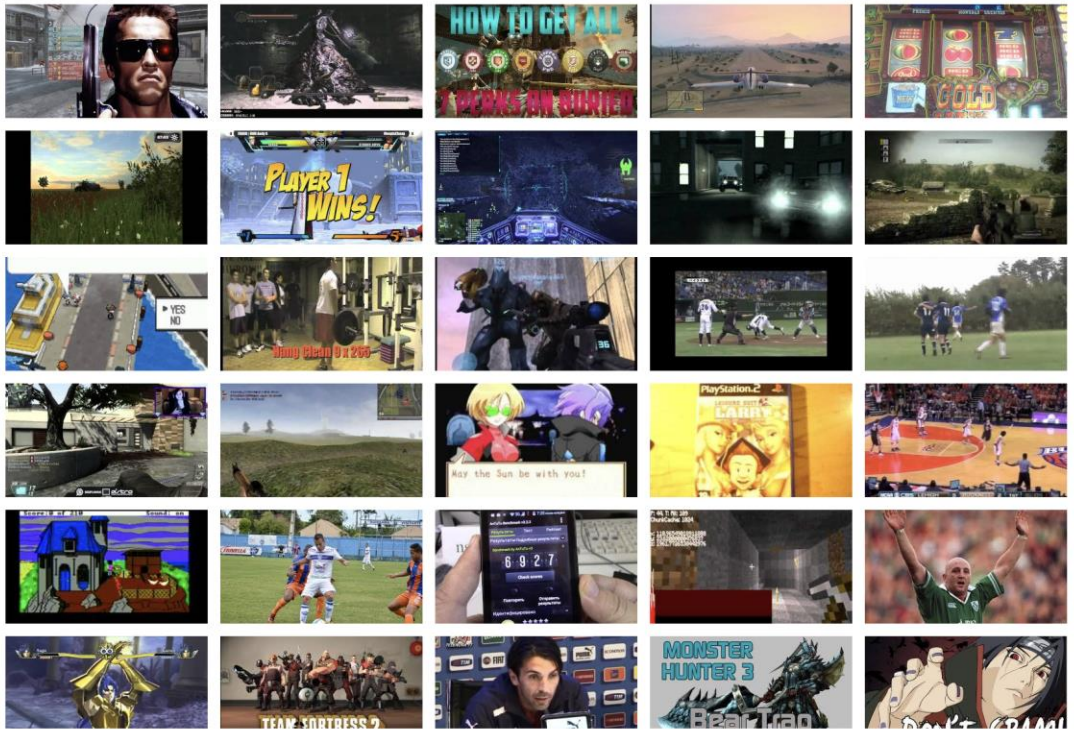
YouTube Data를 이용한 썸네일 추천

CodeStates AI 18 류예린

목 차

썸네일 추천 서비스란



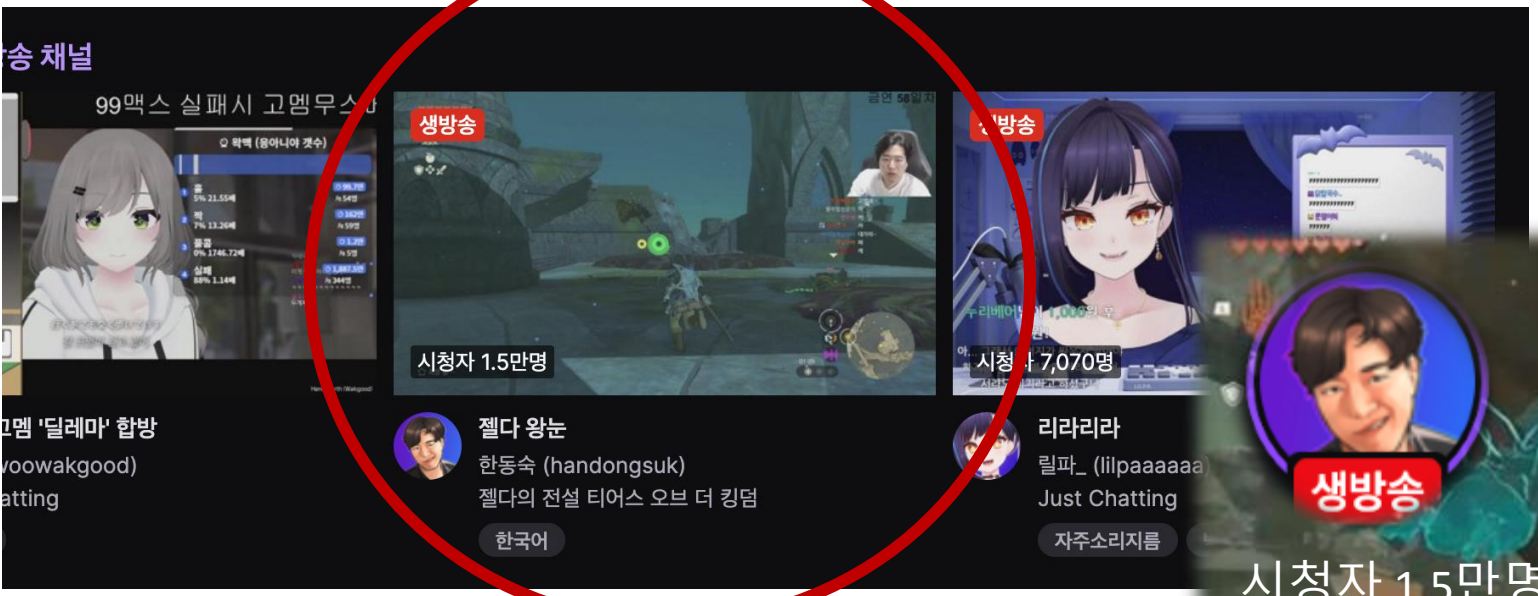
목표: 유튜브의 대량의 썸네일을 학습 한 후,
썸네일 스타일 추천

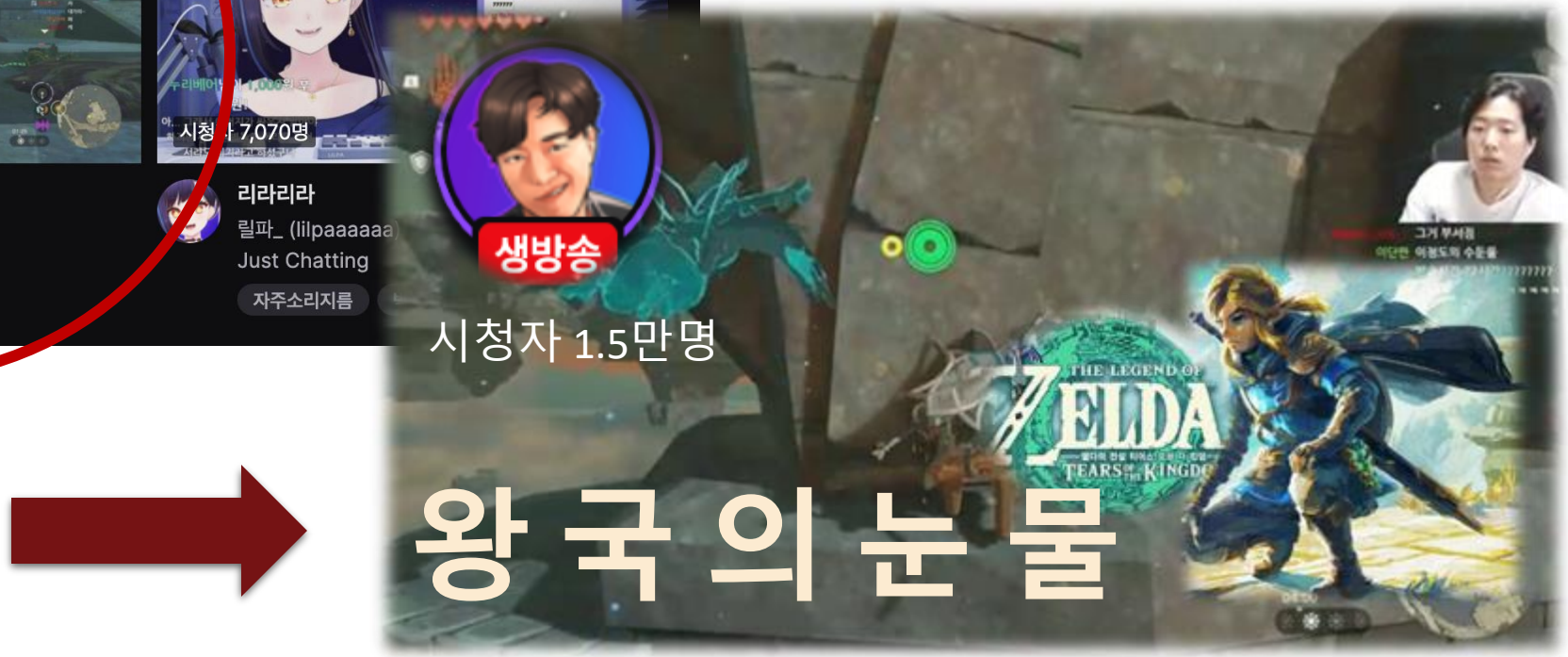
학습한 대량의 썸네일을 바탕으로,
<u>동영상 제작자에게 썸네일 스타일 자동 추천</u>

-> 이는 초심자에게 알맞은 썸네일 스타일을 추천해 줄 수 있음
-> 썸네일 제작의 시간을 줄여 편리함을 제공해 줌
-> 스트리밍 사이트의 경우, 스트리밍 중
썸네일을 자동으로 제작해주는 서비스를 제공할 수 있음

1.자동 추천
2.기존 콘텐츠 기반 추천
3. 개인화 추천

썸네일 추천 서비스란


송 채널
99맥스 실패시 고멤무스
생방송
시청자 1.5만명
젤다 왕눈
한동숙 (handongsuk)
젤다의 전설 티어스 오브 더 킹덤
한국어
생방송
시청자 7,070명
리라리라
릴파_ (lilpaaaaaa
Just Chatting
자주소리지름
고멤 '딜레마' 합방
atting


생방송
시청자 1.5만명
왕 국 의 눈 물

파이프라인 설명

1. 데이터 로드 및 전처리
2. 모델 학습: AutoEncoder
3. 잠재 벡터 추출(모델, 프레임)
4. 벡터간 유사성 계산(유클리드)
5. 유사도가 높은 k개의 프레임 추출
6. 썸네일 추천

전처리

1. Resize
2. Normalize(/ 255.)
3. Expand dim

```
# 이미지 열기
with Image.open(image_path) as image:
    # 이미지 크기 조정
    resized_image = image.resize((32, 32))

    # 컬러 이미지를 흑백으로 변환
    # if grayscale:
        # resized_image = resized_image.convert("L")

    # 이미지 전처리
    normalized_image = np.array(resized_image) / 255.0

    # 차원 조정
    input_image = np.expand_dims(normalized_image, axis=0)
```

```
[8]  Ent_images.shape

     (3329, 1, 32, 32, 3)
```

```
[9]  Ent_images = Ent_images.reshape(3329, 32, 32, 3)
```

시간 관계상 약 3000개의 데이터 훈련

만약 사용자 맞춤 서비스를 제공한다고 했을 때,
사용자 데이터가 몇 개 없는 경우
적은 데이터로도 훈련이 잘 되는 모델을 만들기 위해
이미지 증강 등의 추가적인 전처리 기법이 필요할 것

모델 훈련 과정: AutoEncoder

기본적인 AutoEncoder 구조 사용

```
def create_AE():
    input_img = Input(shape=(32, 32, 3))

    channels = 2
    x = input_img
    for i in range(4):
        channels *= 2
        input1 = Conv2D(channels, (3,3), activation='relu', padding='same')(x)
        input2 = Conv2D(channels, (2,2), activation='relu', padding='same')(x)
        x = Concatenate()([input1, input2])
        x = MaxPooling2D((2,2), padding='same')(x)

    x = Dense(channels)(x)

    for i in range(4):
        x = Conv2D(channels,(3,3), activation='relu', padding='same')(x)
        x = UpSampling2D((2,2))(x)

        channels //= 2
    decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='sa

    autoencoder = Model(input_img, decoded)
    autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
    return autoencoder
```

AutoEncoder는 실제 데이터를 잘 반영하는 모델
따라서, 최대한 사용자의 니즈에 customizing하고
'비슷한' 형태를 추출하여 계산하기 위해 사용

GAN의 경우 생성된 이미지가 현실세계와 거리가 멀 수 있고,
시간이 오래걸려 비효율적이라고 판단

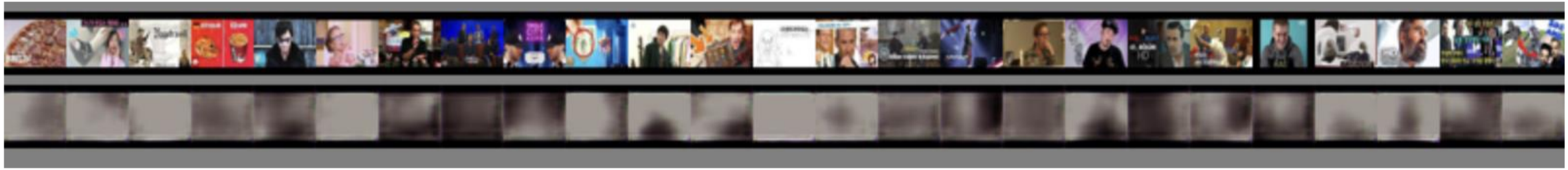
사용될 영상의 프레임들과 유사성을 비교하기 위해
encoder의 각 이미지 잠재 벡터를 추출할 것임

```
Epoch 20000/20000
27/27 [==============================] - 0s 13ms/step - loss: 0.4884
<keras.callbacks.History at 0x7ff3e02b77f0>
```

모델 훈련 과정: AutoEncoder

모델 훈련 결과 (1)

```
Epoch 20000/20000
27/27 [==============================] - 0s 13ms/step - loss: 0.4884
<keras.callbacks.History at 0x7ff3e02b77f0>
```



Epoch: 20000, Loss: 0.4884

흐릿하긴 하지만 형태 정도는 잘 학습한 모습을 보임
그러나 구체적인 스타일을 학습하기에는 아직 미흡

모델 훈련 과정: AutoEncoder

모델 훈련 결과 (2)

```
Epoch 20000/20000
27/27 [==============================] - 0s 11ms/step - loss: 0.4772
<keras.callbacks.History at 0x7ff36a1878b0>
```

Epoch: 40000, Loss: 0.4884

0.01%의 성능 향상
형태를 조금 더 자세히 학습했을 뿐이었고, 더 많은 학습이 필요한 상태
우선 유클리드 거리 계산을 진행하였음

모델 훈련 과정: 유클리디안 거리 계산

Latent Vector

```
[51] encoder_flat = latent_vector.reshape((latent_vector.shape[0], -1))
     frames_flat = frame_latent_vector.reshape((frame_latent_vector.shape[0], -1))

     # Calculate the Euclidean distance between each pair of images
     distances = np.linalg.norm(encoder_flat[:, np.newaxis] - frames_flat, axis=2)

     # Print the shape of the distances array
     print(distances.shape)
```

```
(3329, 8)
```

```
[55] min_distance_indices = np.where(distances == np.min(distances))
```

```
[56] min_distance_indices
```

```
(array([432]), array([3]))
```

딥러닝 모델을 사용하여 거리를
계산하는 이유는
보다 정교한 추천을
수행할 수 있기 때문

그러나 작은 규모의 데이터셋이나
간단한 추천을 위해서는
유클리드 거리를 사용하는 방법도
충분히 효율적일 수 있음

모델에 학습된 index 432 이미지와
테스트에 사용된 index 3 프레임이
가장 유사한 이미지

결 과

왼쪽 이미지를 기반으로 오른쪽 이미지 추천

추천된 프레임 원본

한계점 및 추후 발전 방향

정확도가 많이 떨어졌습니다.

- 학습 시간의 부족
- 테스트 데이터(프레임)의 부족
- 유튜브 영상 이미지를 정사각형으로 resizing함

이는 추후 프로젝트를 이어 진행하며 보완할 수 있을 것 같습니다.

-> 학습을 계속 진행하여 Loss을 감소시킨다. (20000 Epoch당 0.01정도 줄었음)
-> 테스트 데이터를 확보한다. (RAM, 저작권 문제 등 해결)
-> resizing을 영상 비율에 맞게 다시 한다.

한계점 및 추후 발전 방향

1 더욱 세분화 된 카테고리 학습 및 사용자 데이터 수집

2 영상 내용 관련한 관련 이미지 학습(레이블링)

3 자연어 처리를 이용한 알맞은 타이틀 & 타이틀 데이터

4 효율성을 위해 상황에 맞는 기술 적절히 사용

Thank you for listening!