ENGR-UH 4560

Selected Topics in Information and Computational Systems

Deep Learning Project - GAN-Human Face Synthesis

Due Date: Refer to NYU Class

Introduction

Since GANs were introduced in 2014 by Google Researcher Ian Goodfellow, the tech has been widely adopted in image generation and transfer. After some early wiry failures, GANs have made huge breakthroughs and can now produce highly convincing fake images of animals, landscapes, human faces, etc. Researchers know what GANs can do, however a lack of transparency in their inner workings means GAN improvement is still achieved mainly through trial-and-error. This allows only limited control over the synthesized images.

Dataset

You may find the dataset in the following link: CelebA
CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than **200K** celebrity images, each with **40** attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including

- **10,177** number of **identities**
- **202,599** number of **face images**
- **5 landmark locations**, **40 binary attributes** annotations per image.

The dataset can be employed as the training and test sets for the following computer vision tasks: face attribute recognition, face detection, and landmark (or facial part) localization.

Requirements

1. Program a data loader for CelebA dataset:
    a. The data loader needs to be established by *torch.utils.data.DataLoader*
2. Built a generator model:
    a. The model should be established based on *torch.nn.Module* and *torch.nn.functional*
    b. The model (base model) need consist of linear layers, RELU and Tanh
    c. nn.Sequential is recommended to pack all modules in the model
    d. Please First construct the generator module as follow:
        i. The input of the first linear layer is the latent vector size, output of the first layer is 256
        ii. Then followed by a ReLU layer.
        iii. The input of the second layer is the 256 and output channel is 512
        iv. Followed by the ReLU layer
        v. The input of the third layer is the 512 and output channel is 1024
        vi. Followed by the ReLU layer
        vii. The input of the fourth layer is the 1024 and output channel is 1024
        viii. Followed by the ReLU layer
        ix. The input of the final layer is the 1024 and output channel is the image size.
        x. The Tanh is activation function.

   e. You can add convolution layers and other modules in the base model to push performance, but improvement should be written in your report.

3. Built a discriminator model:
   a. The model should be established based on *torch.nn.Module* and *torch.nn.functional*
   b. The model (base model) need consist of Linear layers, LeakRELU and Sigmoid layer
   c. nn.Sequential is recommended to pack all modules in the model
   d. Please First construct the discriminator module as follow:
      i. The input of the first linear layer is the image size, output of the first layer is 256
      ii. Then followed by a LeakyReLU layer.
      iii. The input of the second layer is the 256 and output channel is 512
      iv. Followed by the leakyReLU layer
      v. The input of the third layer is the 512 and output channel is 512
      vi. Followed by the leakyReLU layer
      vii. The input of the final layer is the 512 and output channel is 1
      viii. The sigmoid is activation function.
   e. Afte that ,you can add convolution layers and other modules in the base model to push performance, but improvement should be written in your report.
4. Write GAN frameworks to implement the classification pipeline:
   a. Apply the dataloader you have written in the first step
   b. Create two optimizer (SGD/Adam) for the discriminator and the generator respectively
   c. Train the discriminator and the generator alternately
   d. Define a Binary Cross Entropy loss for your implement
   e. Print out the fake image prediction score and real image prediction score of your framework.
5. Plot out the loss convergence course by Tensorboard or lera.ai or matplotlib.pyplot

Deliverables

A .ipynb file containing the following:

1. source code
2. loss convergence plots
3. detailed description to explain your code if needed

Before submitting your project, please make sure to test your program on the given dataset.

Notes

You may discuss the general concepts in this project with other students, but you must implement the program on your own. ***No sharing of code or report is allowed.*** *Violation of this policy can result in a grade penalty.*

Late submission is acceptable with the following penalty policy:
10 points deduction for every day after the deadline