Assignment 6 : Generative Adversarial Nets (GANs)

In this assignment, you will implement and train GAN models for 2D data (anime faces) and 3D data (IKEA objects).

1. Getting Started

You need to modify the basic DCGAN starter code (**XX_dcgan.ipynb** and **XX_3dgan.ipynb**) or implement from scratch for 2D and 3D data generation.
The starter code is implemented in PyTorch, if you want to try TensorFlow, please check out TensorFlow official DCGAN tutorial .

2. Datasets

- Please download anime faces data for 2D GAN in Kaggle. This dataset has 63,565 well aligned and cropped anime faces.
- Please download data for 3D GAN in 3D shapes. It contains different classes of objects, e.g. 10,668 chair objects in **.mat** format in **3DShapeNets -> volumetric_data -> chair**.

3. Summary of Requirements

You are required to generate novel images and display them in the report (README.pdf):

- Train a 2D GAN for random anime face generation in 64x64, and generate 64 (8x8) results, displaying them in report.
- Train a 3D GAN for random object generation in 64x64x64, and generate some results, displaying them in report.

4. Implementation Notes

4.1. Generator

In the class **Generator**, you are required to implement 2D de-convolutional layers for the Generator in DCGAN. You are required to implement 3D de-convolutional layers for the Generator in 3DGAN according to 3DGAN paper.

4.2. Discriminator

In the class **Discriminator**, you are required to implement 2D convolutional layers for the Discriminator DCGAN. You are required to implement 3D convolutional layers for the Discriminator in 3DGAN according

to 3DGAN paper.

4.3. Inputs

You can feel free to change training hyperparamster for better performance.

You can change data loader, apply different data preprocessing (e.g. resizing) as you like.

4.4. Dataloader

We use default dataloader for image loading, and have a customized dataloader for 3D data loading. You can also change and customize a new dataloader if you need.

4.5. Loss Criteria

Here we use GAN adversarial loss only. Please implement it with BCELoss in training loop.

7. Extra Credits

Here are some extensions of basic GANs to explore more features of GAN generation.

7.1 Easy

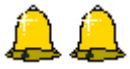
Artifacts Discovery: can you discover any shortcomings and artifacts in your generation? Please display some samples. For example, GANs have well-known Model Collapse Issue where GANs tend to produce the same output (or a small set of outputs) over and over again.

Try to remedy Mode Collapse Issue by applying "Wasserstein loss"

Please apply any data augmentation during training, and see if there is any model performance improvement. For implementation, please refer to Data augmentation in TensorFlow, or Data augmentation in PyTorch https://pytorch.org/vision/main/transforms.html

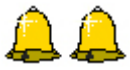
7.2 Medium

Implement ProgressiveGAN for better 2D generation performance, please show anime generation results compared to DCGAN.



Implement Pix2Pix for image conversion. Please download paired data for Pix2Pix training in pix2pix_maps_dataset. This dataset contains 1,096 paired map images (satellite and transit maps) for training, and 1,098 pairs for validation. Note that image pairs have been concatenated (left: satellite map, right: transit map).

Evaluation for Pix2Pix. If you successfully implement Pix2Pix, please evaluate the valuation set (1,098 pairs) with metrics FID, L1 loss.



Implement complete 3DGAN with extended 3D-VAE-GAN and additinoal proposed losses, utilizing both 2D and 3D data. See details in 3DGAN Porject Page.

7.3 Hard

 Implement CycleGAN for map image conversion, and compare the generation quality with Pix2Pix.

 Implement upgraded version of CycleGAN or Pix2Pix for 2D image to 3D object conversion using the same 3D dataset or other datasets you can access.

8. Submission

You are to write a README.pdf that shows 2D generation results, and answers the following questions:

- Did you collaborate with anyone in the class? If so, let us know who you talked to and what sort of help you gave or received.
- Were there any references (books, papers, websites, etc.) that you found particularly helpful for completing your assignment? Please provide a list.
- Are there any known problems with your code? If so, please provide a list and, if possible, describe what you think the cause is and how you might fix them if you had more time or motivation. *This is very important, as we're much more likely to assign partial credit if you help us understand what's going on.*
- Did you do any extra credit? If so, let us know how to use the additional features. If there was a substantial amount of work involved, describe what and how you did it.
- Got any comments about this assignment that you'd like to share? Was it too long? Too hard? were the requirements unclear? Did you have fun, or did you hate it? Did you learn something, or was it a total waste of your time? Feel free to be brutally honest; we promise we won't take it personally.

Submit your assignment online. Please submit a single archive (.zip or .tar.gz) containing:

- Your source code.
- 2D and 3D generated files.
- The README file.
- Any additional files.