2021111470 이상재

1. 수업시간에 배운 Canny Edge 추출을 본인 얼굴이 나온 이미지에 적용하세요(1.py)

이미지에서 경계를 추출하는 Canny Edge Detection 알고리즘을 이용하여, 본인의 얼굴이 포함된 이미지를 처리하는 것이다. OpenCV의 cv2.Canny() 함수를 사용하여 흑백 이미지로부터 경계를 검출하고, matplotlib을 통해 시각적으로 결과를 비교하여 확인하였다.

cv2.imread()로 이미지를 불러오고 cv2.IMREAD\_GRAYSCALE로 흑백 변환

cv2.Canny()를 통해 edge를 추출.

임계값은 100~200으로 설정하여 일반적인 edge 강조

matplotlib을 통해 입력 이미지와 결과 edge 이미지를 나란히 출력

* 왼쪽: 원본 이미지 (흑백)
* 오른쪽: Canny 엣지 결과

눈, 코, 윤곽선 등 경계가 뚜렷한 부위에서 선명한 에지 검출이 되었다.

흑백, 의류, 야외, 해변이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. 성능 및 평가

* 알고리즘은 전통적인 방식으로 간단하고 빠르게 작동함
* 본 과제에서는 시각적으로 잘 구분되는 edge 결과를 얻었음
* 단, 배경이나 바닥 등이 노이즈처럼 검출됨.

1. 결론

Canny 알고리즘은 CNN 같은 학습 기반 접근 없이도 유용한 edge 검출 수단이다. 단순하지만 효과적인 preprocessing 도구로써 후속 모델 입력에 유용하게 사용될 수 있다. 특히나 바닥, 배경들의 edge도 의미가 있는 경우에 이용가능 해 보인다.

2. Holistically-Nested Edge Detection[CVPR 2015]논문에 나온 Edge 추출 딥러닝

모델을 활용하여 위에 활용된 본인 얼굴이 나온 이미지에 적용하세요.(2.py)

### 과제 개요

본 과제는 2015년 CVPR 논문인 Holistically-Nested Edge Detection (HED) 모델을 활용하여, 주어진 입력 이미지에서 학습 기반 edge map을 추출하는 작업이다.

### 코드 설명

|  |  |
| --- | --- |
| 모델 불러오기 | cv.dnn.readNetFromCaffe()로 prototxt + caffemodel 사용 |
| 전처리 | blobFromImage()로 입력 이미지 전처리 (VGG 기반 mean 값 적용) |
| 추론 | net.forward()로 edge map 추출 |
| 후처리 | 결과를 원래 이미지 크기로 resize 및 8-bit 정규화 |
| 출력 | OpenCV로 원본 이미지와 edge map을 시각화 |

모노크롬, 예술, 흑백, 흑백 사진이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

### 결과 이미지 해석

* 딥러닝 기반 HED가 추론한 edge map

눈썹, 얼굴 외곽, 머리카락 등 의미있는 고해상도 edge가 잘 표현되었고, Canny보다 배경, 바닥등의 잡음이 적음

### 결론

HED 모델은 깊은 네트워크 구조를 이용하여 다중 스케일의 피쳐를 조합해 고품질 edge map을 생성한다.

전통적인 Canny 대비 배경 잡음에 강하고 의미 있는 구조만 추출하는 데에 탁월하다.

3.  입력: 이미지, 출력: Canny Edge 결과가 나오는 딥러닝 모델을 설계하고

학습하세요(60점)(HED데이터셋 사용, 3.py, evaluate.py)

### 과제 개요

본 과제는 BSDS500 데이터셋을 기반으로, 각 이미지에 대해 Canny Edge 결과를 정답(GT)으로 사용하여 딥러닝 모델을 학습하는 것이다.

즉, 기존의 수작업 GT가 아닌, cv2.Canny() 알고리즘으로 생성된 엣지맵을 예측하도록 모델을 학습시킨다.

* 1. 학습 설정
* 손실 함수: BCELoss() (이진 분류 문제)
* 옵티마이저: Adam, learning rate = 1e-4
* 에폭: 10
* 저장: torch.save(model.state\_dict(), "model.pth")

for epoch in range(10):

model.train()

...

print(f"Epoch {epoch+1}, Train Loss: {train\_loss:.4f}, Val Loss: {val\_loss:.4f}")

#### Loss 시각화:

loss\_curve.png 생성됨

→ Train/Val loss가 점차 수렴, 학습이 안정적으로 진행됨

### 모델 평가

모델이 예측한 결과와 두 종류의 GT를 비교함:

| 비교 대상 | Accuracy | IoU | F1 Score |
| --- | --- | --- | --- |
| Canny GT | 0.2052 | 0.2052 | 0.3405 |
| BSDS GT | 0.6680 | 0.6680 | 0.8009 |

* Canny GT 기준으로는 낮은 성능 → 모델이 edge sparsity를 잘 반영 못함
* BSDS GT 기준으로는 높은 성능 → Canny보다 BSDS GT의 분포와 비슷하게 예측된 것
* 모델은 거의 모든 픽셀을 edge로 예측하는 경향을 보임 (pred > 0.3: 65536 픽셀)

### 결과 이미지 시각화텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진 AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

아래 4개 항목을 포함한 비교 이미지가 vis\_full/ 폴더에 저장됨:

* 원본 입력 이미지
* Canny GT
* 모델 예측 결과
* BSDS GT스크린샷, 나무, 파노라마, 야외이(가) 표시된 사진

  AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 곰, 스크린샷, 눈이(가) 표시된 사진

  AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

스크린샷, 엑스레이 필름이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

### 결론 및 개선 방향

* 현재 모델은 Edge vs Background의 불균형한 분포를 잘 학습하지 못하고 있음
* 개선 방안:
  1. Class imbalance 대응: pos\_weight 적용된 BCEWithLogitsLoss 사용
  2. Threshold 튜닝: 평가 시 0.3 → 0.5로 조정 필요
  3. U-Net 확장: Skip connection, 더 깊은 네트워크로 확장