

프로젝트 1. 경사 하강법으로 이미지 복원하기

프로젝트 개요와 목표

이번 프로젝트에서 우리가 풀 문제는 다음과 같습니다.

이미지 처리를 위해 만들어 두었던 `weird_function()` 함수에 실수로 버그가 들어가 100×100 픽셀의 오염된 이미지가 만들어졌습니다. 이 오염된 이미지와 오염되기 전 원본 이미지를 동시에 파일로 저장하려고 했으나, 모종의 이유로 원본 이미지 파일은 삭제된 상황입니다. 다행히도 `weird_function()`의 소스코드는 남아 있습니다. 오염된 이미지와 `weird_function()`을 활용해 원본 이미지를 복원해봅시다.

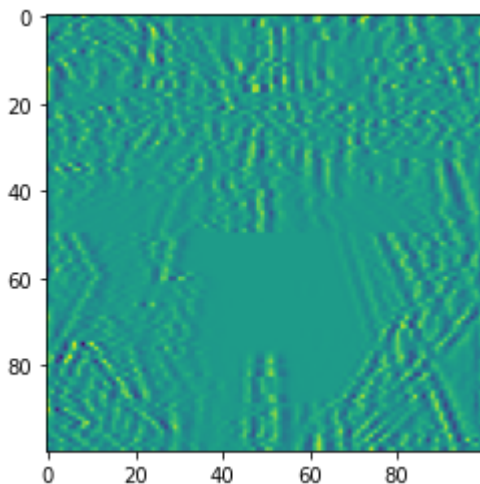
참고자료: <https://github.com/jcjohnson/pytorch-examples>, NYU Intro2ML

```
In [1]: import torch
import pickle
import matplotlib.pyplot as plt
```

```
In [2]: shp_original_img = (100, 100)
broken_image = torch.FloatTensor( pickle.load(open('./broken_image.t.p', 'rb'), encod
```

```
In [3]: plt.imshow(broken_image.view(100,100))
```

Out[3]: <matplotlib.image.AxesImage at 0x222ff9bd5e0>



```
In [4]: def weird_function(x, n_iter=5):
    h = x
    filt = torch.tensor([-1./3, 1./3, -1./3])
    for i in range(n_iter):
        zero_tensor = torch.tensor([1.0*0])
        h_l = torch.cat( (zero_tensor, h[:-1]), 0)
        h_r = torch.cat((h[1:], zero_tensor), 0 )
        h = filt[0] * h + filt[2] * h_l + filt[1] * h_r
        if i % 2 == 0:
            h = torch.cat( (h[h.shape[0]//2:], h[:h.shape[0]//2]), 0 )
    return h
```

```
In [5]: def distance_loss(hypothesis, broken_image):
```

```
return torch.dist(hypothesis, broken_image)
```

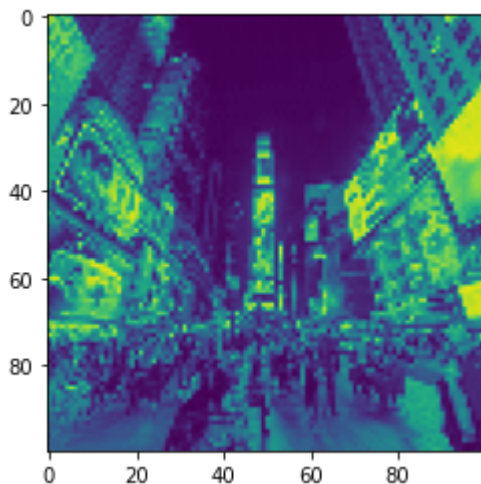
```
In [6]: random_tensor = torch.randn(10000, dtype = torch.float)
```

```
In [7]: lr = 0.8
for i in range(0,20000):
    random_tensor.requires_grad_(True)
    hypothesis = weird_function(random_tensor)
    loss = distance_loss(hypothesis, broken_image)
    loss.backward()
    with torch.no_grad():
        random_tensor = random_tensor - lr*random_tensor.grad
    if i % 1000 == 0:
        print('Loss at {} = {}'.format(i, loss.item()))
```

```
Loss at 0 = 12.233193397521973
Loss at 1000 = 1.1496862173080444
Loss at 2000 = 0.5472530126571655
Loss at 3000 = 0.37465205788612366
Loss at 4000 = 0.29435327649116516
Loss at 5000 = 0.2454175055027008
Loss at 6000 = 0.21033678948879242
Loss at 7000 = 0.1825464367866516
Loss at 8000 = 0.15904547274112701
Loss at 9000 = 0.13827843964099884
Loss at 10000 = 0.11936576664447784
Loss at 11000 = 0.1017724797129631
Loss at 12000 = 0.08515296876430511
Loss at 13000 = 0.069271519780159
Loss at 14000 = 0.05396117642521858
Loss at 15000 = 0.039100199937820435
Loss at 16000 = 0.024600720033049583
Loss at 17000 = 0.021161476150155067
Loss at 18000 = 0.021166730672121048
Loss at 19000 = 0.021167723461985588
```

```
In [8]: plt.imshow(random_tensor.view(100,100).data)
```

```
Out[8]: <matplotlib.image.AxesImage at 0x222ffab66d0>
```



201600282 엄기산

```
In [ ]:
```

