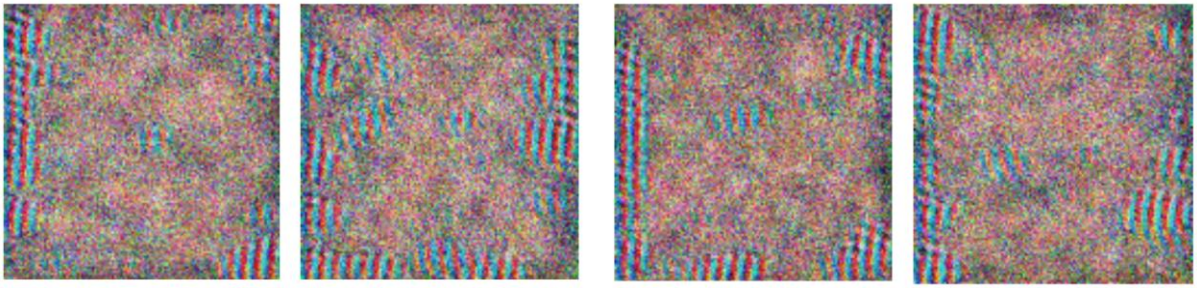
We modified the code from https://github.com/lucidrains/denoising-diffusion-pytorch to provide a hand-on experience to generate fundus photographs using DDPM.

-Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. in *Proceedings of the 34th International Conference on Neural Information Processing Systems* 6840–6851 (2020)

The training with large resolution images (more than 64 x 64 pixels) needs a lot of time. Early stop of training will provide some weird images.



The document was created by tae keun yoo.

CUDA setting

```
!apt-get --purge remove cuda nvidia* libnvidia-*
!dpkg -l | grep cuda- | awk '{print $2}' | xargs -n1 dpkg --purge
!apt-get remove cuda-*
!apt autoremove
!apt-get update
```

```
!wget https://developer.nvidia.com/compute/cuda/9.2/Prod/local_installe
rs/cuda-repo-ubuntu1604-9-2-local_9.2.88-1_amd64 -O cuda-repo-
ubuntu1604-9-2-local_9.2.88-1_amd64.deb
!dpkg -i cuda-repo-ubuntu1604-9-2-local_9.2.88-1_amd64.deb
!apt-key add /var/cuda-repo-9-2-local/7fa2af80.pub
!apt-get update
!apt-get install cuda-9.2
```

```
!pip install git+https://github.com/andreinechaev/nvcc4jupyter.git

!nvcc --version
%load_ext nvcc_plugin
```

```
######## load denoising_diffusion_pytorch ########


pip install denoising_diffusion_pytorch


import torch
from denoising_diffusion_pytorch import Unet, GaussianDiffusion
from denoising_diffusion_pytorch import Trainer
import torchvision.transforms as transforms


model = Unet(
    dim = 64,
    dim_mults = (1, 2, 4, 8)
).cuda()


diffusion = GaussianDiffusion(
    model,
    image_size = 32,
    timesteps = 1000,           # number of steps
    sampling_timesteps = 250,   # number of sampling timesteps (using d
dim for faster inference [see citation for ddim paper])
    loss_type = 'l1'            # L1 or L2
).cuda()



from google.colab import drive
drive.mount('/content/drive')


trainer = Trainer(diffusion, './drive/MyDrive/fundus', train_batch_size
 = 16, train_lr= 2e-
5, train_num_steps = 1000000, gradient_accumulate_every = 1, ema_decay
= 0.995, amp = True)

trainer.train()
```

```
trainer = Trainer(diffusion, './drive/MyDrive/fundus',
                  train_batch_size = 16,
                  train_lr= 2e-5,
                  train_num_steps = 100000,
                  gradient_accumulate_every = 1,
          ema_decay = 0.995,
          amp = True)

trainer.train()
```

```
loss: 0.0855: 5%  4624/100000 [14:17<6:10:45, 4.29it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 24.75it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 23.66it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 23.35it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 25.00it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 22.02it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 23.93it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 23.44it/s]
sampling loop time step: 100%  250/250 [00:10<00:00, 24.00it/s]
```

```
sampled_images = diffusion.sample(batch_size = 100)

### Please modify the number of the images to see the result ####

tf(sampled_images[0])
tf(sampled_images[1])
tf(sampled_images[2])
```

```
[104] sampled_images = diffusion.sample(batch_size = 100)
```

```
sampling loop time step: 100%  250/250 [00:10<00:00, 23.75it/s]
```

```
[105] tf(sampled_images[0])
```

```
[106] tf(sampled_images[1])
```

```
[107] tf(sampled_images[2])
```