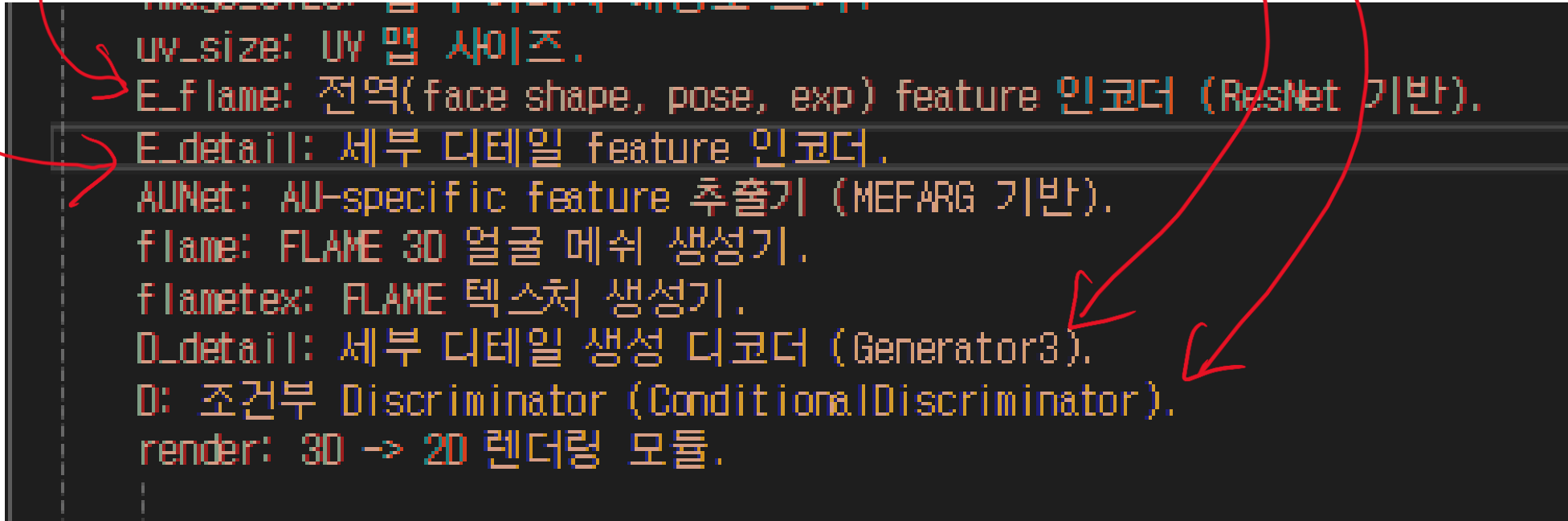


AUCGAN 코드 문제점 현황

20250722

코드의 주요 문제점

1. 코드가 잘 정리되어있지 않음



```
uv_size: UV 맵 사이즈.  
E_flame: 전역(face shape, pose, exp) feature 인코더 (ResNet 기반).  
E_detail: 세부 디테일 feature 인코더.  
AUNet: AU-specific feature 추출기 (MEFARG 기반).  
flame: FLAME 3D 얼굴 메쉬 생성기.  
flametex: FLAME 텍스처 생성기.  
D_detail: 세부 디테일 생성 디코더 (Generator3).  
D: 조건부 Discriminator (ConditionalDiscriminator).  
render: 3D -> 2D 렌더링 모듈.
```

코드의 주요 문제점

ConditionalDiscriminator가 여러 곳에서 반복 정의되어 있으며,
generator는 사용되지 않았음

```
trainer_EM_...gT_aucGAN.py  encoders_enc.py  deca_EM_AU_e...gT_a

97
98 class ConditionalDiscriminator(nn.Module): #torch.nn.Module 상속
99     """
100     설명:
101     Discriminator 모듈. cGAN의 Discriminator 역할 수행.
102     클래스를 상속받음.
103     torch.nn.Module 상속 -> PyTorch에서 학습가능 신경망 모듈로 정의됨.
104
105     attributes:
106     l1: Conv2d 레이어. AU feature vector 관련 전처리나 조건부 feature 위해 선언
107     model: nn.Sequential로 정의된 여러 Conv2d 레이어들로 구성된 모델.
108
109     method
110     __init__: 클래스 생성자. 초기화 테스트. Conv2d 레이어 정의.
111     forward(img, condition):
112     """
113
114     def __init__(self):
115         """
116         클래스 생성자.
117         사실상, attribute에 torch.nn.module 신경망 레이어 정의
```

```
def generator_step(self, cond_img, final_image, losses):
    """
    Generator의 학습 단계를 수행하고 손실을 계산.

    input:
    cond_img (torch.Tensor): 조건 이미지 텐서. [B, 3, H, W]
    final_image (torch.Tensor): Generator가 생성한 이미지.
    losses (dict[str, torch.Tensor]): 이전 단계의 손실값 사전.

    changing attributes:
    self.optimizer_G (torch.optim.Adam): Generator 최적화 수행.

    readonly attributes:
    self.D (ConditionalDiscriminator): Conditional Discriminator 모델.
    self.adversarial_loss (torch.nn.BCELoss): Binary Cross Entropy 손실 함수.

    output:
    G_loss (torch.Tensor): Generator 손실.
    """

    content_loss = losses['photo_detail'] + losses['z_reg'] + losses['z_diff'] + losses['z_sym']
    real_label = 1.0
    fake_label = 0.0
    self.D.zero_grad()
    self.optimizer_D.zero_grad()
    self.deca.E_detail.zero_grad()
    self.deca.D_detail.zero_grad()
```

코드의 주요 문제점

- 2. AU를 Condition으로 넣는 ConditionalDiscriminator가 있으나, 잘 작동하지 않았을 가능성 있음

AU: (B, 27, 512)형태. AU27개 512차원벡터.

```
def forward(self, img, condition):  
    condition = condition.squeeze(0) # batch size 차원 제거 (필요한 경우)  
    condition = condition.view(condition.size(0), 3, 9, 512)  
    condition = F.interpolate(  
        condition, size=(224, 224), mode='bilinear', align_corners=False  
    )  
    d_in = torch.cat((img, condition), 1) # 이미지와 채널 방향으로 concat  
    return self.model(d_in)
```

코드의 주요 문제점

- view: 4D 텐서 변환. 27개 AU를 (3,9) spatial grid로 해석하려는 의도.
- interpoate: AU grid를 2D 공간상에서 이미지 크기 맞춰 업샘플링
- img와 concat

```
def forward(self, img, condition):  
    condition = condition.squeeze(0) # batch size 차원 제거 (필요한 경우)  
    condition = condition.view(condition.size(0), 3, 9, 512)  
    condition = F.interpolate(  
        condition, size=(224, 224), mode='bilinear', align_corners=False  
    )  
    d_in = torch.cat((img, condition), 1) # 이미지와 채널 방향으로 concat  
    return self.model(d_in)
```

코드의 주요 문제점

AU는 AUnet에서 위치를 뽑아서 각각의 AU 위치를 추론한 것.
이미지에서 AU 추론은 되는데,
이후 그 정보를 손실해서 AU -> 이미지 대응이 안되는 상태
27개의 AU feature (batch, 27, 512)로부터
41개의 AU 실제 활성화 여부 (batch, 41)을 추론하게 됨.

코드의 주요 문제점

2.1. AU feature는 공간적 의미가 없음. 얼굴 이미지의 "특정 위치" 에 대응되는 좌표를 보장하지 않음.

AU1 = inner brow raiser, AU12 = lip corner puller...

이미지 "위 특정 위치" 와 연관하나, AU는 1D 정보.

그런데 여기서 view -> interpolate -> concat은 마치 AU가 2D grid인것처럼 취급함.

(B, 27, 512) -> (B, 3,9, 512)로 그냥 27개 AU가 3*9 grid로 배치되었다고 가정.

마치 AU가 2D 이미지 위에 퍼진것처럼 만듦

코드의 주요 문제점

2.2. interpolate: AU feature가 얼굴 공간좌표랑 직접 관계가 없음
에도 bilinear interpolate.

(3,9) -> (224,224) 업샘플링 시, 단순히 bilinear interpolation으로
값 확장. 하지만 AU feature의 위치적 연관성을 만들지 않는다.

코드의 주요 문제점

2.3. 해결점

- A. AU feature를 뽑을 때, AU가 추출되던 CNN feature map을 그대로 활용하면 어떨까? 이 feature map은 이미지 좌표계와 그 대로 정렬되어 있지 않나?
- B. 각 AU 벡터를 얼굴 랜드마크 좌표 기반으로 배치하는 spatial map을 생성하면 어떨까?

코드의 주요 문제점

3. 사용하는 GAN 방식이 2014년 방식

다양한 분야에서 사용하는 Wasserstein Distance 대신
2014년 GoodFellows에서 적용한 loss function을 사용함.
(Jensen-Shannon divergence 기반)

⇒수렴 속도가 2~5배 정도 느리고, 학습 초기 불안정성이 너무
큼