

# EnlightenGANを使ってみる

論文：<https://arxiv.org/pdf/1906.06972.pdf>

コード：<https://github.com/TAMU-VITA/EnlightenGAN>

どんな研究？

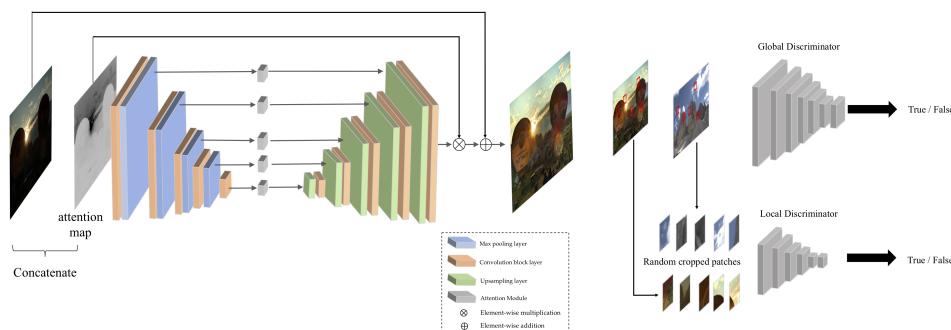


左のような低照度画像を入力としてあたえると、右のような照度を調整された画像が出力されるようなネットワークです。

どこがすごいの？

今までの既存手法では教師データとして同じ場所の低照度画像と通常照度画像のペアが必要だったが本手法では教師データは通常照度画像のみ。

どうやってるの？



- GAN(LSGAN,PatchGAN)を使用している。
  - U-netを元にしたGeneratorを採用することで暗い部分に注目し画像の再構成を行う。
  - 画像全体の判別を行うGlobalDiscriminatorとランダムに取り出されたパッチ(1画像につき5パッチ)の判別を行うLocalDiscriminatorを組み合わせる。
  - attention mapとLocalDiscriminatorが存在することで画像の一部分の色が歪んでいたり周りとくらべて不自然に明るいということがなくなる。

(損失関数の話とか結構大事なので、ここらへんはまた加筆します。.)

## 環境構築

1. DockerでKerasを使ったディープラーニングの環境を構築するを実施してください。

2. 適当にディレクトリを作成し、そこに移動してください。

```
mkdir engan  
cd engan
```

3. コードをクローンしましょう。

```
git clone https://github.com/TAMU-VITA/EnlightenGAN.git
```

4. 必要なディレクトリの作成し、必要な学習済みの重みファイル、入力ファイルなどを入れてください。  
(著者の[git](#)にアクセスするとDLするためのリンクがあります)

```
tree -d
```

を実行したときに

```
.  
├── EnlightenGAN  
│   ├── ablation [error opening dir]  
│   ├── assets  
│   ├── checkpoints [error opening dir]  
│   ├── configs  
│   ├── data  
│   │   └── __pycache__ [error opening dir]  
│   ├── datasets  
│   │   └── bibtex  
│   ├── imgs  
│   └── lib  
│       ├── nn  
│       │   ├── __pycache__ [error opening dir]  
│       │   ├── modules  
│       │   │   ├── __pycache__ [error opening dir]  
│       │   │   └── tests  
│       │   └── parallel  
│       │       └── __pycache__ [error opening dir]  
│       └── utils  
│           └── data  
└── model  
└── models  
    └── __pycache__ [error opening dir]  
└── options  
    └── __pycache__ [error opening dir]  
└── scripts  
└── seg  
└── util  
    └── __pycache__ [error opening dir]
```

```

final_dataset
├── trainA
└── trainB
test_dataset
├── testA
│   ├── data
│   │   ├── DICM
│   │   ├── LIME
│   │   ├── MEF
│   │   ├── NPE
│   │   ├── VV
│   │   └── data 11
│   │       └── NPE-ex1
│   ├── data 12
│   │   └── NPE-ex2
│   └── data 13
│       └── NPE-ex3
└── image
testB

```

という構造になっていれば問題ないはずです。

#### 5. Dockerイメージを引っ張ってきて、起動します。

```

sudo docker login
sudo docker pull yaegasikk/olab-engan:latest
sudo docker run --gpus all --shm-size=4gb -v $(pwd):/user/local -d -p
10000:22 yaegasikk/olab-engan:latest

```

ここでは使用するポート番号を適当に10000を使用していますが必要に応じて変更してください。`--shm-size`は`/dev/shm`に割り当てる容量を変更するオプションです。デフォルトでは64MBですが、PyTorchでは足りないようで指定しないとエラーが出ます。4 GBあれば私の環境では動いたので4 GBあればいいと思います。起動するときは必ず最初に作成したディレクト内で行ってください。

#### 6. 起動したらコンテナの中にアクセスします。

```
ssh -X yaegasi@localhost -p 10000
```

このとき、パスワードは*screencast*です。また、rootでログインしてもいいですが、VScodeが使えません。

## 学習

論文のなかではバッチサイズを32にして200エポック学習するのにNvidia 1080Ti三枚使用して3時間かかったとあり、学習にはそれなりに時間がかかるようです。著者のgitによると3枚GPU用意するかバッチサイズを変更しろとのことなので学習するのにバッチサイズとGPUの枚数を変更する。

```
code ./EnlightenGAN/scripts/script.py
```

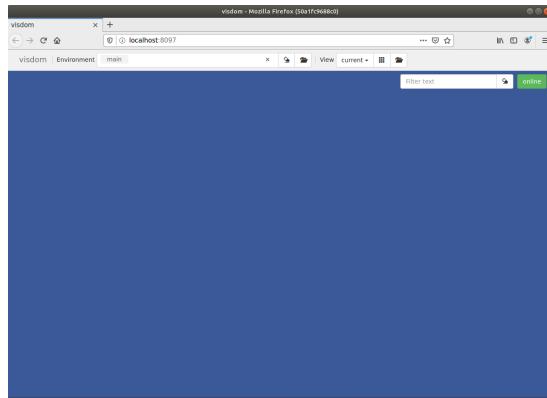
でVScodeを起動し,27行目を `--batchSize 32` から `--batchSize 16` に, 37行目を `--gpu_ids 0,1,2` から `--gpu_ids 0` に変更する. (GPU二枚使用可能な場合は `--gpu_ids 0,1`とする. )

保存できたら `visdom.server` を起動する.

```
sudo nohup python -m visdom.server -port=8097
```

起動に成功したら `firefox` を起動する.

```
firefox localhost:8097 &
```

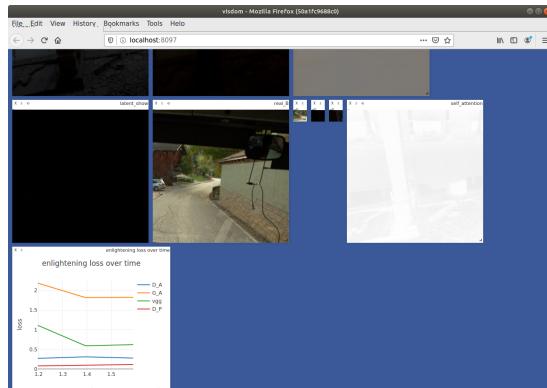


こんな画像がでてくるはず. やっと学習です.

```
cd cd /user/local/EnlightenGAN/
sudo python scripts/script.py --train
```

(このdockerイメージでは `python` でバージョンが3.5のPythonが動きます. `python3` で動かす環境でこのコードを動かす場合 `script.py` の中を変更する必要があります. )

うまくいけば `firefox` が下のような状態になるはずです.



```
RuntimeError: cuda runtime error (2) : out of memory at
/pytorch/torch/lib/THC/generic/THCStorage.cu:58
```

みたいなエラーが出たらバッチサイズをさらに小さくしてください。

## 予測

低照度画像を *test\_dataset/testA* の中に入れてください。上記の著者のサイト通りに入力ファイルを持ってきていればテスト用の画像がそこにいるはずです。

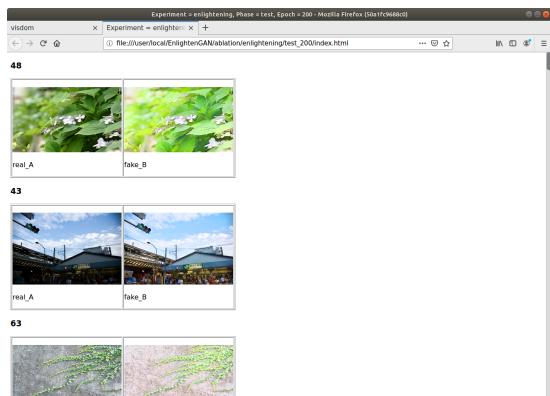
また、学習を行わない場合は著者のサイトから学習済みの重みファイルをDLして *checkpoints/enlightening* の中にいれてください。

```
cd /user/local/EnlightenGAN/
sudo python scripts/script.py --predict
```

で動くはずです。予測が終了したら

```
firefox ./ablation/enlightening/test_200/index.html &
```

を実行すれば



ここから、結果の方をみることができます。

## 参考サイト

[yaegasikk/gui-docker Dockerイメージとコンテナの削除方法](#)

## 備考

CVPR2020でEnlightenGANを上回る手法として、Zero-DCEという手法が提案されています。

論文：[http://openaccess.thecvf.com/content\\_CVPR\\_2020/papers/Guo\\_Zero-Reference\\_Deep\\_Curve\\_Estimation\\_for\\_Low-Light\\_Image\\_Enhancement\\_CVPR\\_2020\\_paper.pdf](http://openaccess.thecvf.com/content_CVPR_2020/papers/Guo_Zero-Reference_Deep_Curve_Estimation_for_Low-Light_Image_Enhancement_CVPR_2020_paper.pdf)

コード：<https://github.com/Li-Chongyi/Zero-DCE>

(2020年6月22日現在コード未公開)