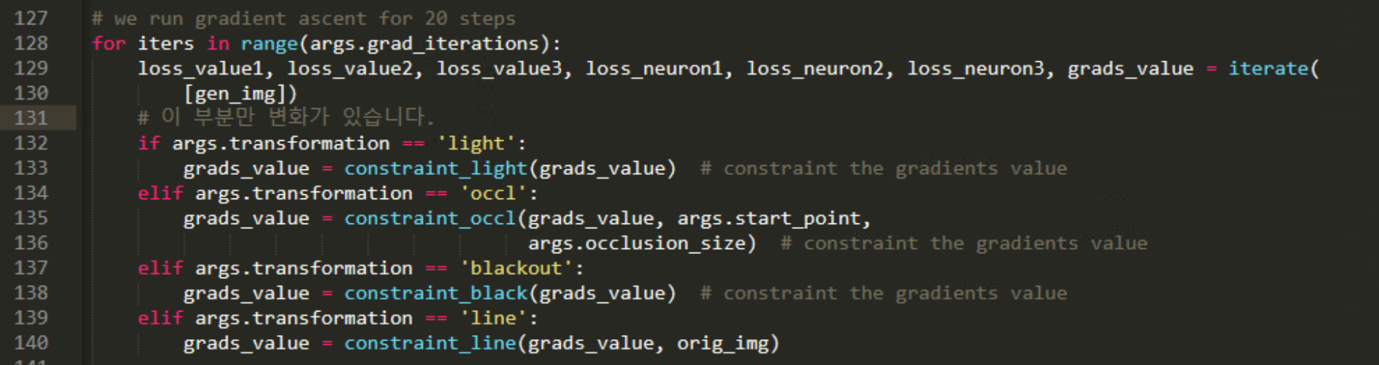
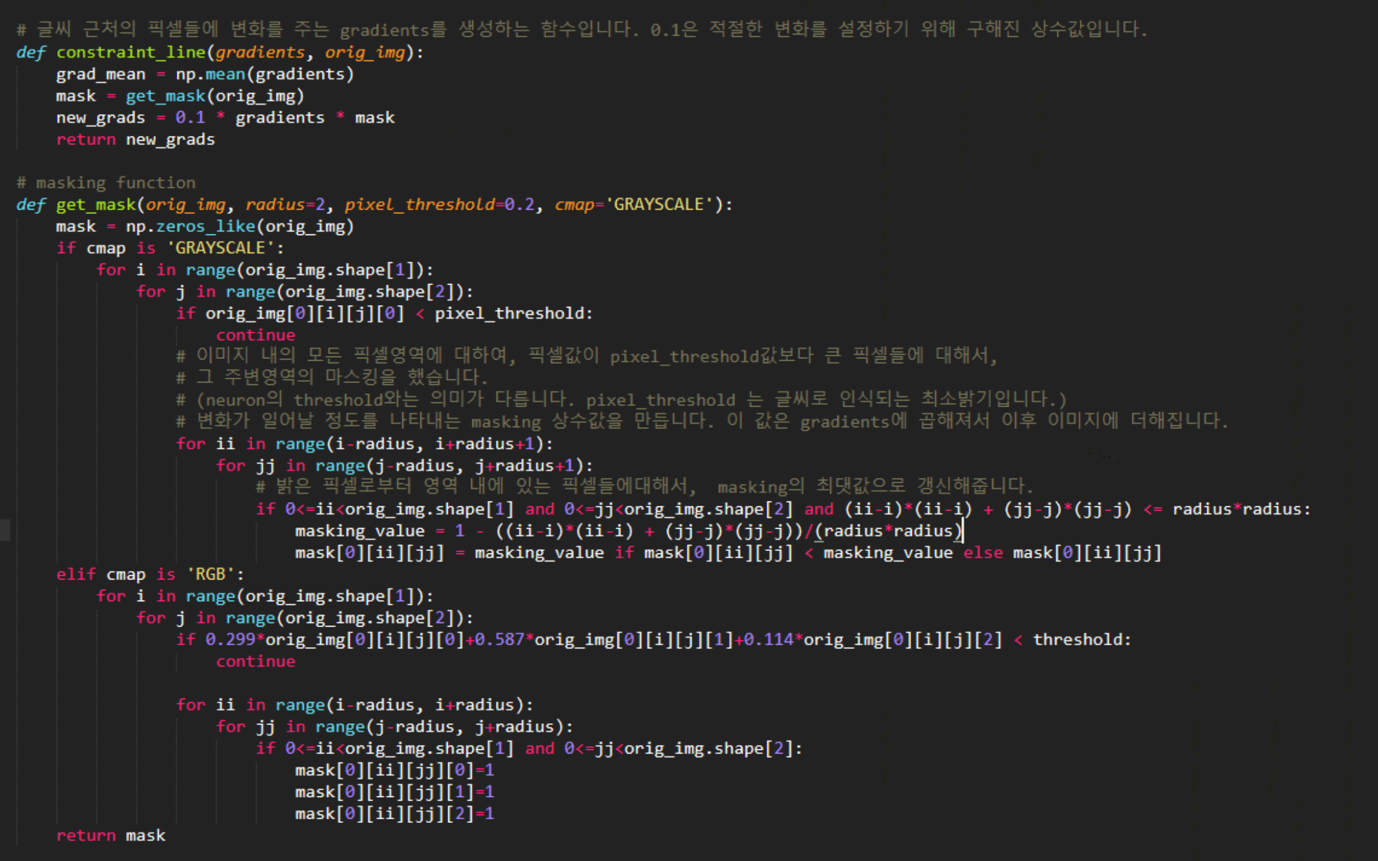
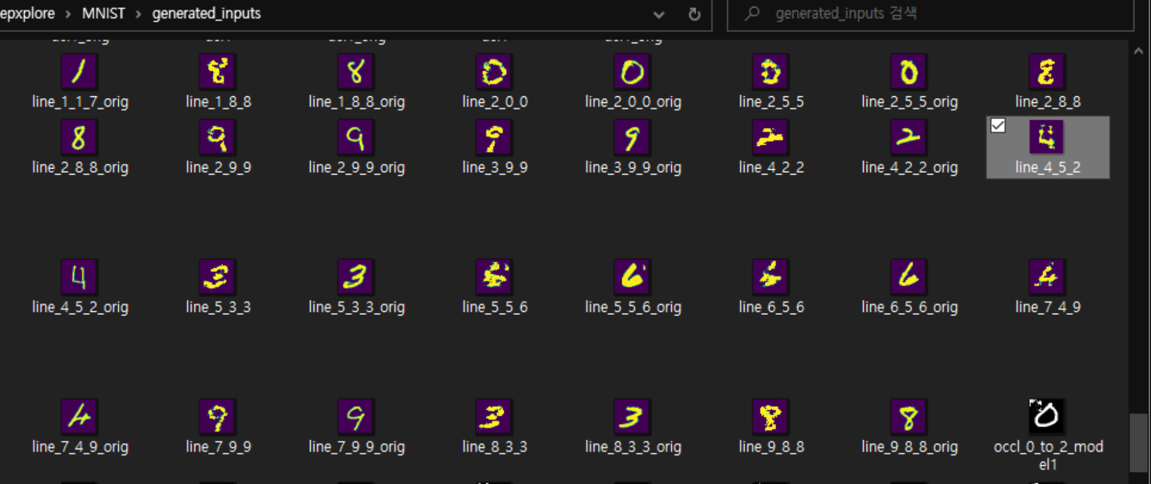
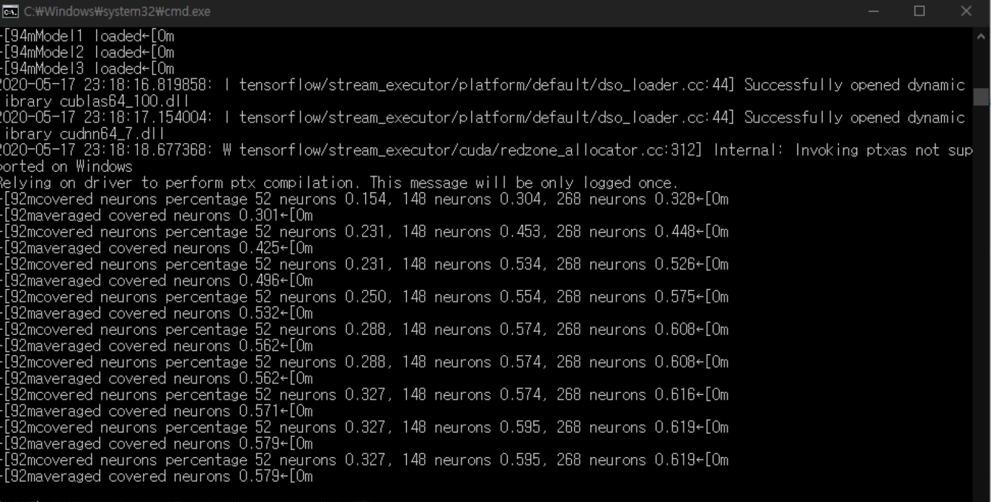
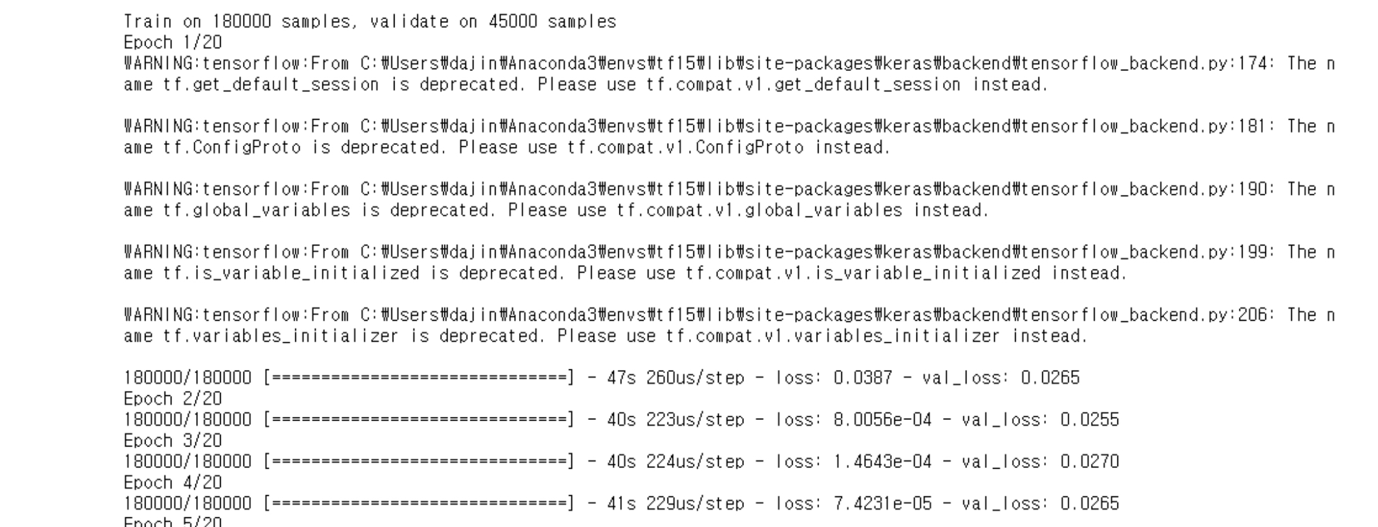
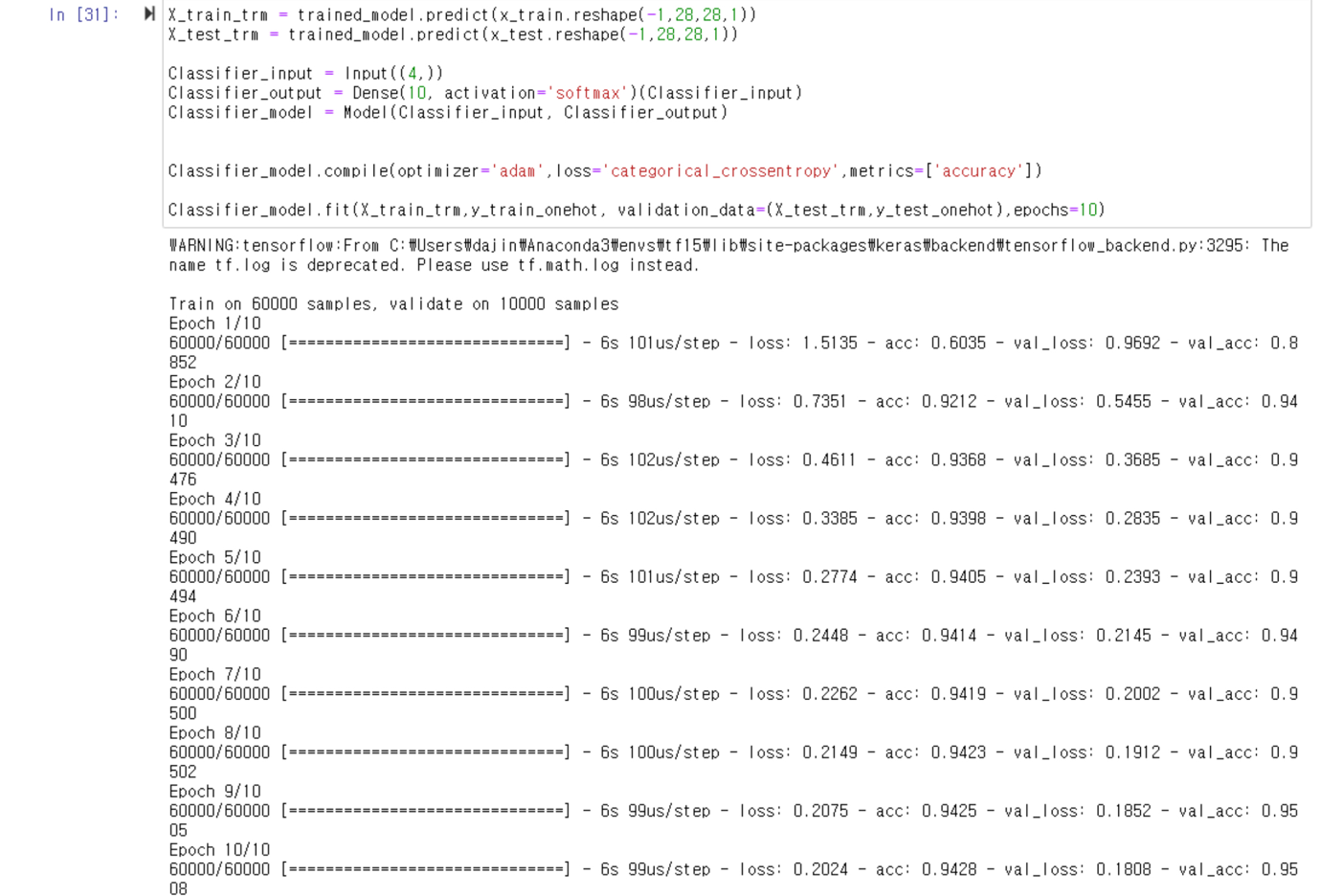
PBL assignment\_1

2016026080 한다진

* MNIST 데이터셋으로 학습된 CNN 모델에서 입력값의 획과 인접한 부분만 변하도록 하여 이미지를 생성하기

1. Quick start 및 핵심 parameters  
   cd MNIST  
   python gen\_diff.py --transformation line --weight\_diff 1 --weight\_nc 0.1 --step 10 --seeds 24 --grad\_iterations 20 --threshold 0.25
2. Key Concept  
   DeepXplore는 기본적으로 DNN의 edge case에 해당하는 이미지를 추출해내는 모델입니다. 기존의 고전적인 방식은 레이블링을 하는 수고가 많고, 아주 일부 픽셀에 대해서만 변화가 생깁니다. 그리고 그렇게 생성된 이미지들이 막는 edge case는 지나치게 edge case 입니다. 인간이 눈으로 보고 일부 픽셀만 인위적으로 바꾸는 과정이기 때문에 나타나는 상태입니다. Deepxplore는 이를 해결하기 위해 유사한 모델간의 방해과정을 통해 워본이미지와 다른 형태의 이미지를 만들어내는 아이디어를 냈습니다.   
   일반적인 CNN classifier 모델들은 back propagation을 하여 모델의 weight를 변경하는 것입니다. DeepXplore 논문의 아이디어는, back propagation과 유사하게 동작하여 모델의 resolution을 바꿔가며 결국 input image의 값에 변화를 주는 것입니다. 상수와 변수의 관게가 바뀐 것입니다.DeepXplore는 이 아이디어를 활용해, 이미지 전체에 변화를 조금씩 주는 모델을 생성했습니다.  
   생성된 이미지가 워본이미지와 다르게 판단하는 모델이 있도록 변화를 주어야 하고, 모델 내의 많은 뉴런이 이미지 판단에 영향력이 있도록 하는 방향으로 손실함수등이 설정되었습니다. 뉴런이 이미지 분류에 영향력이 있는지를 판단하는 기준은 해당 뉴런을 통해 나온 resoluution의 값이 threshold 보다 큰지로 판단합니다. 그리고 이러한 기준을 통해 영향력이 있는 뉴런과, 아직 없는 뉴런 등으로 나뉘게 됩니다. 이미지 입력이 들어올 때 마다 아직 활성화 되지 않은 뉴런을 활성화 시키도록, neuron\_to\_cover로 랜덤으로 고른 뒤 활성화 시키도록 원본 이미지에 변화를 줍니다. 이러한 방식으로 전반적인 코드가 돌아갑니다.
3. Code Explanation  
   constant\_line 함수를 통해 constant\_light 함수와 유사하게 gradients에 변화를 줍니다. 0.1은 원활한 이미지 변화를 위해 적용한 값입니다. 자세한 내용은 주석에 적혀 있습니다.  
     
   MNIST/gen\_diff.py  
   MNIST/utils.py  
   두 부분이 변경되었습니다.
4. result  
   코드를 실행한 결과 획 근처에 있는 픽셀 부분에서 변화가 생기는 이미지들이 생성되는 것을 확인할 수 있었습니다.

* 자유주제 – triplet loss 를 사용하는 모델고, 일반 cross entrophy를 쓰는 모델간의 비교를 통해 서로에게 부족한 이미지 케이스 생성하기

1. Triplet loss  
   triplet loss 는 입력으로 받은 이미지를 n차원 벡터값으로 만들고 같은 레이블의 데이터는 서로 가까워지게, 다른 레이블의 데이터를 멀어지도록 좌표값에 변화를 주는 방식으로 모델을 트레이닝 시키는 모델입니다. 그렇기 때문에 결과값은 n차원 벡터인데, 저는 mnist예제를 사용해서 4차원 벡터를 반환하는 모델을 트레이닝시켜서 사용했습니다.  
     
     
   4차원 벡터를 구한 다음 classifier 부분의 모델을 따로 생성하여 트레이닝 시켜서 이어붙였습니다. 두개의 모델을 한번에 이어서 하기가 까다로워서 나눠서 진행했습니다.  
     
     
     
     
   자세한 코드내용은 triplet\_loss폴더 내의 ipynb파일 안에 작성되어있습니다.
2. 코드 구현  
   MNIST\_triplet\_loss 폴더 내에서 작성되어있습니다. 앞서 트레이닝 시킨 모델을을 불러와서 이어붙이는 방식으로 했고 loss 함수를 개편하는데 어려움이 있어서 끝내 동작시키지는 못했습니다. Sequential statement와 functional statement에 따라서 활용할 수 있는 함수의 범위가 다른 것이 큰 문제였습니다. 또한 네트워크 안에 두개의 모델이 있기 때문에, model.get\_layyer(‘by name’) 방식으로 특정 레이어의 resolution을 구하는 것이 불가능 했기 때문에 프로그램을 돌리는데 문제가 있었습니다.  
     
     
   