24조 코드 보고서

network

`ConditioningAugmention`

클립에 임베딩된 텍스트를 받아 mu와 sigma를 통해 augmentation을 해준다.

`ImageExtractor`

layer들을 통과해서 나온 것을 3*H*W의 이미지 형식으로 바꿔준다.

`Generator_type_1`

1개의 mapping층, 4개의 upsample층을 지나 이미지를 출력한다. condition과 noise를 입력받아 mapping을 지나며 upsample층에 들어갈 수 있게 shape을 바꾸고 upsample층을 지나며 채널은 2배씩 작아지고 크기는 2배씩 커진다. Generator_type_1을 통해 (N_g/16)*64*64 차원을 가지는 output과 3*64*64의 이미지가 만들어진다.

`Generator_type_2`

1개의 joining층, n개의 res층(default=2), 1개의 upsample층을 지나 이미지를 출력한다. 입력변수는 앞에서 augumentation된 condition과 바로 전 generator에서 나온 output을 입력받는다. joining층과 res층에서는 크기는 변화시키지 않고 채널수만 바꿔준 후 upsample층에서 크기는 2배로 채널 수는 반으로 바꿔준다. 두번째 generator와 세번째 generator는 모두 Generator_type_2를 사용하고 두번째 generator에서는 3*128*128의 이미지를 세번째 generator에는 3*256*256의 이미지를 생성해준다. 이렇게 각 generator를 하나씩 지나며 더 고해상도의 이미지를 얻게 된다.

`UncondDiscriminator`

fake이미지가 real이미지처럼 잘 생성되었는지 파악하고 real이미지와 유사하게 만들도록 도와주는 것이다. 이전 prior층을 나온 (8*N_d)*4*4 차원의 값을 입력 받아 unconddiscriminator로 1*1*1 차원의 output을 출력한다. 이 output은 이미지에 대해 예측된 값으로 나중에 sigmoid를 거치며 확률값으로 변환된다.

`CondDiscriminator`

생성된 이미지가 설명 텍스트와 일치하는지 실제 이미지와 유사한지 등을 판단한다. Conddiscriminator 역시 이전 prior층을 나온 (8*N_d)*4*4 차원의 값을 입력 받고 추가로

mu값도 입력 받아 1*1*1 차원의 output을 출력한다. 이후 나중에 sigmoid를 거치며 확률값으로 변환된다.

`AlignCondDiscriminator`

텍스트와 이미지간 유사성을 판단하고 contrastive learning에 사용된다. Alignconddiscriminator도 input으로 prior층을 나온 나온 (8*N_d)*4*4 차원의 값과 mu값을 입력 받아 1*1*1 차원의 output을 출력한다. 이후 나중에 sigmoid를 거치며 확률값으로 변환된다.

`Discriminator`

global층과 prior층 그리고 세개의 discriminator들을 연결시켜 Discriminator class를 만들어준다. global층은 채널수는 증가시키고 크기는 줄여 3*H*W 차원의 iuput을 (8*N_d)*(H/16)*(W/16) 차원으로 출력해주고 prior층에서 각 discriminator로 들어갈 수 있게 (8*N_d)*4*4 차원으로 바꿔준다. 이후 condition을 입력 받았다면 cond output과 aligncond output을 출력해주고 condition을 입력 받지 않으면 uncond output을 출력해준다.

train

본 train.py 코드의 역할은 로스함수를 구현하고 학습을 진행하는 것이다. 로스함수의 경우 크게 cont_loss_G, con_losss_D, D_loss, G_loss 4개의 함수로 구성되어 있다. Cont_loss_G 함수는 생성자의 contrastive loss를 계산해주는 함수로, 가짜 이미지, 클립모델 그리고 임베딩된 텍스트를 인풋으로 받는다. 사전에 학습된 클립 모델을 통해서 가짜 이미지를 임베딩된 벡터로 전환하고, 이미 임베딩된 텍스트 벡터를 가지고 두 벡터간의 코사인 유사도를 바탕으로 로스함수를 정의한다. 이때 해당 배치 내에 서로 매칭되는 이미지-텍스트 임베딩 벡터간 코사인 유사도는 최대로, 서로 매칭되지 않는 이미지-텍스트 임베딩 벡터간 거리는 최대화하는 방향으로 로스함수는 정의된다. Cont_loss_D 함수는 위 함수와 똑 같은 구조이나, 사전에 임베딩된 이미지 벡터를 g_out_align꼴로 받기 때문에, 클립 모델을 통해 이미지를 임베딩하는 과정이 생략된다. D_loss는 구별자의 총 로스함수를 정의해주는 역할을 하며, binary cross entropy를 통해서 진짜 이미지는 1로, 가짜 이미지는 0으로 예측하도록 훈련된다. 이때 구별자가 자연어에 대한 조건 mu를 받는 경우와 아닌 경우를 모두 계산하며 이를 각각 cond_loss, uncond_loss라 한다. Cont_loss는 위 함수로부터 호출하며 하이퍼파라미터 감마를 곱해주어 사용한다. 이 세 로스를 선형결합한 것이 최종 로스로 호출된다. G_loss는 위 함수와 구조가 유사하나,

생성자에 대한 로스이기 때문에 구별자가 가짜 이미지에 대해 예측한 것이 1이 되도록 binary cross entropy가 정의된다. 생성자 로스 또한 언어조건 mu를 사용한것과 안한 것 그리고 contrastive loss 세개의 선형결합으로 표현된다.

학습 과정은 두가지 함수로 정의된다. Train_step는 스텝별로 batch_optimization을 실시한다. 이때 최적화 기법은 Adam을 사용하며 구별자와 생성자를 차례대로 최적화한다. 로스함수는 사전에 선언한 함수들을 사용한다. Train 함수는 실제로 학습을 진행시키는 함수로, 학습률, 에폭 수 등 각종 하이퍼라미터를 인풋으로 받으며, 사전에 선언한 생성자와 구별자 클래스들을 활용해 각각 선언하고, 사전에 선언한 train step 함수를 이용해 에폭별로 학습을 진행한다.