<5조> 발표:??

[데이터]

??

[모델]

1. Yolov8 모델 이용 차량 번호판 탐지
2. 노이즈 평가
3. 기존 데이터셋을 노이즈 적용 데이터셋으로 변경
4. ??

[학습결과]

1. Confusion matrix로 표현

노이즈 추가 전 – 배경을 노이즈로 잘못 분류한 경우 많음

노이즈 추가 후 – 배경을 번호판으로 잘못 분류한 경우 많아졌지만, 정확도는 상승

의의 : 노이즈 추가 후, 번호판을 번호판이라 예측한 경우 많아짐

2. Precision-Recall 그래프

높은 recall에도 precision 안정적이게 됨

3. 이미지

번호판 인식 못해 LicensePlate 누락되던 게 나옴

하지만, 배경을 번호판으로 잘못 인식하는 경우도 나옴.

[성능]

상승..?

[목표]

1. recognition 모델에 대해 노이즈 포함된 데이터셋 학습

2. recognition 모델에 대해 해상도가 낮은 번호판에서도 높은 정확도 제공

3. ??

4. ??

\*조교님 질문

1. 노이즈 만든 과정 더 정확히 설명해달라. 모션 블러 효과 자세히 설명해달라.

A. 노이즈 패턴을 만들어 빗줄기로 적용. 이를 기존 사진에 합성.

2. Confusion Matfix에서 test는 전부 noise 이미지에 대해서만 하나? Train은 어떤가?

A. noise가 없는 것도 있다. Train 역시 noise가 없는 것도 있다.

Q. 그러면 train과 test가 분포가 비슷할 것 같은데, 결과는 반대로 나오는 것으로 보인다. 왜 그런지 생각해봤나?

A. 번호판이 아닌 것도 번호판으로 인식해서 그런 것으로 예상.

Q. Precision-Recall curve에서도 map 옆에 있는 수치가 성능이라고 보면 되나?

A. 그렇다. 0.768에서 0.930으로 상승했다.

3. 베이스라인이 적절한 건지 모르겠다. 보통 데이터셋은 고정해 놓고, 모델의 차이를 보이는 건데 이러면 데이터셋이 개선된 건지, 모델이 개선된 건지 모호하다.

A. 그래서 OCR로 하고자 했으나, 시간 부족으로 베이스라인을 그렇게 선정했다.

<6조> 발표:홍여원

UFC 경기 결과 예측 모델 구현

[모델 구현]

데이터셋 전처리 다시 함.

기존 – 키, 무게, 리치, 스탠스 4가지만 사용

현재 – 사용할 수 있는 선수들 모든 특성으로 확장

결측치 제거, one-hot encoding

체급별 분류

선수를 노드, 경기를 양방향 엣지로 변경 (gcn이 양방향 엣지 사용하는 것 확인)

선수1, 선수2 1:0으로 라벨링

6:2:2

[모델 구현]

GCN convolution – 두 선수의 노드 임베딩을 이용해 엣지를 예측 (0~1)

초기 threshold : 0.5 -> 0.6~0.7 사이 값들이 모두 승리로 예측됨

Threshold 상승시킴

k-fold cross validation 이용해 hyperparameter별 최적의 epoch 조합

결과 : 0.001, 8, 2

[학습 결과]

1. 베이스라인

Light, welter, heavy 3개 체급 사용

지난주와 동일한 4개 모델 사용

키, 무게, 리치, 스탠스 적용 / 모든 feature 사용

데이터셋 분리 6:2:2

평가 – val 정확도&F1, test 정확도&F1

All features의 logisticregression이 가장 높은 성능 보임.

우리 모델은 미치지 않았다.

\*체급마다 달랐음..

[모델 장단점]

상대 전적 고려 학습 가능

경기기록 적은 선수 예측은 힘들고, 시간적 요소 고려X

기존 회귀분석보다 떨어지는 성능

-> 특성을 그룹화하여 노드로 사용 / EvolveGCN 등의 동적 그래프 모델 사용 계획

[데이터 시각화]

특성을 상중하 25% / 50% / 25%로 나누어 그룹화

이는 승률 추이 변화 관찰하기 용이한 값

Weight – red와 blue의 상관관계가 반대

Age – red와 blue의 상관관계가 반대

Reach – 상관관계 안 보임

Wins total – 상관관계 동일

결론 : red 측 승률이 더 높기 때문에, blue 측 기준으로 학습하면 틀린 경우가 많음

[EvolveGCN]

기존 GNN+LSTM은 시간적 변화 반영 위한 것이었으나, 이것이 힘들어 해당 모델 시도.

EvolveGCN-H : hidden state 중점으로 학습

EvolveGCN-O : parameter 중점으로 학습

이 중 H를 사용해 학습할 예정

[질문]

1. 학습할 때 사용하는 loss function이 기본?

A. 그렇다

2. 체급별로 나눈다고 했는데, edge가 없는 경우도 있었나? 그 경우는 제거했나?

A. 있지만 따로 제거하진 않았다.

3. GNN은 주변 정보로부터 정보를 받는 것. 타인의 reach 정보가 나에게 들어오면 예측에 도움이 될지 의문이다.

A. 아직 생각해 보지 않았다.

4. link classification 사용할 땐 train에 비율이 적을 때 결과가 좋은 것으로 알고있다. gnn에서 6:2:2는 여전히 train에 매우 높은 비율. baseline보다 결과가 낮은 문제는 이걸로 해결해보라.

A. 감사합니다.

5. red/blue는 랜덤하게 되면 안되는건가? 누가 어느 색깔인지 알아야 하나? 레이블을 랜덤으로 지정해서 반반으로 만들어 보면 안되나?

A. 랜덤하게 하려 시도 중.

[7조] 발표:지난주와 동일. 박진호

Visual realism 향상 목표

모델 - Dreamgaussian, crm, instantmesh 사용

코드 직접 보면서 분석

GPU 소모량 체크함

[DreamGaussian]

입력 이미지 전처리

1) 배경 없애고, 객체 중앙으로 옮기기

2) iter를 거치며 처음 구에서 객체 모양으로 변화

3) 그래픽 관련 알고리즘 사용해 mesh&coarse texture

4) texture refinement

Inference time 총 94초

총 3000MB 사용

[CRM]

1) 배경 제거

2) 6개의 orthographic 이미지 생성

3) CCM

4) final mesh&texture

78초

21962MB (Nerf 기반이라 훨씬 많이 사용)

[InstantMesh]

LRM : 이미지 인코딩 -> … -> 플렉시 큐브

1) 배경 제거

2) 멀티 뷰 이미지 생성

3) 최종 결과 생성 – 이전 모델들보다 나아 보인다

22초

27186MB

-> 메모리 사용량은 많지만, inference time과 quality 측에서 instantmesh로 결정

[Super-resolution]

저화질 이미지를 고화질로 만들어주는 기술

Instantmesh+superresolution 사용 예정

DRCT(dense residual connected Transformer) : 얕은 이미지에서 얕은 feature 추출, deep feature로 발전시켜 deep 이미지로 변환

IPG : 기존 이미지를 neural net에 태워 classification 통해 선명하게.

[InstantMesh]

1) single image -> Multiview image

2) model initialization (3가지 view로)

Incoder -> decoder -> triplane synthesizer

4개의 MLP로 이루어졌다고 생각하라…

[Fine tuning 구현]

1. update whole LRM – 너무 많은 메모리와 학습시간.. 버림

2. update part of LRM (Transformer) - 1번보단 낫지만 여전히 많은 메모리와 학습시간.. 버림

3. update part of Transformer (layers) – 메모리와 학습시간 감소됨. 채택.

\*일부 레이어만 사용한 것.

GPU 사용량 아끼는 방법들 시도해서 아낌.

[결과]

터미널에 로그 찍힘.

사과 3d 모델링 결과 나옴.

사과는 복잡한 모델 아니므로, 비교 위해 인형과 공장 이미지에 대해서도 시도.

SR 적용 전/후 결과 3d 모델로 비교. 큰 차이 없어보임. (apple)

꼬리와 귀 쪽 잘 살아남. (doll)

결과 많이 좋아짐. (factory)

Fine tuning 결과 동영상으로 확인 : loss가 떨어지는 것 확인(apple, doll). 다만 factory의 경우, noise가 좀 낀다.

Loss graph : epoch 200까지 loss가 떨어진다. 잘 학습되고 있다.

[한계]

1. 3D metric 적용한 평가 : 아직 어려워 2D metric 까지만..

2. 다른 high tech 사용하는데 어려움.

3. ??

[향후계획]

Sketch-to-image 적용 (~12/08)

Evaluation (~12/08)

[질문]

1. fine tuning 어떻게 했는지 다시 설명해달라.

A. 설명함

2. 원래 instantMesh 모델은 어떻게 학습하는가? 다른 방식을 적용한건가, 같은 방식으로 한건가?

A. MSE+LPIPS+reg 적용함

3. fine tuning 할 때 사용한 데이터셋은 pre training될 때 데이터셋과 동일한가?

A. 논문에서 제시한 인형 같은 이미지 계속 사용할 예정.

4. training과 test가 같은 데이터셋을 사용하는 게 맞는가?

A. 그렇다.

5. fine tuning할 때 hyperparameter 변화를 적용한 건지?

A. 해 볼 예정.

6. GPU 사용량 낮게 쓰는 모델을 선택한 과정이 어떻게 보면 당연해 보이는데 필요했나?

A. 사실 있는 라이브러리를 갖다쓴 과정에 불과.

7. 2d로 랜더링해서 평가하는 과정도 실행해본 건가?

A. 실행할 예정.

[전체 comment]

Cross-validation 방식 적용하는 게 좋을 것. 지금은 6조만 한 것으로 보임.