GAN: 얼굴 이미지

2022.08.02

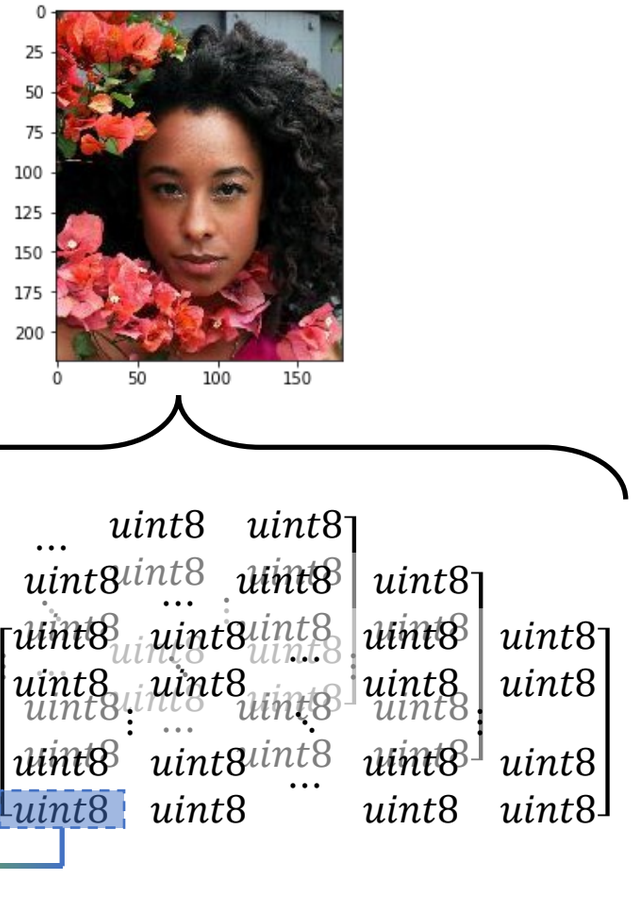
발제자: 오효근

[1] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

CelebA Dataset

- Image Data: $H \times W \times 3$
  - Height (높이): 218
  - Width (너비): 178
  - RGB (Red, Green, Blue): 3

- Training Data Description [1]
  - $218 \times 178 \times 3$ 사이즈의 20,000장 이미지
  - 눈과 입의 위치가 비슷한 좌표에 위치하도록 조정됨
  - .jpeg 포맷 사용



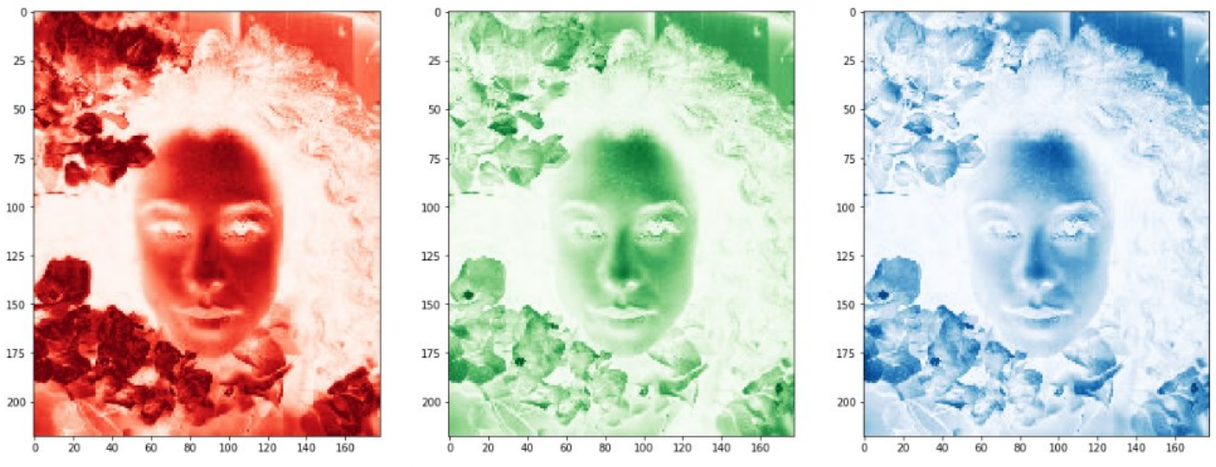


Fig 1. RGB separated example image

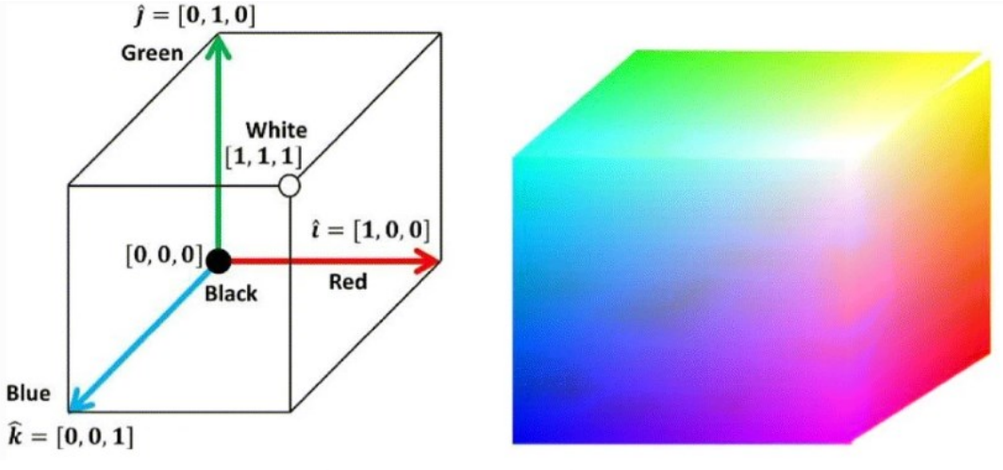


Fig 2. Schematic of image data

Hierarchical Data Format (HDF)

- 문제점: 다수의 .jpeg 파일을 훈련 과정에서 열고 닫을 시 시간 소모가 매우 큼

- Hierarchical Data Format: 용량이 매우 큰 데이터에 효과적으로 접근하기 위해 만들어진 데이터 형식
  - 하나 이상의 그룹을 가질 수 있어 계층적이라 불림
  - 그룹 안에 여러 개의 데이터셋이 포함될 수 있음
  - HDF5 (HDF version 5)를 이용해 훈련 과정에서의 <u>시간 소모 개선</u> 및 <u>메모리의 한계 극복</u>

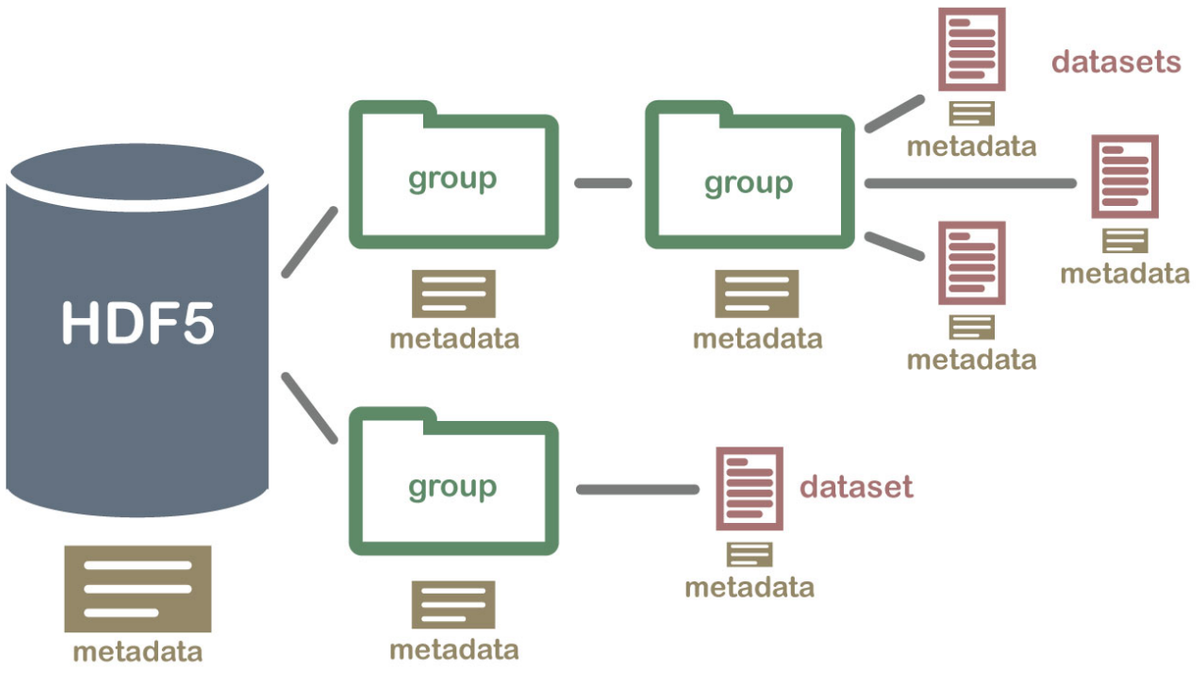


Fig 3. Schematic of HDF5

```python
import torchvision.datasets

CelebA_dataset = torchvision.datasets.CelebA(root='.', download=True)

import h5py
import zipfile
import imageio
import os

hdf5_file = 'celeba/celeba_aligned_small.h5py'

total_images = 20000

with h5py.File(hdf5_file, 'w') as hf:
    count = 0
    with zipfile.ZipFile('celeba/img_align_celeba.zip', 'r') as zf:
        for i in zf.namelist():
            if (i[-4:] == '.jpg'):
                ofile = zf.extract(i)
                img = imageio.imread(ofile)
                os.remove(ofile)
                hf.create_dataset('img_align_celeba/'+str(count)+'.jpg', data=img, compression="gzip", compression_opts=9)
                count = count + 1
                if (count%1000 == 0):
                    print("images done .. ", count)
                    pass
                if (count == total_images):
                    break
                pass
            pass
        pass
```

Fig 4. The process of converting the saved image dataset to HDF5 through the h5py library

Hierarchical Data Format (HDF)

- __init__(): HDF5 파일을 열고 img_align_celeba로 각각의 이미지에 접근
- __len__(): 그룹 안의 데이터 수 반환
- __getitem__(): Index를 이미지의 이름으로 변환하고 이미지 데이터 반환

```python
class CelebADataset(Dataset):
    def __init__(self, file):
        self.file_object = h5py.File(file, 'r')
        self.dataset = self.file_object['img_align_celeba']
        pass

    def __len__(self):
        return len(self.dataset)

    def __getitem__(self, index):
        if (index >= len(self.dataset)):
            raise IndexError()
        img = np.array(self.dataset[str(index)+'.jpg'])
        return torch.cuda.FloatTensor(img) / 255.0

    def plot_image(self, index):
        plt.imshow(np.array(self.dataset[str(index)+'.jpg']), interpolation='nearest')
        pass
    pass

celeba_dataset = CelebADataset('celeba/celeba_aligned_small.h5py')
```

Fig 5. class CelebADataset

Discriminator

- Input of Discriminator: $(218, 178, 3) \rightarrow 218 \times 178 \times 3 = 116412$
  - 구성한 신경망이 완전 연결 신경망이므로 **일관된 기준**을 통해 정렬
  - 이미지를 어떻게 풀어서 정렬하는지는 중요하지 않음

- Class View()
  - nn.Module을 상속하여 Sequential 내에서 다른 모듈과 함께 사용 가능

```python
class View(nn.Module):
    def __init__(self, shape):
        super().__init__()
        self.shape = shape,

    def forward(self, x):
        return x.view(*self.shape)
```

Fig 6. class View

```python
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            View(218*178*3),
            nn.Linear(3*218*178, 100),
            nn.LeakyReLU(),
            nn.LayerNorm(100),
            nn.Linear(100, 1),
            nn.Sigmoid()
        )
        self.loss_function = nn.BCELoss()
        self.optimiser = torch.optim.Adam(self.parameters(), lr = 0.0001)
        self.optimiser.param_groups[0]['capturable'] = True
        self.counter = 0
        self.progress = []
        pass
```

(218, 178, 3) → 218 × 178 × 3 = 116412

Fig 7. class Discriminator

Generator

- Input of Generator: 100
  - $100 \rightarrow 300 \rightarrow 3 \times 218 \times 178 \rightarrow (218, 178, 3)$

```
class Generator(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(100, 3*10*10),
            nn.LeakyReLU(),
            nn.LayerNorm(3*10*10),
            nn.Linear(3*10*10, 3*218*178),
            nn.Sigmoid(),
            View((218, 178, 3))
        )
        self.optimiser = torch.optim.Adam(self.parameters(), lr = 0.0001)
        self.optimiser.param_groups[0]['capturable'] = True
        self.counter = 0
        self.progress = []
        pass
```

Fig 8. class Generator

```
G = Generator()
G.to(device)

output = G.forward(generate_random_seed(100))
img = output.detach().cpu().numpy()
plt.imshow(img, interpolation = 'none', cmap = 'Blues')
```

```
<matplotlib.image.AxesImage at 0x2c61548ab20>
```

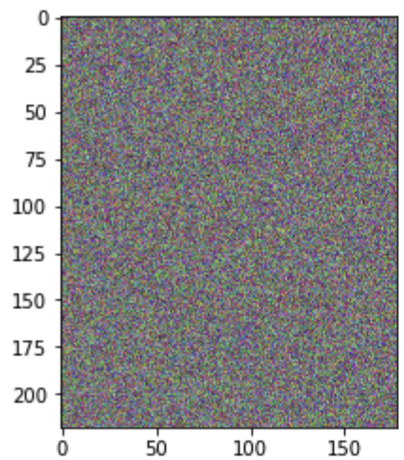


Fig 9. Generated image from untrained generator

GAN Training

❑ 생성기는 직접 이미지를 통해 훈련하지 않고 이미지를 생성할 때 **훈련 데이터의 우도 (likelihood)** 사용

```
%%time

D = Discriminator()
D.to(device)
G = Generator()
G.to(device)

epochs = 1

for epoch in range(epochs):
    for image_data_tensor in celeba_dataset:
        D.train(image_data_tensor, torch.cuda.FloatTensor([1.0]))
        D.train(G.forward(generate_random_seed(100)).detach(), torc.cuda.FloatTensor([0.0]))
        G.train(D, generate_random_seed(100), torch.cuda.FloatTensor([1.0]))
        pass
    pass
```

Fig 10. GAN training

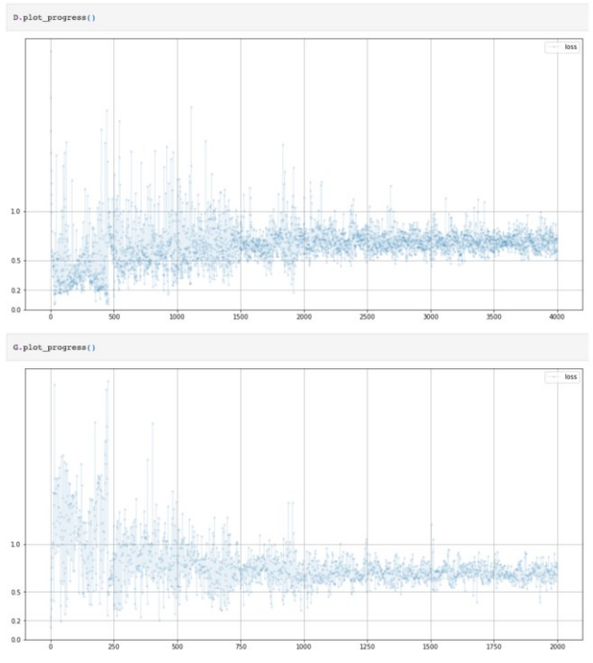


Fig 11. Loss of GAN training process

```
f, axarr = plt.subplots(2, 3, figsize = (16, 8))
for i in range(2):
    for j in range(3):
        output = G.forward(generate_random_seed(100))
        img = output.detach().cpu().numpy()
        axarr[i, j].imshow(img, interpolation = 'none', cmap = 'Blues')
        pass
```

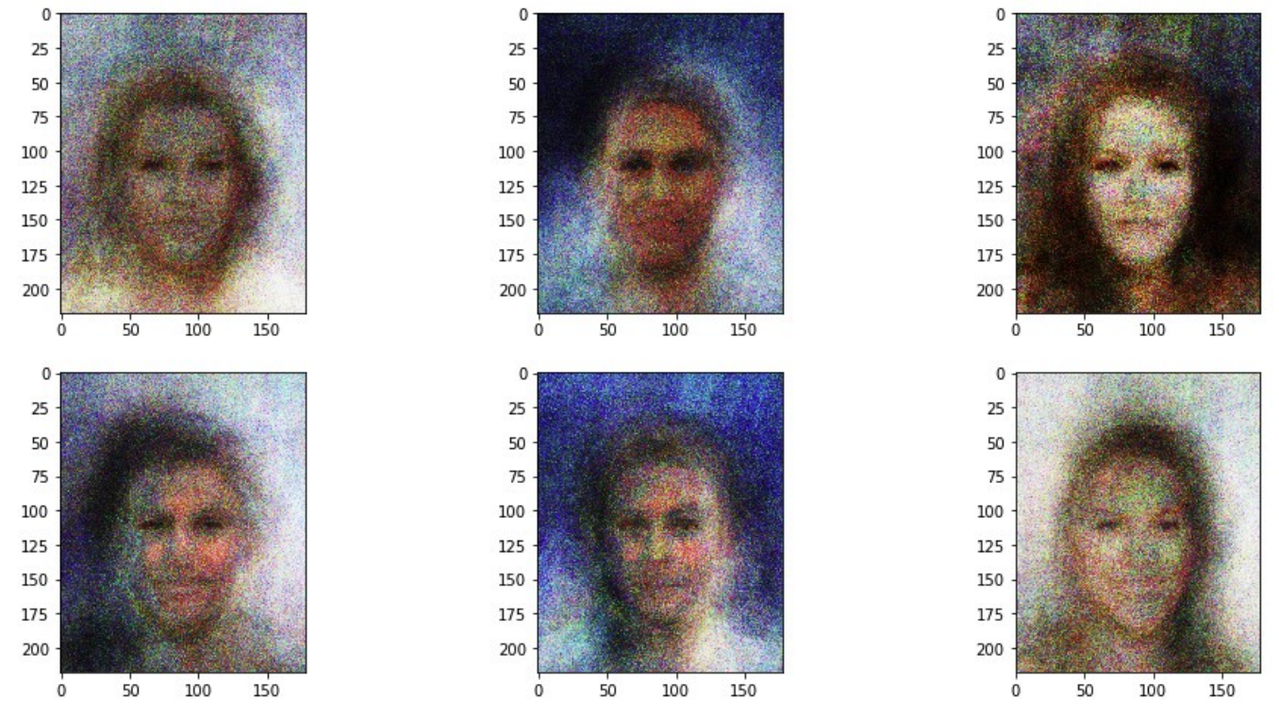


Fig 12. Results of generator

[2] https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=618

Montage Dataset

- Training Data Description [2]
  - 720 × 540 × 3 사이즈의 8,071장 이미지
  → GPU memory 부족
  → 240 × 180 × 3 사이즈로 압축
  - .png 포맷 사용

Fig 13. Sample image of montage dataset

```
%%time

import h5py
import zipfile
import imageio
import os
import cv2

hdf5_file = 'montage/montage_small.h5py'

total_images = 8071

with h5py.File(hdf5_file, 'w') as hf:
    count = 0
    with zipfile.ZipFile('montage/montage.zip', 'r') as zf:
        for i in zf.namelist():
            if (i[-4:] == '.png'):
                ofile = zf.extract(i)
                img = imageio.imread(ofile)
                img = cv2.resize(img, dsize = (180, 240), interpolation=cv2.INTER_CUBIC)
                os.remove(ofile)
                hf.create_dataset('montage/'+str(count)+'.png', data=img, compression="gzip", compression_opts=9)
                count = count + 1
                if (count%1000 == 0):
                    print("images done .. ", count)
                    pass
                if (count == total_images):
                    break
                pass
            pass
        pass
```

Fig 14. Save as HDF5 after resizing the image

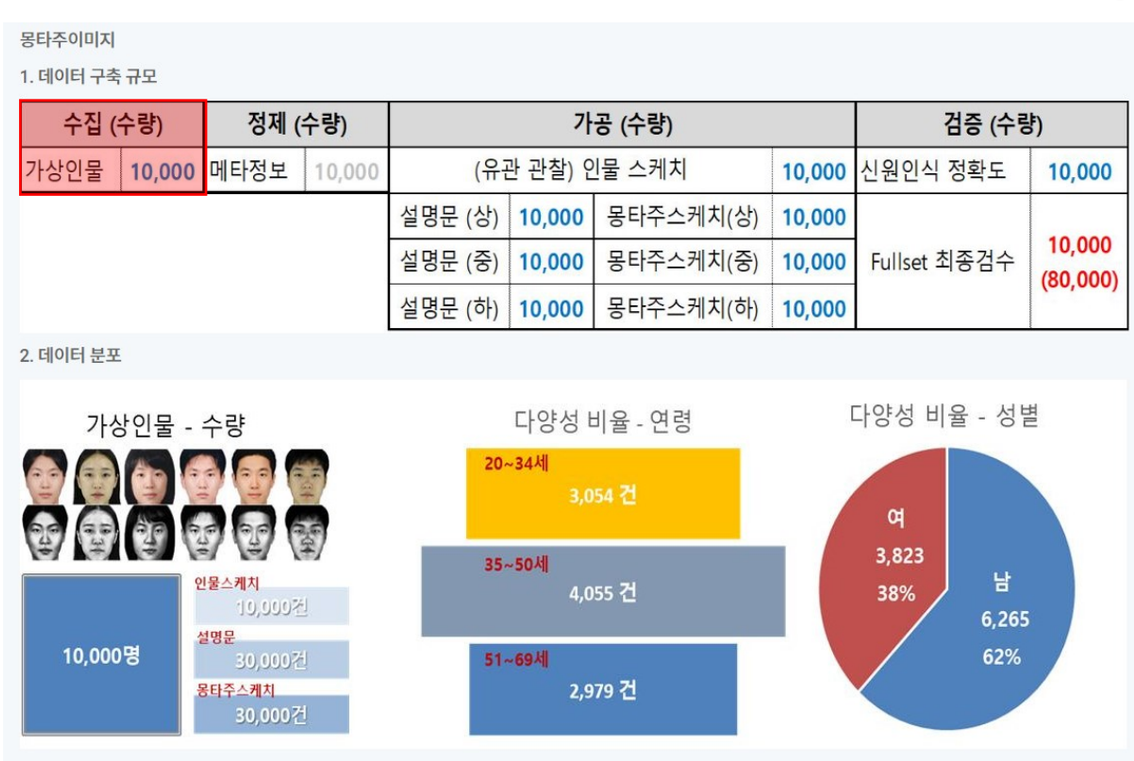


Fig 15. Montage dataset description

```
f, axarr = plt.subplots(2, 3, figsize = (16, 8))
for i in range(2):
    for j in range(3):
        output = G.forward(generate_random_seed(100))
        img = output.detach().cpu().numpy()
        axarr[i, j].imshow(img, interpolation = 'none', cmap = 'Blues')
        pass
    pass
```

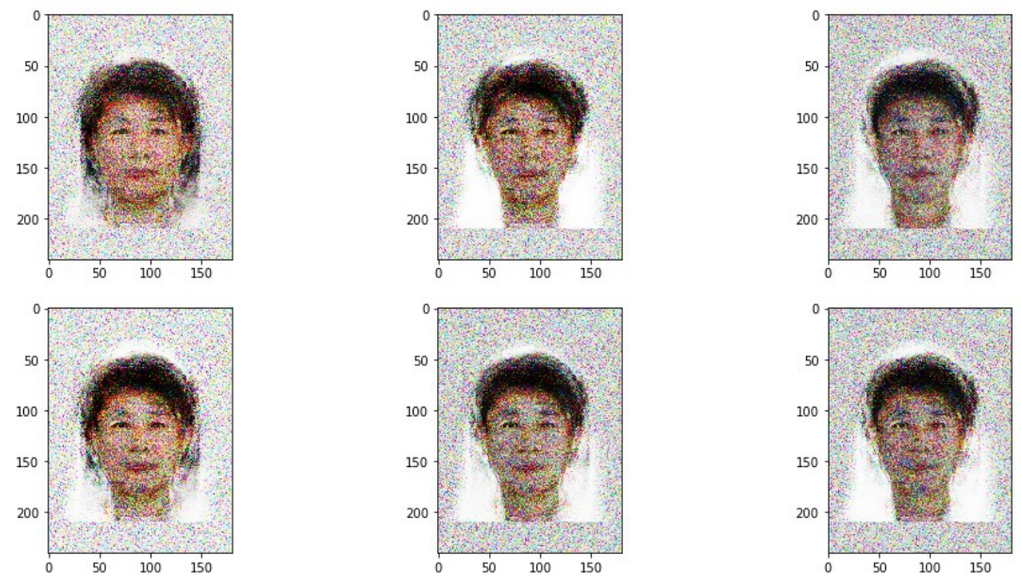


Fig 16. Results of generator

Thank you