Project 5 : Generative Adversarial Networks (Due: Dec. 24)

The goal of this project is generating images with concepts of Generative Adversarial Networks (GAN). First problem is generating images from latent vector and the second problem is image-to-image translation. This project takes a lot of time to training. So we recommend you start this project as soon as possible.

Requirements

1. PyTorch (Version >= 0.4.0) with GPU
2. Torchvision (version >= 0.2)
3. Pillow (version <= 7.0.0)
   (you can downgrade pillow using pip. Ex) pip install pillow==6.2.1)
4. visdom

Data Preparation

1. In Problem #1, you can download dataset with masked faces from URL ([3-6]) or use any other face dataset
2. In Problem #2-2) and Problem #3-2), you can download dataset from pix2pix & CycleGAN github repository ([8])
3. In Problem #2-3) and Problem #3-3), you have to prepare your own dataset and apply image-to-image translation to them

Notice

1. You can use pretrained backbone network.
2. Submit your training and test source and trained models
   (If size is large, upload them to Google Drive and submit the sharing URL)

Problem #1: Training and Testing DC-GAN and Vector Arithmetic

1) Training DC-GAN model

1. Read and understand the DC-GAN [1] using Generator and Discriminator network.
2. Implement the DC-GAN model by following the instruction from github repository ([2]).
3. In GAN, input latent vector z is converted into image $\mathrm{G(z)}$ as it passes through the generator network G. Fig. 1. shows DC-GAN's architecture.

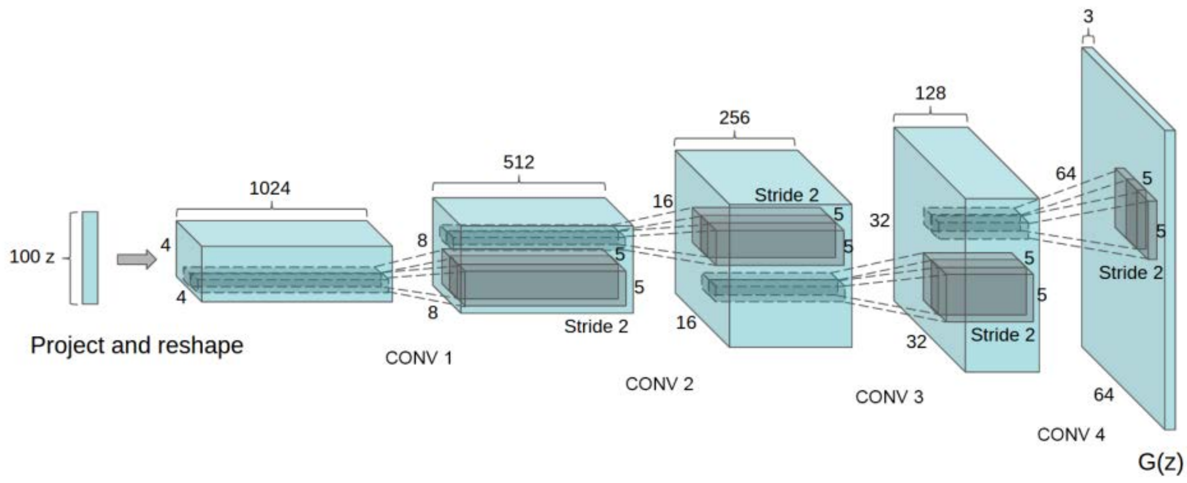

Figure 1. Basic architecture of DC-GAN Generator

After training, generated output images $\mathrm{G(z)}$ depends on the input latent vector z. If a randomly generated vector z_1 generates image of man wearing sunglasses, we can consider that z_1 contains information of man wearing sunglasses. Similarly, if a vector z_2 generates man without sunglasses, z_2 is containing information of man without sunglasses.
With this basis, we can generate images we want using input specific latent vectors. For example, as shown in Fig. 2, assume we have found three z_1, z_2, z_3 vectors that generate man with sunglasses ($\mathrm{G(z_1)}$), man without sunglasses ($\mathrm{G(z_2)}$) and woman without sunglasses ($\mathrm{G(z_3)}$).

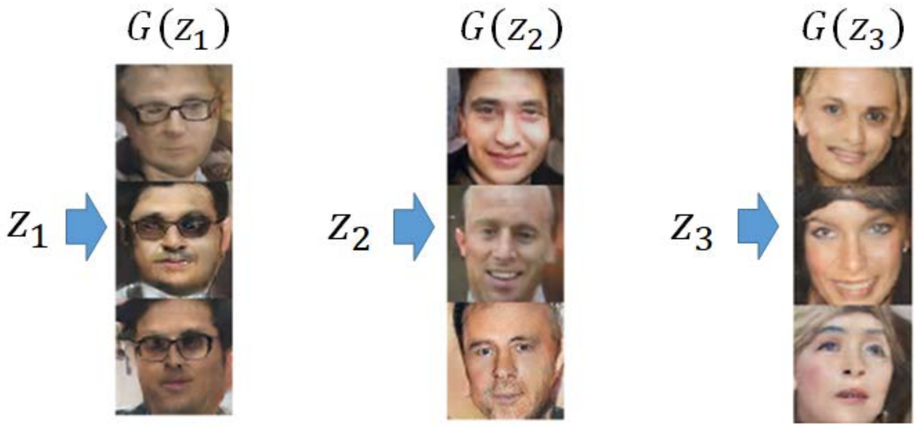

Figure 2. Example of three latent vectors and their results

Then, we can generate woman with sunglasses through the computation of those vectors. Fig. 3. shows the example.

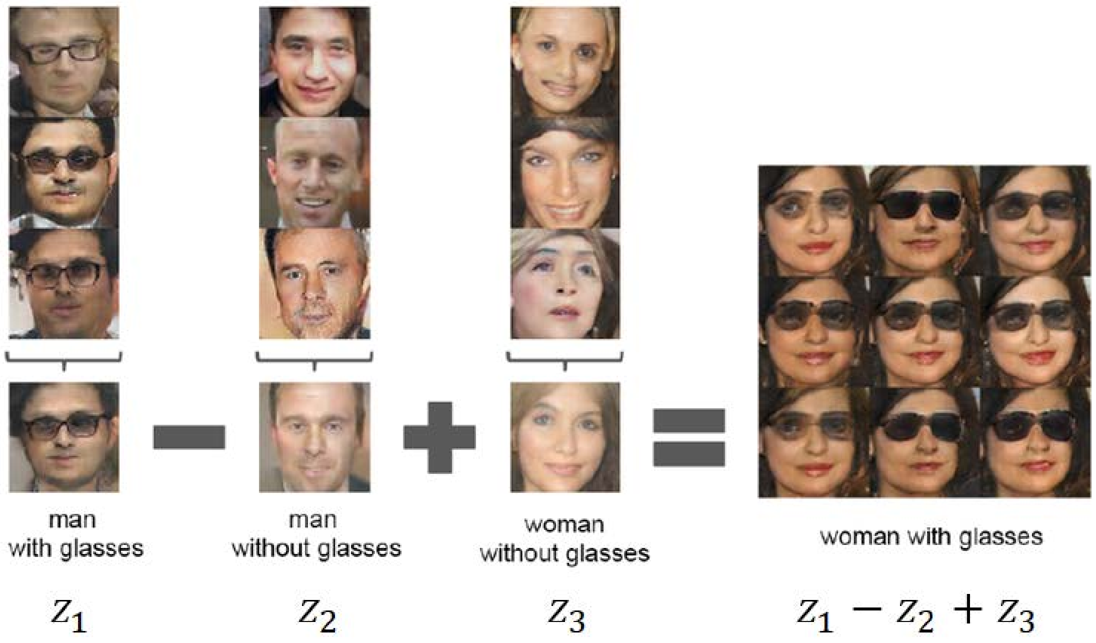

Figure 3. Example of vector arithmetic in DC-GAN.

Now, our goal of problem #1 is applying this to generation of woman wearing mask as described in Fig. 4. Train the DC-GAN model with face images and generate the goal image. Write the process of performing this project and your efforts in the report.

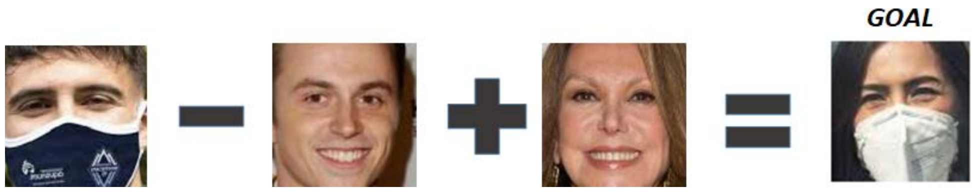

Figure 4. Our goal: generating woman wearing mask from $\mathbf{z}$ vectors

+) You can use any dataset or combine several different dataset. Just make sure to prepare images containing man, woman, wearing mask and without mask. We can give you some examples of available dataset.

- Images with wearing mask and without mask : [3]
- Images with wearing mask : [4-5]
- Or any other face dataset (For example, CelebA, LFW ....)

2) Report experimental results

1. Try the efforts to improve the performance on your network. For example, your hyper-parameter setting or collecting dataset or your network improvements that are not provided by the basic codes.
2. Some result images including generated images using DC-GAN.
3. What did you learn through this problem #1.
4. Discuss about the experimental results, network architecture, and training techniques.

Problem #2: Training and Testing Paired Image-to-Image Translation

1) Training Pix2pix model

1. Read and understand the pix2pix [6] using Generator and Discriminator network.
2. Implement the pix2pix model by following the instruction from github repository ([7]).
3. Train and Test the pix2pix network with Facade Dataset. You can download the dataset from that github repository. Fig. 5. Shows example of Facades Dataset
   (You can modify parameters in options/train_options.py)

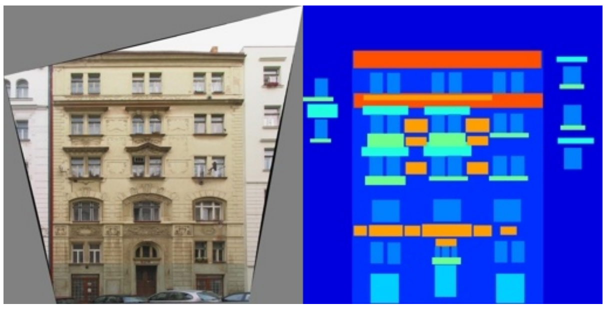

Figure 5. Example of Facade Dataset

2) Report experimental results

1. Try the efforts to improve the performance on your network. For example, your hyper-parameter setting or your network improvements that are not provided by the basic codes.
2. Some result images including generated images using pix2pix.
3. What did you learn through this problem #2.
4. Discuss about the experimental results, network architecture, and training techniques.

3) Create your own idea and show the implementation results

1. Collect the dataset for your idea.
2. Implement the code that realizes your idea.
3. Demonstrate the implementation results.
4. Discuss about your achievement from Problem #2-3).

Problem #3: Training and Testing Unpaired Image-to-Image Translation

1) Training CycleGAN model

1. Read and understand the CycleGAN [8] using Generator and Discriminator network.
2. Implement the CycleGAN by following the instruction from github repository ([7]).
3. Train and Test the CycleGAN network with Horse-to-Zebra Dataset. You can download horse-to-zebra dataset from that github repository. Fig. 6. Shows example of Horse to Zebra image translation.

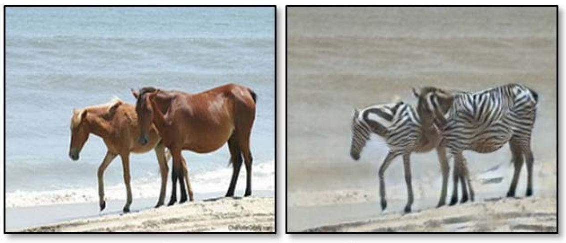

Figure 6. Example of Horse-to-Zebra Dataset using CycleGAN

2) Report experimental results

1. Try the efforts to improve the performance on your network. For example, your hyper-parameter setting or your network improvements that are not provided by the basic codes.
2. Some result images including generated images using CycleGAN.
3. What did you learn through this problem #3.
4. Discuss about the experimental results, network architecture, and training techniques.

3) Create your own idea and show the implementation results

1. Collect the dataset for your idea. Fig. 7. shows example results using various custom background images.

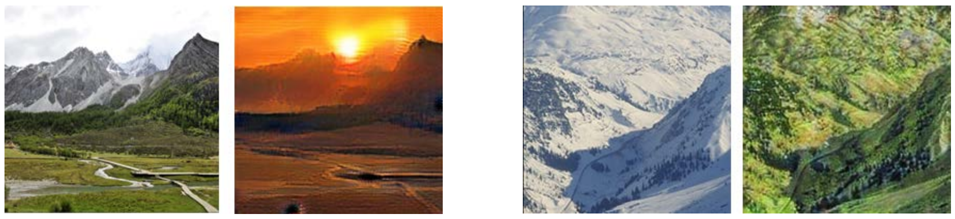

Figure 7. Example of Custom Dataset. Left images are example of mountain->sunset and right images are example of snow mountain->green mountain

2. Implement the code that realizes your idea.
3. Demonstrate the implementation results.
4. Discuss about your achievement from Problem #3-3).

References

[1] Alee Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In ICIR, 2015.

[2] https://github.com/pytorch/examples/tree/master/dcgan

[3] https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset/activity

[4] https://public.roboflow.com/object-detection/mask-wearing/4/download/coco

[5] https://www.kaggle.com/andrewmvd/face-mask-detection

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In ICCV, 2017.

[7] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In ICCV, 2017.