Open in Colab

T81-558: Applications of Deep Neural Networks

Module 7: Generative Adversarial Networks

- Instructor: Jeff Heaton, McKelvey School of Engineering, Washington University in St. Louis
- For more information visit the class website.

Module 7 Material

- Part 7.1: Introduction to GANs for Image and Data Generation [Video] [Notebook]
- **Part 7.2: Train StyleGAN3 with your Own Images** [Video] [Notebook]
- Part 7.3: Exploring the StyleGAN Latent Vector [Video] [Notebook]
- Part 7.4: GANs to Enhance Old Photographs Deoldify [Video] [Notebook]
- Part 7.5: GANs for Tabular Synthetic Data Generation [Video] [Notebook]

Google CoLab Instructions

The following code ensures that Google CoLab is running the correct version of TensorFlow.

In [ ]:
```
try:
    from google.colab import drive
    drive.mount('/content/drive', force_remount=True)
    COLAB = True
    print("Note: using Google CoLab")
    %tensorflow_version 2.x
except:
    print("Note: not using Google CoLab")
    COLAB = False
```

Part 7.2: Train StyleGAN3 with your Images

Training GANs with StyleGAN is resource-intensive. The NVIDA StyleGAN researchers used computers with eight high-end GPUs for the high-resolution face GANs trained by NVIDIA. The GPU used by NVIDIA is an A100, which has more memory and cores than the P100 or V100 offered by even Colab Pro+. In this part, we will use StyleGAN2 to train rather than StyleGAN3. You can use networks trained with StyleGAN2 from StyleGAN3; however, StyleGAN3 usually is more effective at training than StyleGAN2.

Unfortunately, StyleGAN3 is compute-intensive and will perform slowly on any GPU that is not the latest Ampere technology. Because Colab does not provide such technology, I am keeping the training guide at the StyleGAN2 level. Switching to StyleGAN3 is relatively easy, as will be pointed out later.

Make sure that you are running this notebook with a GPU runtime. You can train GANs with either Google Colab Free or Pro. I recommend at least the Pro version due to better GPU instances, longer runtimes, and timeouts. Additionally, the capability of Google Colab Pro to run in the background is valuable when training GANs, as you can close your browser or reboot your laptop while training continues.

You will store your training data and trained neural networks to GDRIVE. For GANs, I lay out my GDRIVE like this:

- ./data/gan/images - RAW images I wish to train on.
- ./data/gan/datasets - Actual training datasets that I convert from the raw images.
- ./data/gan/experiments - The output from StyleGAN2, my image previews, and saved network snapshots.

You will mount the drive at the following location.

```
/content/drive/MyDrive/data
```

What Sort of GPU do you Have?

The type of GPU assigned to you by Colab will significantly affect your training time. Some sample times that I achieved with Colab are given here. I've found that Colab Pro generally starts you with a V100, however, if you run scripts non-stop for 24hrs straight for a few days in a row, you will generally be throttled back to a P100.

- 1024x1024 - V100 - 566 sec/tick (CoLab Pro)
- 1024x1024 - P100 - 1819 sec/tick (CoLab Pro)
- 1024x1024 - T4 - 2188 sec/tick (CoLab Free)

By comparison, a 1024x1024 GAN trained with StyleGAN3 on a V100 is 3087 sec/tick.

If you use Google CoLab Pro, generally, it will not disconnect before 24 hours, even if you (but not your script) are inactive. Free CoLab WILL disconnect a perfectly good running script if you do not interact for a few hours. The following describes how to circumvent this issue.

- How to prevent Google Colab from disconnecting?

Set Up New Environment

You will likely need to train for >24 hours. Colab will disconnect you. You must be prepared to restart training when this eventually happens. Training is divided into ticks, every so many ticks (50 by default), your neural network is evaluated, and a snapshot is saved. When CoLab shuts down, all training after the last snapshot is lost. It might seem desirable to snapshot after each tick; however, this snapshotting process itself takes nearly an hour. Learning an optimal snapshot size for your resolution and training data is important.

We will mount GDRIVE so that you will save your snapshots there. You must also place your training images in GDRIVE.

You must also install NVIDIA StyleGAN2 ADA PyTorch. We also need to downgrade PyTorch to a version that supports StyleGAN.

In [ ]:
```
!pip uninstall jax jaxlib -y
!pip install "jax[cuda11_cudnn805]==0.3.10" -f https://storage.googleapis
!pip install torch==1.8.1 torchvision==0.9.1
!git clone https://github.com/NVlabs/stylegan2-ada-pytorch.git
!pip install ninja
```

Find Your Files

The drive is mounted to the following location.

```
/content/drive/MyDrive/data
```

It might be helpful to use an `ls` command to establish the exact path for your images.

In [ ]:
```
!ls /content/drive/MyDrive/data/gan/images
```

Convert Your Images

You must convert your images into a data set form that PyTorch can directly utilize. The following command converts your images and writes the resulting data set to another directory.

In [ ]:
```
CMD = "python /content/stylegan2-ada-pytorch/dataset_tool.py "\
  "--source /content/drive/MyDrive/data/gan/images/circuit "\
  "--dest /content/drive/MyDrive/data/gan/dataset/circuit"

!{CMD}
```

You can use the following command to clear out the newly created dataset. If something goes wrong and you need to clean up your images and rerun the above command, you should delete your partially completed dataset directory.

In [ ]:
```
#!rm -R /content/drive/MyDrive/data/gan/dataset/circuit/*
```

Clean Up your Images

All images must have the same dimensions and color depth. This code can identify images that have issues.

In [ ]:
```
from os import listdir
from os.path import isfile, join
import os
from PIL import Image
from tqdm.notebook import tqdm

IMAGE_PATH = '/content/drive/MyDrive/data/gan/images/fish'
files = [f for f in listdir(IMAGE_PATH) if isfile(join(IMAGE_PATH, f))]

base_size = None
for file in tqdm(files):
  file2 = os.path.join(IMAGE_PATH,file)
  img = Image.open(file2)
  sz = img.size
  if base_size and sz!=base_size:
    print(f"Inconsistant size: {file2}")
  elif img.mode!='RGB':
    print(f"Inconsistant color format: {file2}")
  else:
    base_size = sz
```

Perform Initial Training

This code performs the initial training. Set SNAP low enough to get a snapshot before Colab forces you to quit.

In [ ]:
```
import os

# Modify these to suit your needs
EXPERIMENTS = "/content/drive/MyDrive/data/gan/experiments"
DATA = "/content/drive/MyDrive/data/gan/dataset/circuit"
SNAP = 10

# Build the command and run it
cmd = f"/usr/bin/python3 /content/stylegan2-ada-pytorch/train.py "\
  f"--snap {SNAP} --outdir {EXPERIMENTS} --data {DATA}"
!{cmd}
```

Resume Training

You can now resume training after you are interrupted by something in the pervious step.

In [ ]:
```
import os

# Modify these to suit your needs
EXPERIMENTS = "/content/drive/MyDrive/data/gan/experiments"
NETWORK = "network-snapshot-000100.pkl"
RESUME = os.path.join(EXPERIMENTS, \
                "00008-circuit-auto1-resumecustom", NETWORK)
DATA = "/content/drive/MyDrive/data/gan/dataset/circuit"
SNAP = 10

# Build the command and run it
cmd = f"/usr/bin/python3 /content/stylegan2-ada-pytorch/train.py "\
  f"--snap {SNAP} --resume {RESUME} --outdir {EXPERIMENTS} --data {DATA}
!{cmd}
```