Latent Space Exploration

```
# Mount G-Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

Clone StyleGAN2 ADA PyTorch from GitHub.

```
!git clone https://github.com/NVlabs/stylegan2-ada-pytorch.git
!pip install ninja
```

```
Cloning into 'stylegan2-ada-pytorch'...
remote: Enumerating objects: 125, done.
remote: Total 125 (delta 0), reused 0 (delta 0), pack-reused 125
Receiving objects: 100% (125/125), 1.12 MiB | 17.95 MiB/s, done.
Resolving deltas: 100% (55/55), done.
Collecting ninja
  Downloading ninja-1.10.2.2-py2.py3-none-manylinux_2_5_x86_64.manylinux1_x86_64
     |████████████████████████████████| 108 kB 8.1 MB/s
Installing collected packages: ninja
Successfully installed ninja-1.10.2.2
```

Verify that StyleGAN has been cloned.

```
!ls /content/stylegan2-ada-pytorch/
```

```
calc_metrics.py  docker_run.sh  LICENSE.txt   style_mixing.py
dataset_tool.py  docs           metrics       torch_utils
dnnlib           generate.py    projector.py  training
Dockerfile       legacy.py      README.md     train.py
```

Run StyleGAN2 From Python Code

The code below is based on code from NVIDIA. This will generate the images.

```
import sys
sys.path.insert(0, "/content/stylegan2-ada-pytorch")
import pickle
import os
import numpy as np
import PIL.Image
from IPython.display import Image
```

```
import matplotlib.pyplot as plt
import IPython.display
import torch
import dnnlib
import legacy

def seed2vec(G, seed):
  return np.random.RandomState(seed).randn(1, G.z_dim)

def display_image(image):
  plt.axis('off')
  plt.imshow(image)
  plt.show()

def generate_image(G, z, truncation_psi):
    # Render images for dlatents initialized from random seeds.
    Gs_kwargs = {
        'output_transform': dict(func=tflib.convert_images_to_uint8, nchw_to_nhwc=True
        'randomize_noise': False
    }
    if truncation_psi is not None:
        Gs_kwargs['truncation_psi'] = truncation_psi

    label = np.zeros([1] + G.input_shapes[1][1:])
    images = G.run(z, label, **G_kwargs) # [minibatch, height, width, channel]
    return images[0]

def get_label(G, device, class_idx):
  label = torch.zeros([1, G.c_dim], device=device)
  if G.c_dim != 0:
      if class_idx is None:
          ctx.fail('Must specify class label with --class when using a conditional net
      label[:, class_idx] = 1
  else:
      if class_idx is not None:
          print ('warn: --class=lbl ignored when running on an unconditional network')
  return label

def generate_image(device, G, z, truncation_psi=1.0, noise_mode='const', class_idx=Non
  z = torch.from_numpy(z).to(device)
  label = get_label(G, device, class_idx)
  img = G(z, label, truncation_psi=truncation_psi, noise_mode=noise_mode)
  img = (img.permute(0, 2, 3, 1) * 127.5 + 128).clamp(0, 255).to(torch.uint8)
  #PIL.Image.fromarray(img[0].cpu().numpy(), 'RGB').save(f'{outdir}/seed{seed:04d}.png
  return PIL.Image.fromarray(img[0].cpu().numpy(), 'RGB')


URL = "/content/drive/MyDrive/grappleGAN/results/00006-bjj1024-mirror-auto1-resumecust

print(f'Loading networks from "{URL}"...')
device = torch.device('cuda')
with dnnlib.util.open_url(URL) as f:
```

```
    G = legacy.load_network_pkl(f)['G_ema'].to(device) # type: ignore

    Loading networks from "/content/drive/MyDrive/grappleGAN/results/00006-bjj1024-m
```

Preview ranges of fakes to choose subjects for interpolation video

```
# Choose starting and ending seed.
SEED_FROM = 7000
SEED_TO = 7040

# Generate the images for the seeds.
for i in range(SEED_FROM, SEED_TO):
  print(f"Seed {i}")
  z = seed2vec(G, i)
  img = generate_image(device, G, z)
  display_image(img)
```

Seed 7000

Seed 7001

Seed 7002

Seed 7003

Seed 7004

Seed 7005

Seed 7006

Seed 7007

Seed 7008

Seed 7009

Seed 7010

Seed 7011

Seed 7012

Seed 7013

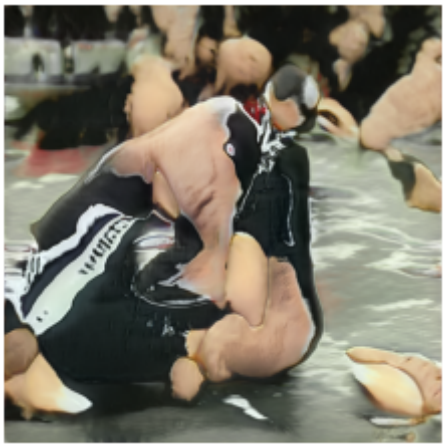
Seed 7014

Seed 7015

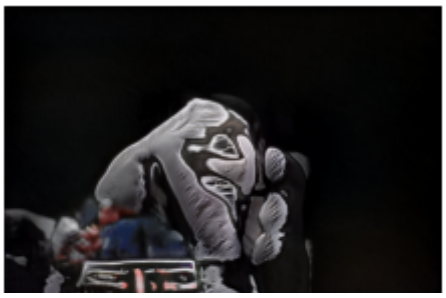
Seed 7016

Seed 7017

Seed 7018



Seed 7019



Seed 7020

Seed 7021

Seed 7022

Seed 7023

Seed 7024

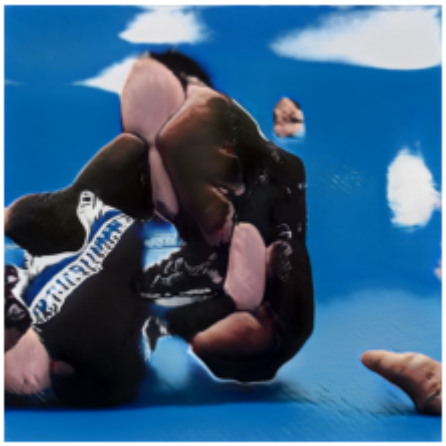
Seed 7025

Seed 7026

Seed 7027

Seed 7028

Seed 7029



Seed 7030

Seed 7031

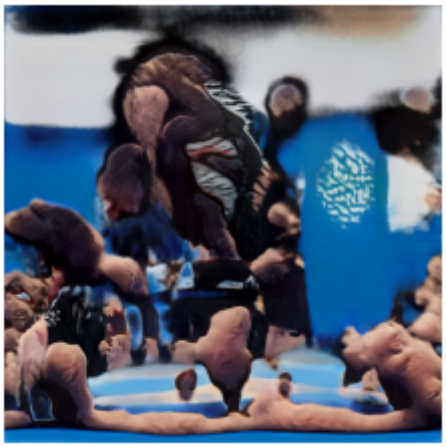
Seed 7032

Seed 7033

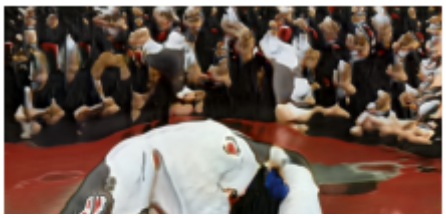
Seed 7034

Seed 7035

```
Seed 7036
```



```
Seed 7037
```



Preview list of chosen fakes for interpolation video

```
# Choose list of seeds
SEED_LIST = [7030, 7028, 700, 7039, 201, 107, 2018, 607, 70024]

# Generate the images for the seeds.
for i in SEED_LIST:
  print(f"Seed {i}")
  z = seed2vec(G, i)
  img = generate_image(device, G, z)
  display_image(img)
```