PBL assignment\_1

2016026080 한다진

**#0 DeepXplore 논문 공부**

모든 Deep Neural Network에는 취약점이 있습니다. 학습데이터로 사용한 이미지가 아닌 새로운 이미지에 대해서는 어쩔 수 없이 정확도가 떨어지기도 하고, 이미지에 약간의 변화만 주어도 전혀 다른 결과를 내놓기도 합니다. 그래서 새로운 데이터셋을 생성하는 것이 중요하게 여겨졌고, 기존 데이터셋에서 새로운 데이터셋을 만드는 여러 고전적인 방법론이 제시되었습니다. 하지만 Edge-case 이미지를 생성하는 방식들은 대부분 레이블링을 매번 하는 수고가 동반되고, 특정 픽셀에만 큰 변화를 주는 방식이었습니다. 인간이 직접 하기 때문에 새로운 이미지를 생성하는 방식에 패턴이 존재했고, 이렇게 생성된 이미지는 실제 상황에서도 매우 edge-case였습니다. 그래서 효율적인 방식으로 새로운 데이터셋을 생성하고자 하는 방법이 많이 연구되었고 그중 하나가 DeepXplore입니다.

DeepXplore는 앞의 문제를 해결하기 위해, 학습된 딥러닝 모델을 활용하여 기존 학습데이터로부터 새로운 학습데이터를 생성하는 방식을 취했습니다. DeepXplore는 pre-trained된 모델을 사용하여 기존 모델에 취약한 이미지를 생성하고, 기존 모델을 fine tuning 합니다.

새로운 이미지를 생성하는 방식은 다음과 같습니다. 모델에 현재 이미지를 입력으로 줍니다. 모델은 현재 이미지를 받아서 결과값을 내고, 정답값과 비교하여 gradient값을 계산합니다. Pre-training 시에는 이 gradient 값을 모델의 layer의 weight값에 반영해줬지만, 지금은 이 gradient값을 첫번째 tensor에 영향을 주도록 합니다. 즉 입력으로 주어진 이미지에 반영해줍니다. 그렇게 새로운 이미지를 생성하는 방식입니다.

Fine-tuning의 핵심은 neuron coverage입니다. 모델 내부의 가능한 모든 뉴런이 활성화 되도록 추가 학습을 시킵니다. 여기서 활성화 되었다는 것은 해당 뉴런의 가중치값이 일정 threshold 값 이상이라는 것입니다. 모든 뉴런이 결과에 유효한 영향을 끼치도록 하는 방식입니다.

앞의 두 과정을 효율적으로 진행하기 위해서 DeepXplore는 input tensor와 output tensor 그리고 모델의 용도가 동일한 모델을 3개 사용하여 loss function을 normalize 했습니다. 특정 모델이 이미지 결과값에 방해를 주도록 영향을 끼치도록 하고 다른 모델은 이를 막도록 하여 특정 모델의 취약점이 부각되도록 새로운 이미지를 생성하도록 했습니다. 그리고 매번 각각의 모델들로부터 활성화 시킬 레이어의 노드를 찾아 그 값을 중심으로 fine tuning이 일어나도록 했습니다.

**#1 MNIST 이미지에서 획과 인접한 부분만 변하도록 하여 이미지를 생성하기**

1. Quick start

Run jupyter/0\_DeepXplore\_MNIST/gen\_diff.ipynb

1. Parameters

transformation line

weight\_diff 1

weight\_nc 0.1

step 10

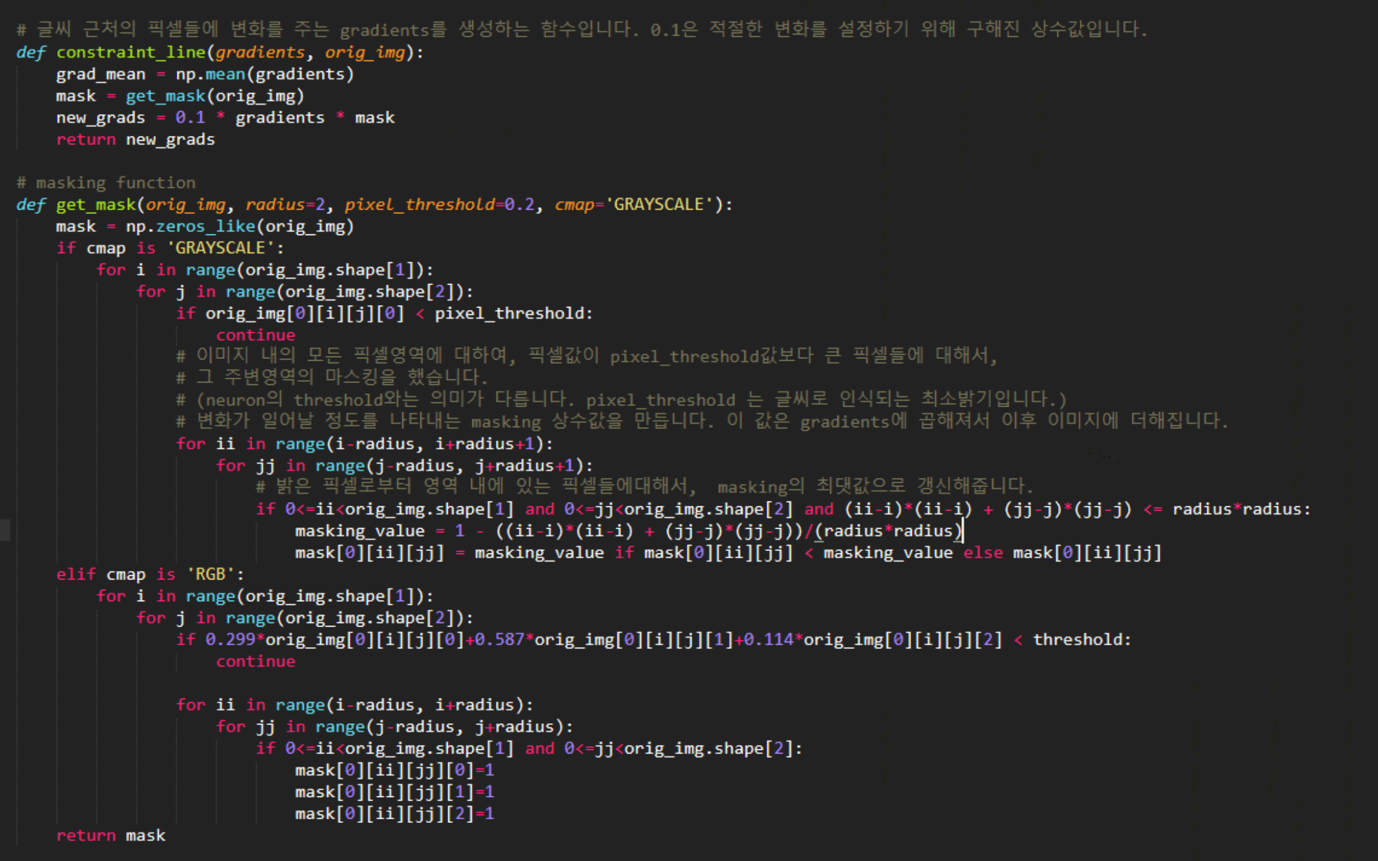
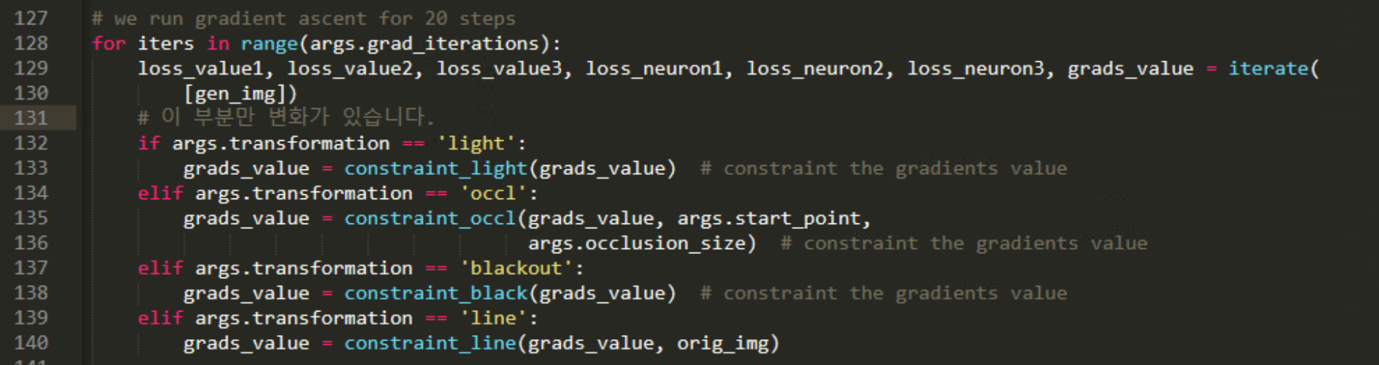
seeds 24

grad\_iterations 20

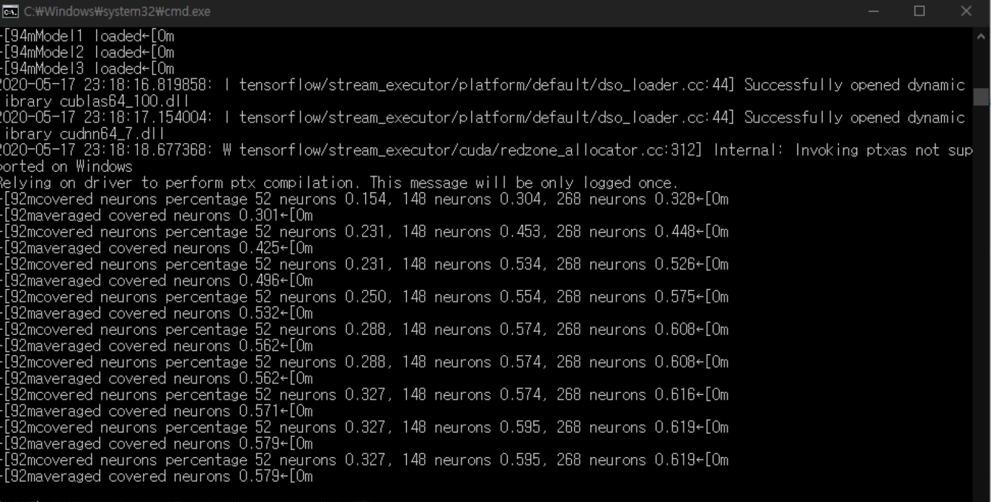
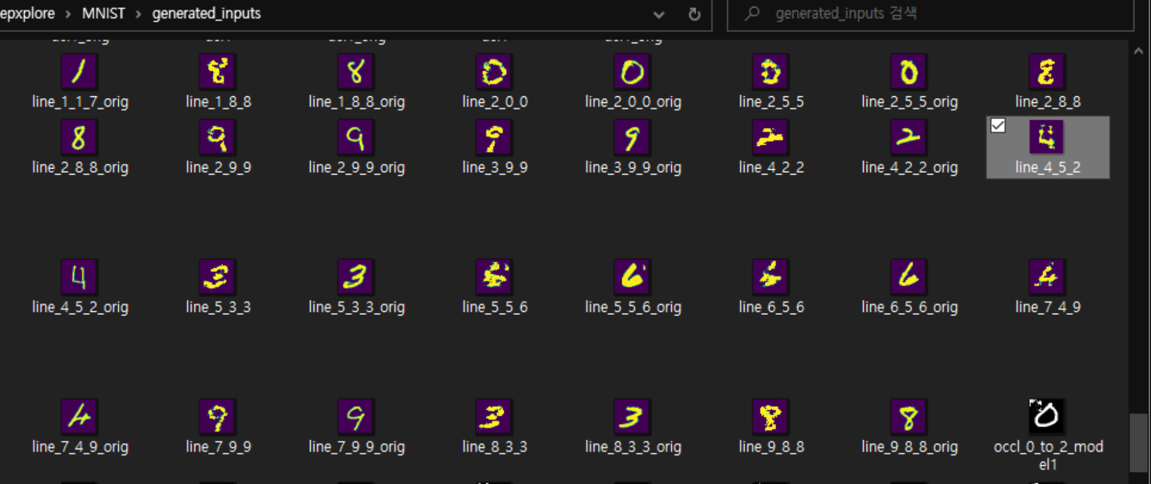
threshold 0.25

논문에 기재된 파라미터값과 동일한 값을 사용했습니다.

1. Code Explanation

- jupyter/0\_DeepXplore\_MNIST/utils.py  
  
- jupyter/0\_DeepXplore\_MNIST/gen\_diff.ipynb

constant\_line 함수를 통해 constant\_light 함수와 유사하게 gradients에 변화를 줍니다. 0.1은 원활한 이미지 변화를 위해 적용한 값입니다. 자세한 내용은 주석에 적어두었습니다.

1. Results

실행 결과, 손글씨의 획 근처부분에 변화를 주어 새로운 이미지를 생성하는 것을 확인했습니다. 대부분의 경우 논문의 내용처럼 target 으로 선정한 모델만 잘못 판단하도록 새로운 이미지가 생성되는 경향이 있는 것을 확인했습니다.

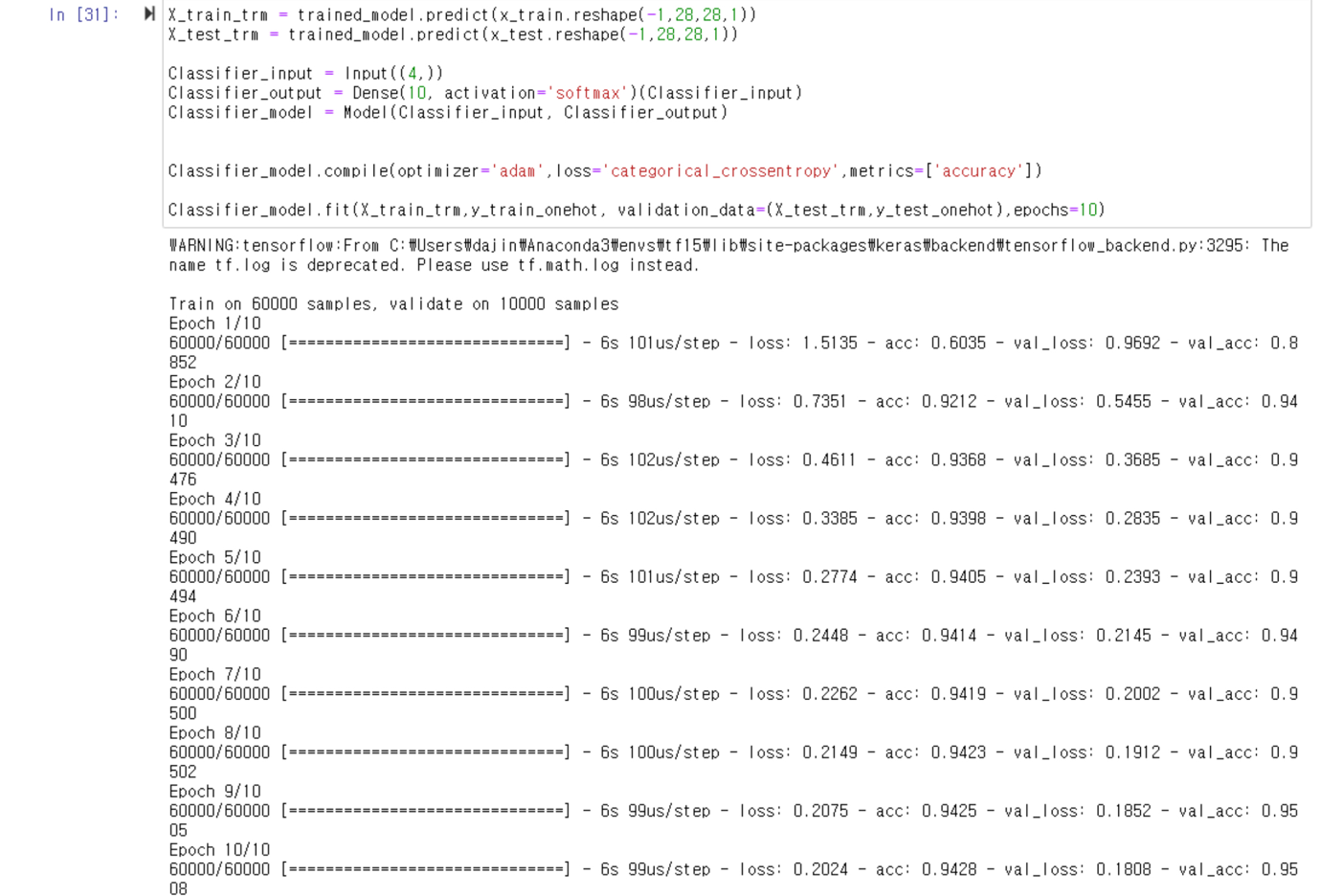
**#2 Triplet Loss Model**

1. 목표

Gradient 값을 원본 이미지에 반영해 주어서 이미지를 변형하는 것이, 좌표의 이동과 유사하다고 느껴졌습니다. 그래서 이미지 변화를 줄 때, A라는 레이블에서 B라는 레이블에 가까워지도록 이미지를 변형(이동)시킨다면 보다 직관적으로 다양한 이미지를 생성할 수 있을 것 같아서 관련 내용을 찾아보았습니다. 그 결과 이미지를 특정 벡터로 변환하는 모델을 찾을 수 있었고, 뒤의 추가적인 내용을 진행하기 전에 네트워크 구조를 알고자 소주제로 진행했습니다.

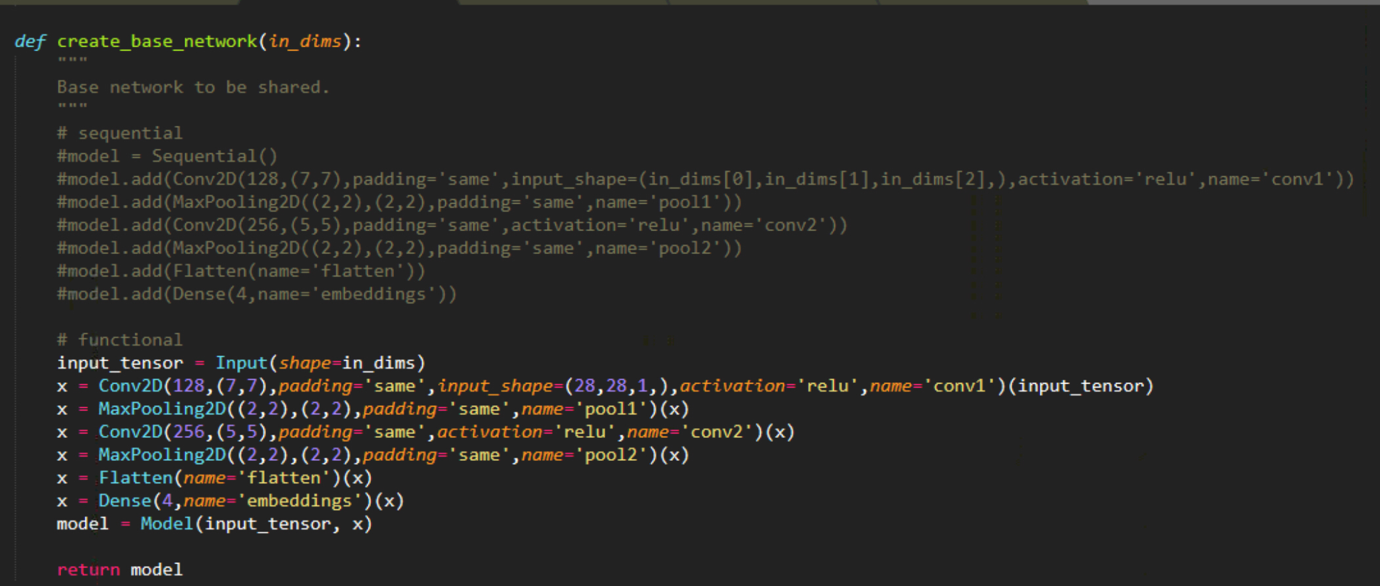
1. Triplet loss

기존의 CNN model은 이미지를 입력받아 해당 이미지가 각 label에 속할 확률이 얼마일지를 결과값으로 내는 모델입니다. 그에 반해 triplet loss를 사용한 model은 이미지를 n차원 벡터공간의 점으로 임베딩시키는 모델입니다. 유사한 레이블의 데이터들과 가까워지고, 다른 레이블의 데이터들과는 멀어지도록 이미지를 좌표값으로 임베딩시킵니다. 저는 최종적으로 4차원 벡터로 변환되도록 하였고, TSNE 함수를 사용하여 2차원의 점으로 표현하여 학습이 얼마나 잘 되었는지 시각적으로 확인했습니다.



1. Triplet Loss model Quick start

Run jupyter/1\_Triplet\_Loss/ModelTriplet\_MNIST.ipynb

1. Code Explanation

create\_base\_network 로 shared\_dnn을 만들어서 input\_anchor, positive\_anchor, negative\_anchor가 모두 거치도록 합니다. 그렇게해서 나온 결과를 받아와서 같은 레이블의 좌표는 가까워지도록, 다른 레이블의 좌표는 멀어지도록 학습시키는 구조입니다. 구체적인 사항은 jupyter/1\_Trplet\_Loss/ModelTriplet\_MNIST.ipynb에 작성하여 정리했습니다.

**#3 [실패] 서로 다른 loss function을 사용하는 두 모델이 서로의 약점을 파악하여 취약한 이미지 생성해주기**

1. 목표

서로 다른 loss function 을 사용하여 데이터를 트레이닝 시키는 두 모델이 각각 어떤 강점이 있고 어떤 약점이 있는지 파악해보고자 자유주제로 선정하여 실험을 해보았습니다. Triplet loss 로 학습된 모델이 기존 cnn모델이 좀더 유기적으로 이미지를 인식할 수 있도록 영향을 줄 수 있을 것이라 생각하여 진행했습니다.

1. Code explanation

Input tensor 와 output tensor의 규격이 동일해야 DeepXplore의 기존 코드를 활용하기 용이하기 때문에 pre-trained된 모델의 뒷부분에 4차원 입력을 받아 10개의 레이블별 확률값을 반환하는 pre-trained 된 classifier를 추가하였습니다.

1. 실패 원인

기본적으로 DeepXplore에서 사용된 모델들이 layer로만 구성된 것에 반해, triplet\_loss를 활용한 모델은 create\_base\_network로 생성된 sub model을 내부 망으로 사용하고 있습니다. 따라서 활성화할 레이어의 뉴런을 고를 때, 접근하는 방식이 기존의 방식과 매우 달랐습니다. 그리고 iterate function에 대한 사용법 및 전반적인 tensorflow를 사용하는 방식이 서툴러서 헤매는 시간이 길었습니다. 만일 지금 다시 진행한다면 성공적으로 실험을 할 수 있을 것 같습니다.

**#4 Triplet Loss로 만든 모델의 gradient 값을 사용하여 이미지를 바꾸고, 일반적인 cnn 모델로 검증하여, triplet loss를 사용한 모델에 취약한 이미지 생성하기.**

1. 목표

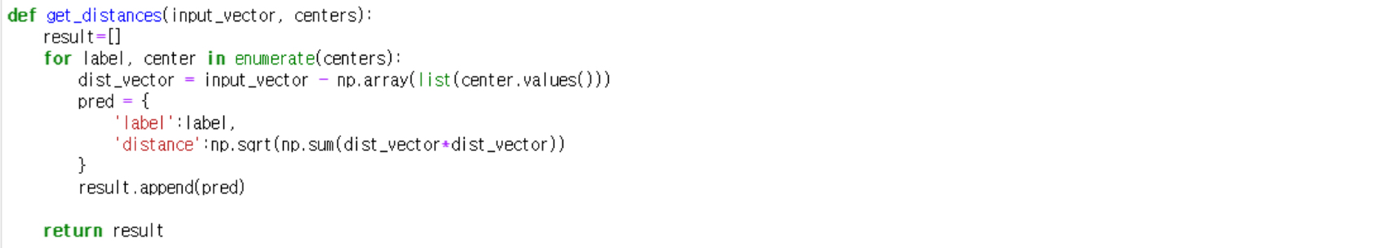
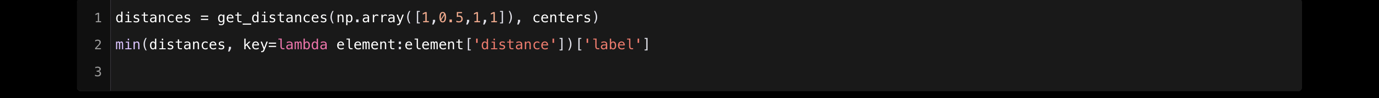
이미지를 임베딩하여 나온 벡터값을 이용해서 현재 이미지가 다른 레이블에 가까워지도록 변형시킨다.

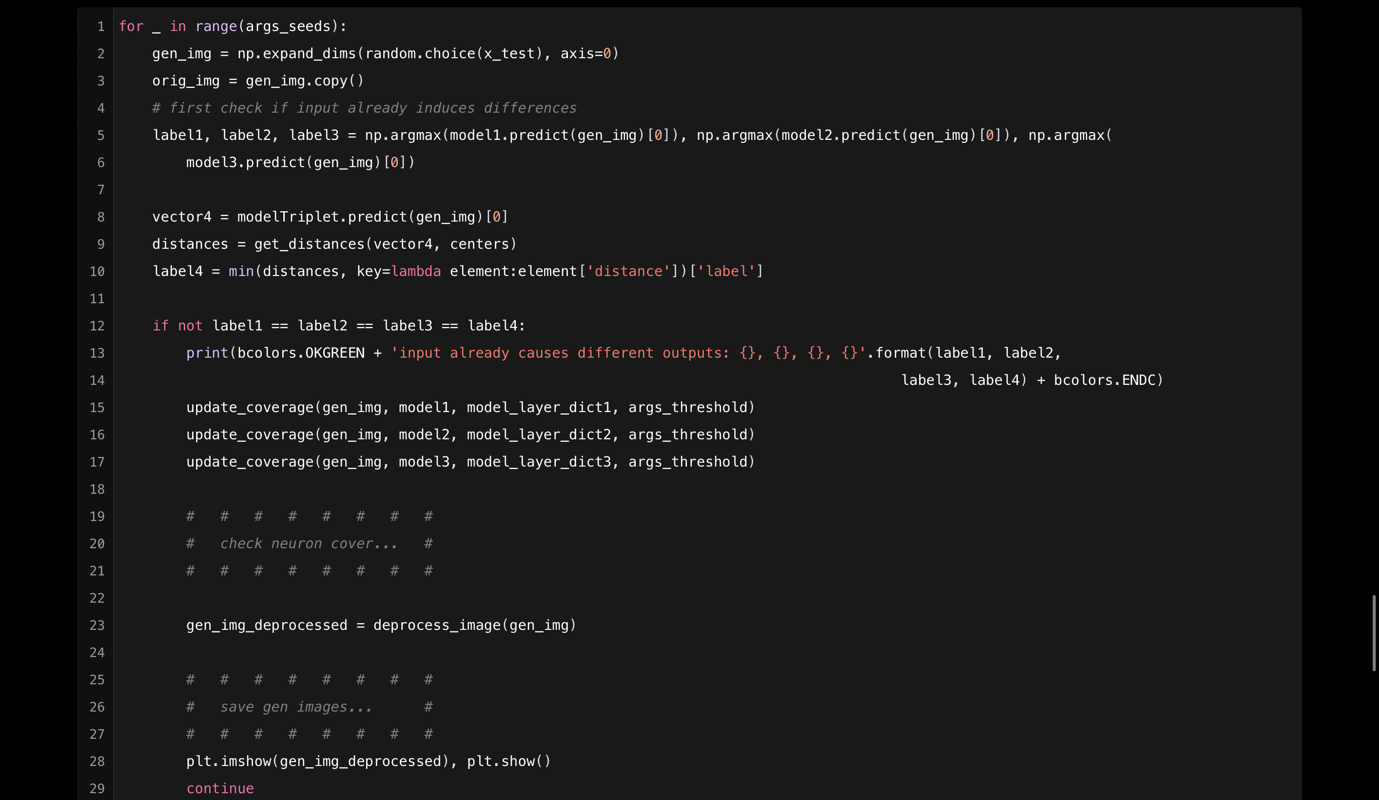
1. Quick start

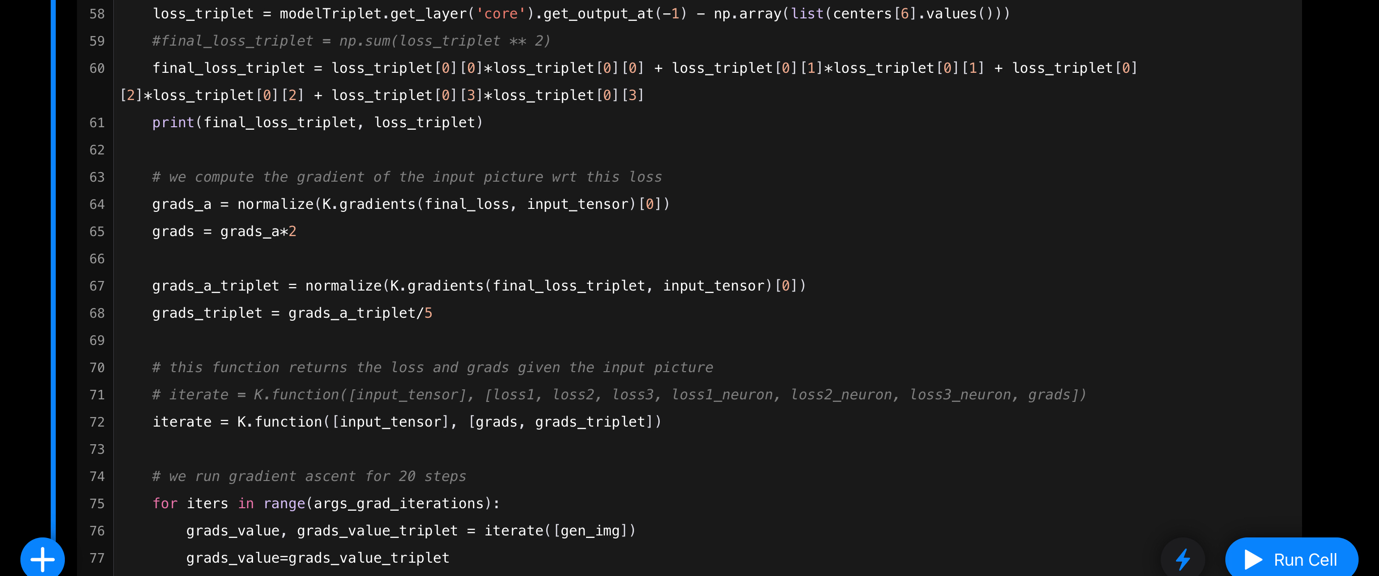
Run jupyter/2\_DeepXplore\_Triplet\_Loss/gen\_diff.ipynb

1. Code explanation

이미지를 임베딩하여 나온 벡터값을 이용해서 현재 이미지가 다른 레이블에 가까워지도록 변형시킨다. 일반적인 loss function을 가진 모델은 위와 같이 생성된 모델에 대해서 robust 한 특징을 갖고 있다. 그래서 일반적인 모델로 검증을 하며, triplet\_loss 를 사용한 모델이 보다 잘 학습되도록, 취약한 이미지를 추가생성한다.

- jupyter/2\_DeepXplore\_Triplet\_Loss/ModelTriplet.py

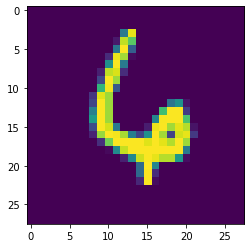
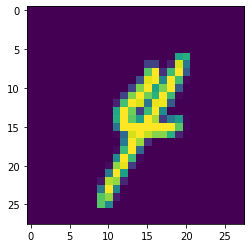
- jupyter/2\_DeepXplore\_Triplet\_Loss/gen\_diff.ipynb



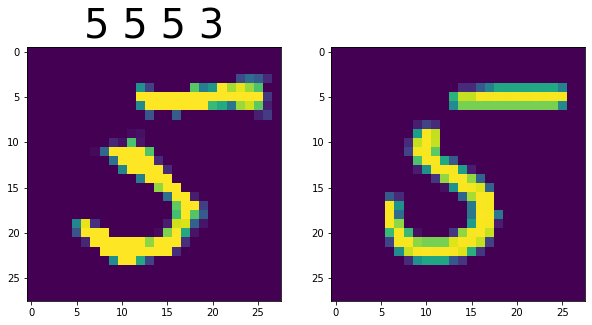
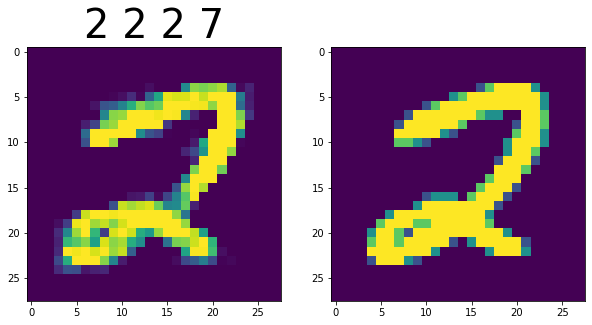
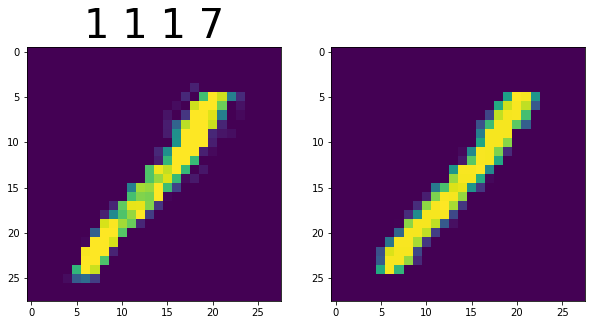
이번에는 triplet loss model이 label을 판단하는 방식을 바꾸었습니다. #3에서는 4차원 벡터값을 입력으로 받고, 10개 레이블의 확률값을 출력하는 pre-trained 된 모델을 뒤에 붙여서 동작시켰습니다. 이번에는 training dataset에 있는 모든 이미지가 임베딩된 좌표값을 구하고, 레이블별로 좌표값의 평균을 구해 그 값을 해당 레이블의 중심좌표로 설정했습니다. 그리고 새로운 이미지가 들어올 때 마다, 가장 가까이에 있는 중심 좌표의 레이블값을 입력으로 주어진 이미지의 레이블로 판단하도록 했습니다.

현재 입력 이미지의 embedding 값을 image\_vector, 이동시키고자 하는 target label의 중심 좌표를 target\_vector 라고 표현한다면, gradient 값은 input\_vector와 target\_vector의 mean squared error 값으로 처리했습니다. 그렇게 해서 현재 input vector가 target\_vector로 이동하도록 입력 이미지에 변화를 주었습니다.

Iterate 함수를 변형하여, 매 iteration 마다, 원하는 gradient 값을 tensor로 부터 받을 수 있도록 설정하였고, 원활한 실험을 위해 neuron coverage 부분은 생략했습니다.

1. results

우선 triplet\_loss 로 학습된 모델과 일반적인 ce loss 로 학습된 모델의 차이를 확인할 수 있었습니다. ce loss 로 학습된 모델의 경우, 왼쪽과 오른쪽의 이미지를 모두 4로 판단하였고, triplet loss로 학습된 모델의 경우 왼쪽을 6, 오른쪽을 9로 판단했습니다.



이후 light, line 등 파라미터 값을 바꾸어 가며, triplet loss 로 학습된 모델에 취약한 이미지를 생성하는 것을 확인할 수 있었습니다. 위의 이미지는 line에 변화를 주어 이미지 변형을 한 예시입니다. 왼쪽의 이미지가 새로 생성된 이미지이고, 오른쪽의 이미지가 원본입니다.