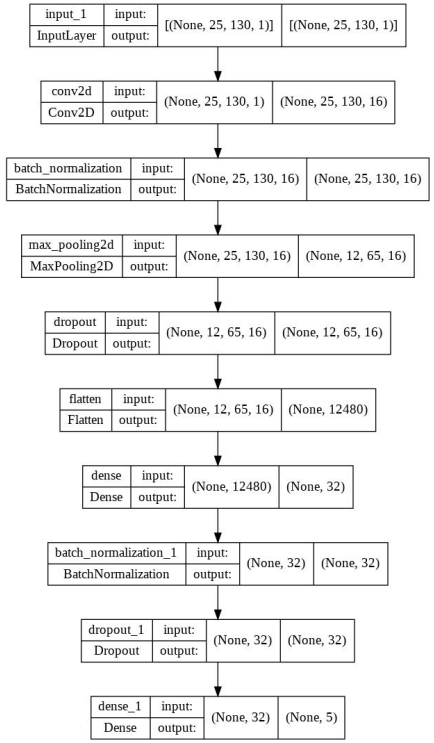
```
# class 불균형 해결하기

from sklearn.utils import class_weight

class_weights = class_weight.compute_class_weight(
                class_weight = 'balanced',
                 classes = np.unique(train_generator.classes),
                 y = train_generator.classes)
class_weights = dict(zip(np.unique(train_generator.classes), class_weights))
class_weights
```

```
{0: 0.8993918331885317,
 1: 0.7479768786127168,
 2: 2.974712643678161,
 3: 0.9558633425669437,
 4: 0.8555371900826446}
```

```
# 모델 학습
history = model.fit(train_generator,
                    verbose=1,
                    callbacks=[es, cp],
                    epochs=1000,
                    validation_data=validation_generator,
                    class_weight = class_weights)
```

-> class weight 설정으로 상대적으로 적은 class 2의 가중치는 2.97,
가장 많은 class 1의 가중치를 0.74로 설정해서 모든 클래스를 찾을 수 있도록 함

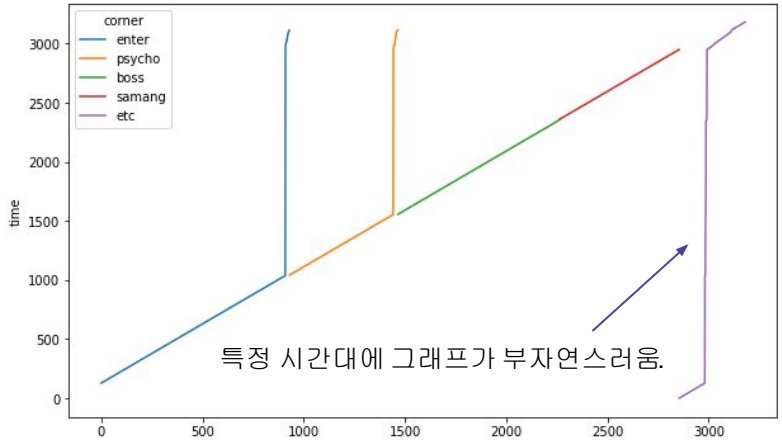
custom model



지현 에이블러닝 모델_**CNN** 모델 결과

```
enter = predict_v2.loc[predict_v2['corner']=='enter']
psycho = predict_v2.loc[predict_v2['corner']=='psycho']
boss = predict_v2.loc[predict_v2['corner']=='boss']
samang = predict_v2.loc[predict_v2['corner']=='samang']
etc = predict_v2.loc[predict_v2['corner']=='etc']

corner_list = [enter, psycho, boss, samang, etc]
temp = pd.concat(corner_list, ignore_index=True)
```

```
fig = plt.figure(figsize=(10, 6))
sns.lineplot(data=temp, x=temp.index, y='time', hue='corner')
plt.show()
```



결과

	conner	start_time	end_time
0	etc	00:00:00	00:53:01
1	enter	00:02:08	00:51:52
2	psycho	00:17:21	00:51:54
3	boss	00:25:56	00:39:16
4	samang	00:39:19	00:49:09

	corner	start_time	end_time
0	etc	00:00:00	00:53:01
1	enter	00:02:08	00:17:17
2	psycho	00:17:21	00:25:53
3	boss	00:25:56	00:39:16
4	samang	00:39:19	00:49:09

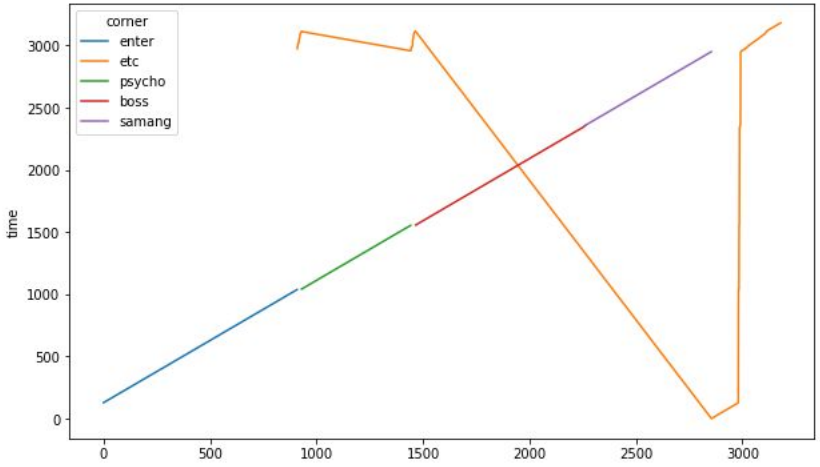
	corner	start_time	end_time
0	etc	00:00:00	00:53:01
1	enter	00:02:08	00:17:18
2	psycho	00:17:21	00:25:53
3	boss	00:25:57	00:39:17
4	samang	00:39:20	00:49:09

-> 4개의 코너 + 코너가 아닌 부분까지 총 5개의 범주로 분류하여 구간을 예측.

해결방안

```
# 코너 시작 시간에서 20분 이상이 경과한 구간은 etc로 한다
for i in corner_list:
    if str(i) != 'etc': # 기타는 제외
        i.loc[i['time'] > i['time'].min() + 1200, 'corner'] = 'etc'

temp = pd.concat(corner_list, ignore_index=True)
```

```
fig = plt.figure(figsize=(10, 6))
sns.lineplot(data=temp, x=temp.index, y='time', hue='corner')
plt.show()
```



각 코너는 최대 20분을 넘기지 않는 것을 train 데이터를 보고 파악함.

코너 시작 시간 20분 이상 경과 시 etc로 구분함.

그래프가 자연스러워짐.

감사합니다!