StyleGan2 설치 및 실행

(참조 : https://github.com/NVlabs/stylegan2)

(https://github.com/NVlabs/stylegan2-ada, https://github.com/NVlabs/ffhq-dataset)

StyleGAN: A Style-Based Generator Architecture for Generative Adversarial Networks

[1] 그래픽 카드 및 컴파일 관련 S/W 설치

Ram 16 Giga 이상

NVIDIA GPU 카드 설치 (16Giga 이상, 가능하면 여러개의 GPU 권장)

MS Visual Studion 2029 Communication 버전(무료) 설치 : source 컴파일을 위하여 필요

NVIDIA CUDA TOOLKIT(v11.4) 설치 : GPU사용과 Nvidia 컴파일러(nvcc.exe)를 위하여 필요

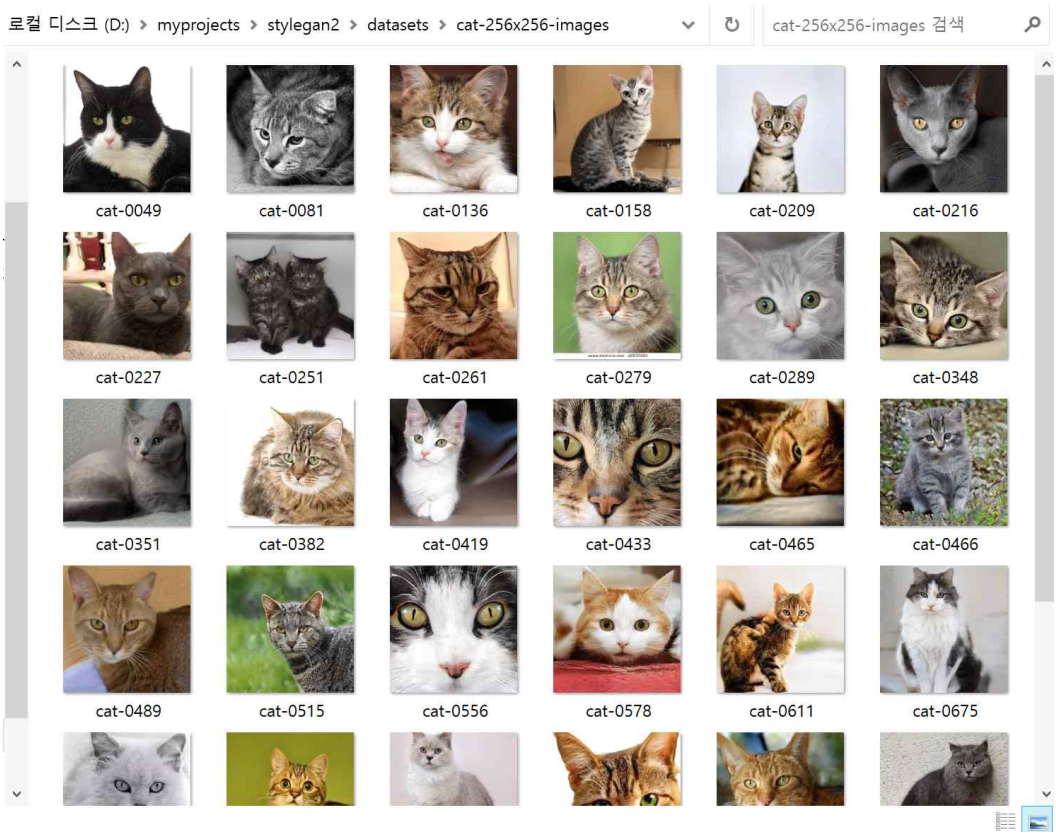
[2] 가상환경 생성 및 관련 패키지 설치

```
conda create -n stylegan2 git
conda activate stylegan2
                                                    # tensorflow 1.15, 2.x 버전은 호환 안됨
conda install tensorflow-gpu=1.14.0 keras scipy requests pillow opencv
```

[3] StyleGan2 소스 및 학습용 데이터 다운로드

```
git clone https://github.com/NVlabs/stylegan2.git
cd stylegan2
```

 # 강의 사이트에서 CAT data(cat-256x256.zip) 다운받아서 stylegan2\datasets\cat-256x256-images 폴더 아래에 압축해제 및 저장



[4] 컴파일 환경 설정

CUDA nvcc 컴파일러 위치를 운영체제 Path 환경변수에 반영

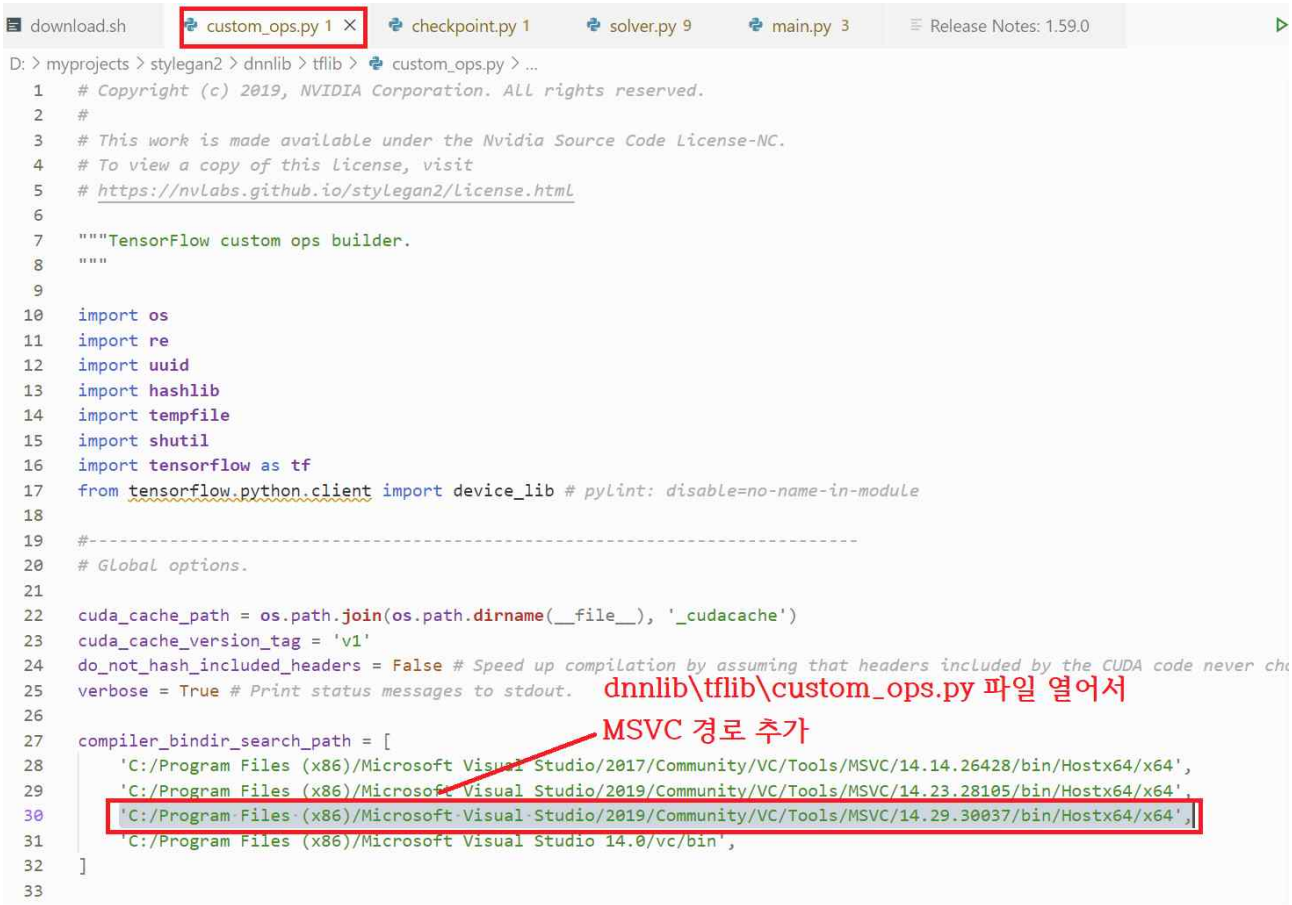
```
setx path=%path%;C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.4\bin
```

nvcc 동작 확인

```
nvcc --version
```

MSVC 컴파일러 경로를 custom_ops.py 파일에 반영

D:\myprojects\stylegan2\dnnlib\tflib\custom_ops.py 파일 수정




```
# Copyright (c) 2019, NVIDIA Corporation. All rights reserved.
#
# This work is made available under the Nvidia Source Code License-NC.
# To view a copy of this license, visit
# https://nvlabs.github.io/stylegan2/license.html

"""TensorFlow custom ops builder.
"""

import os
import re
import uuid
import hashlib
import tempfile
import shutil
import tensorflow as tf
from tensorflow.python.client import device_lib # pylint: disable=no-name-in-module

#----------------------------------------------------------------------------
# Global options.

cuda_cache_path = os.path.join(os.path.dirname(__file__), '_cudacache')
cuda_cache_version_tag = 'v1'
do_not_hash_included_headers = False # Speed up compilation by assuming that headers included by the CUDA code never ch
verbose = True # Print status messages to stdout.

compiler_bindir_search_path = [
    'C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.14.26428/bin/Hostx64/x64',
    'C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/VC/Tools/MSVC/14.23.28105/bin/Hostx64/x64',
    'C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/VC/Tools/MSVC/14.29.30037/bin/Hostx64/x64',
    'C:/Program Files (x86)/Microsoft Visual Studio 14.0/vc/bin',
]
```

[5] Custom Dataset 생성과 multi-resolution TFRecords 생성

(TF레코드 설명 :https://bcho.tistory.com/1190)

TFRecord 파일 포맷이란

TFRecord 파일은 텐서플로우의 학습 데이타 등을 저장하기 위한 바이너리 데이타 포맷으로, 구글의 Protocol Buffer 포맷으로 데이타를 파일에 Serialize 하여 저장한다.
CSV 파일에서와 같이 숫자나 텍스트 데이타를 읽을때는 크게 지장이 없지만, 이미지를 데이타를 읽을 경우 이미지는 JPEG나 PNG 형태의 파일로 저장되어 있고 이에 대한 메타 데이타와 라벨은 별도의 파일에 저장되어 있기 때문에, 학습 데이타를 읽을때 메타데이타나 라벨용 파일 하나만 읽는 것이 아니라 이미지 파일도 별도로 읽어야 하기 때문에, 코드가 복잡해진다.

또한 이미지를 JPG나 PNG 포맷으로 읽어서 매번 디코딩을 하게 되면, 그 성능이 저하되서 학습단계에서 데이타를 읽는 부분에서 많은 성능 저하가 발생한다.

이와 같이 성능과 개발의 편의성을 이유로 TFRecord 파일 포맷을 이용하는 것이 좋다.

Custom dataset 생성

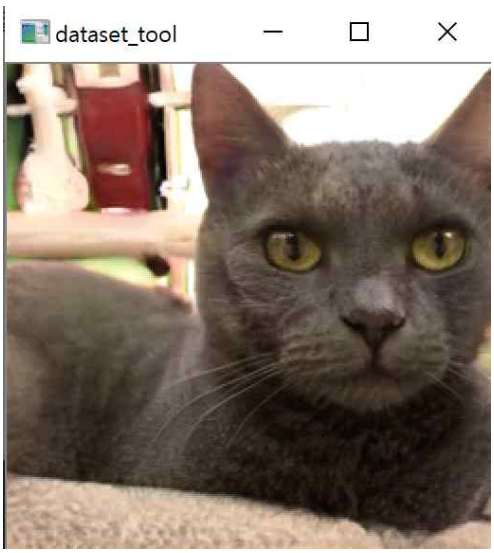
(원본데이터 이미지 폴더: datasets/cat-256x256-images,
생성될 dataset 폴더: datasets/cat-256x256-datasets)

```
python dataset_tool.py create_from_images
        datasets/cat-256x256-datasets datasets/cat-256x256-images
```

DataSet 으로부터 이미지 Display 테스트

```
python dataset_tool.py display datasets/cat-256x256-datasets
```

```
Loading dataset "datasets/cat-256x256-datasets"
Displaying images
Press SPACE or ENTER to advance, ESC to exit

idx = 0
label = []

idx = 1
label = []

Displayed 2 images.
```



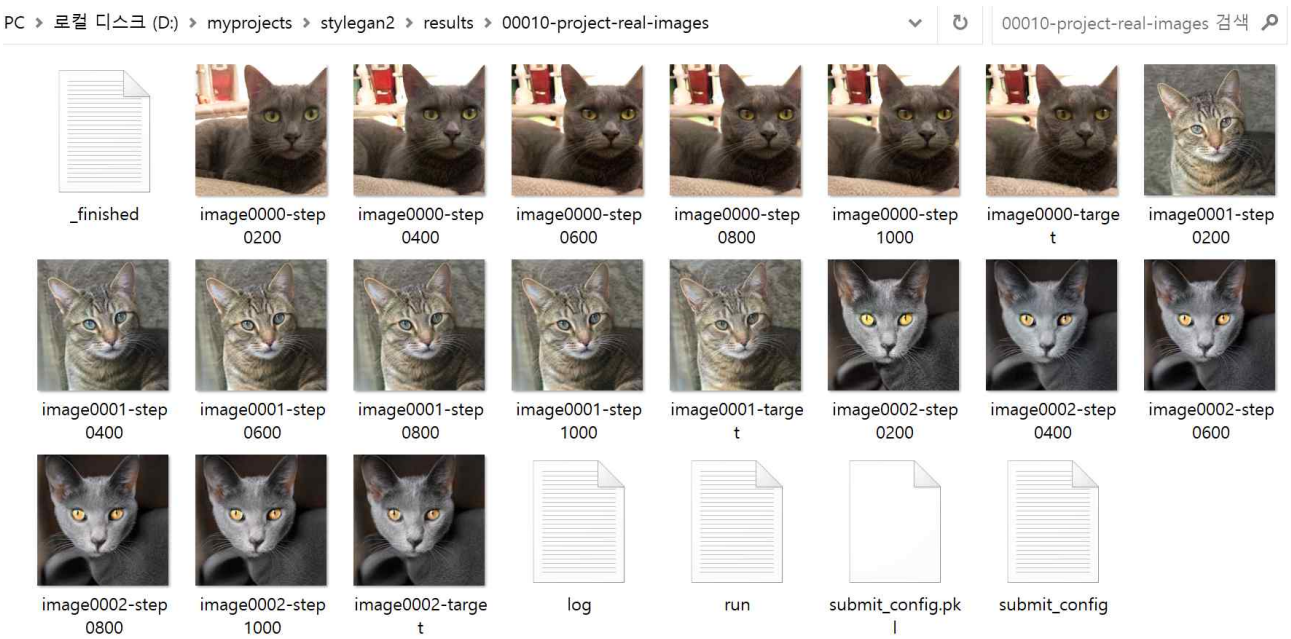
[6] Latent Space로 Images Projection 하기

Project real images

```
python run_projector.py project-real-images
        --network=gdrive:networks/stylegan2-cat-config-f.pkl
        --dataset=cat-256x256-datasets --data-dir=./datasets
```

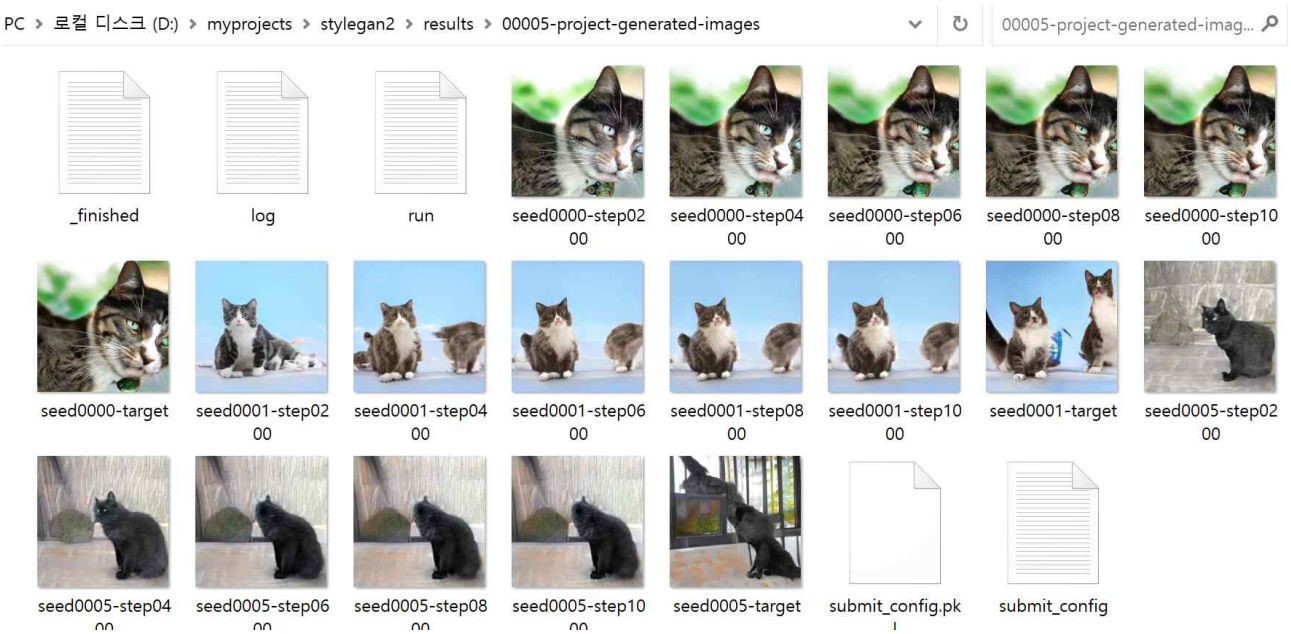
실행결과는 results\00004-project-real-images 폴더에 아래와 같이 저장됨

```
Local submit - run_dir: results₩00010-project-real-images
dnnlib: Running run_projector.project_real_images() on localhost...
Loading networks from "gdrive:networks/stylegan2-cat-config-f.pkl"...
Setting up TensorFlow plugin "fused_bias_act.cu": Preprocessing... Loading... Done.
Setting up TensorFlow plugin "upfirdn_2d.cu": Preprocessing... Loading... Done.
Loading images from "cat-256x256-datasets"...
Projecting image 0/3 ...
Projecting image 1/3 ...
Projecting image 2/3 ...
dnnlib: Finished run_projector.project_real_images() in 6m 00s.
```



Project generated images

python run_projector.py project-generated-images --network=gdrive:networks/stylegan2-cat-config-f.pkl —seeds=0,1,5



[7] Network 학습

학습환경 (Config)

We have verified that the results match the paper when training with 1, 2, 4, or 8 GPUs. Note that training FFHQ at 1024×1024 resolution requires GPU(s) with at least 16 GB of memory. The following table lists typical training times using NVIDIA DGX-1 with 8 Tesla V100 GPUs:

Configuration	Resolution	Total kimg	1 GPU	2 GPUs	4 GPUs	8 GPUs	GPU mem
`config-f`	1024×1024	25000	69d 23h	36d 4h	18d 14h	9d 18h	13.3 GB
`config-f`	1024×1024	10000	27d 23h	14d 11h	7d 10h	3d 22h	13.3 GB
`config-e`	1024×1024	25000	35d 11h	18d 15h	9d 15h	5d 6h	8.6 GB
`config-e`	1024×1024	10000	14d 4h	7d 11h	3d 20h	2d 3h	8.6 GB
`config-f`	256×256	25000	32d 13h	16d 23h	8d 21h	4d 18h	6.4 GB
`config-f`	256×256	10000	13d 0h	6d 19h	3d 13h	1d 22h	6.4 GB

 # 품질 측정 기준들

Metric	FFHQ config F	1 GPU	2 GPUs	4 GPUs	Description
`fid50k`	2.84 ± 0.03	22 min	14 min	10 min	Fréchet Inception Distance
`is50k`	5.13 ± 0.02	23 min	14 min	8 min	Inception Score
`ppl_zfull`	348.0 ± 3.8	41 min	22 min	14 min	Perceptual Path Length in Z, full paths
`ppl_wfull`	126.9 ± 0.2	42 min	22 min	13 min	Perceptual Path Length in W, full paths
`ppl_zend`	348.6 ± 3.0	41 min	22 min	14 min	Perceptual Path Length in Z, path endpoints
`ppl_wend`	129.4 ± 0.8	40 min	23 min	13 min	Perceptual Path Length in W, path endpoints
`ppl2_wend`	145.0 ± 0.5	41 min	23 min	14 min	Perceptual Path Length without center crop
`ls`	154.2 / 4.27	10 hrs	6 hrs	4 hrs	Linear Separability
`pr50k3`	0.689 / 0.492	26 min	17 min	12 min	Precision and Recall

다른 학습환경(config-?) 확인

```
python run_training.py —help
```

```
optional arguments:
  -h, --help            show this help message and exit
  --result-dir DIR      Root directory for run results (default: results)
  --data-dir DATA_DIR   Dataset root directory
  --dataset DATASET     Training dataset
  --config CONFIG       Training config (default: config-f)
  --num-gpus N          Number of GPUs (default: 1)
  --total-kimg KIMG     Training length in thousands of images (default:25000)
  --gamma GAMMA         R1 regularization weight (default is config dependent)
  --mirror-augment BOOL Mirror augment (default: False)
  --metrics METRICS     Comma-separated list of metrics or "none" (default: fid50k)
```

Custom dataset 으로 학습

```
python run_training.py --num-gpus=1 --data-dir=./datasets
        --config=config-f  --dataset=cat-256x256-datasets
```

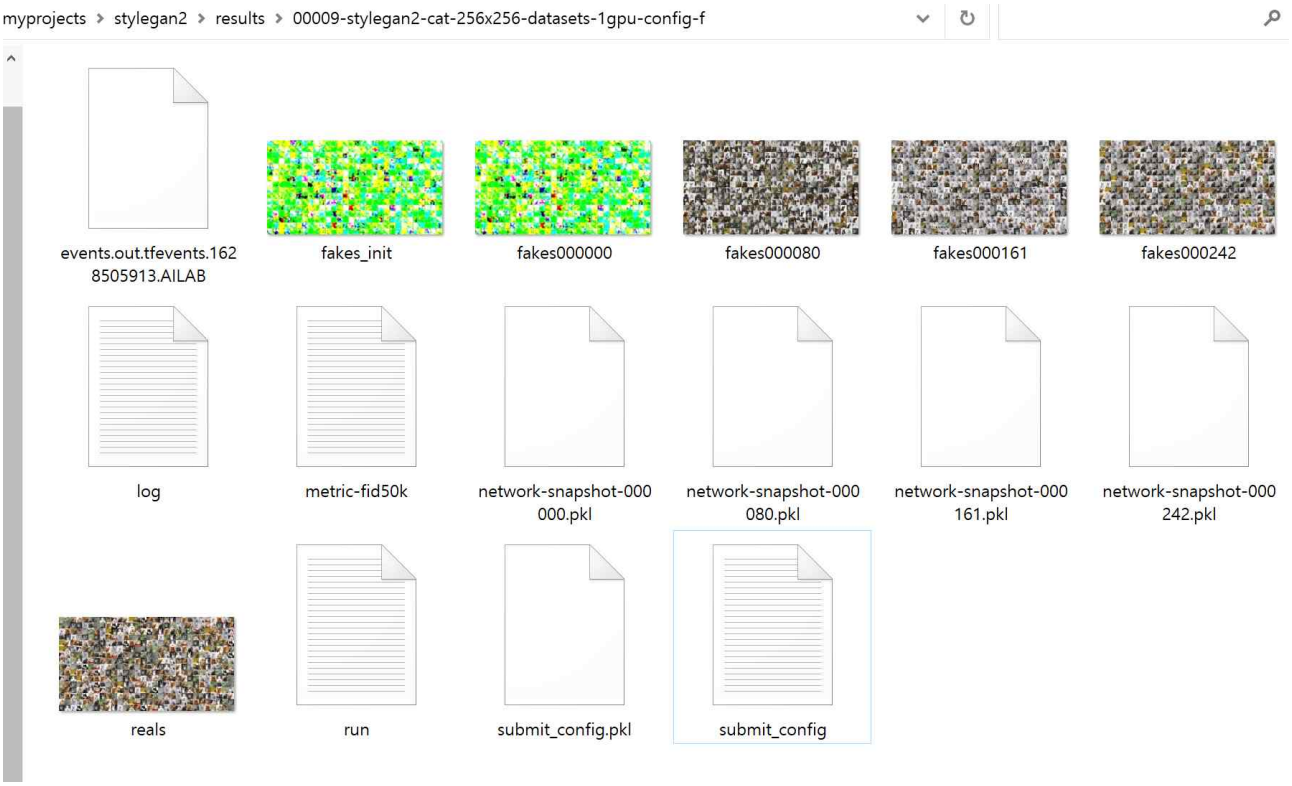
```
Streaming data using training.dataset.TFRecordDataset...
Dataset shape = [3, 256, 256]
Dynamic range = [0, 255]
Label size    = 0
Constructing networks...
Setting up TensorFlow plugin "fused_bias_act.cu": Preprocessing... Loading... Done.
Setting up TensorFlow plugin "upfirdn_2d.cu": Preprocessing... Loading... Done.

G                                  Params    OutputShape         WeightShape
---                                ---       ---                 ---
latents_in                         -         (?, 512)            -
labels_in                          -         (?, 0)              -
lod                                -         ()                  -
dlatent_avg                        -         (512,)              -
G_mapping/latents_in               -         (?, 512)            -
G_mapping/labels_in                -         (?, 0)              -
G_mapping/Normalize                -         (?, 512)            -
G_mapping/Dense0                   262656    (?, 512)            (512, 512)
G_mapping/Dense1                   262656    (?, 512)            (512, 512)
G_mapping/Dense2                   262656    (?, 512)            (512, 512)
G_mapping/Dense3                   262656    (?, 512)            (512, 512)
G_mapping/Dense4                   262656    (?, 512)            (512, 512)
G_mapping/Dense5                   262656    (?, 512)            (512, 512)
G_mapping/Dense6                   262656    (?, 512)            (512, 512)
G_mapping/Dense7                   262656    (?, 512)            (512, 512)
G_mapping/Broadcast                -         (?, 14, 512)        -
G_mapping/dlatents_out             -         (?, 14, 512)        -
Truncation/Lerp                    -         (?, 14, 512)        -
G_synthesis/dlatents_in            -         (?, 14, 512)        -
G_synthesis/4x4/Const              8192      (?, 512, 4, 4)      (1, 512, 4, 4)
G_synthesis/4x4/Conv               2622465   (?, 512, 4, 4)      (3, 3, 512, 512)
G_synthesis/4x4/ToRGB              264195    (?, 3, 4, 4)        (1, 1, 512, 3)
G_synthesis/8x8/Conv0_up           2622465   (?, 512, 8, 8)      (3, 3, 512, 512)
G_synthesis/8x8/Conv1              2622465   (?, 512, 8, 8)      (3, 3, 512, 512)
G_synthesis/8x8/Upsample           -         (?, 3, 8, 8)        -
G_synthesis/8x8/ToRGB              264195    (?, 3, 8, 8)        (1, 1, 512, 3)
G_synthesis/16x16/Conv0_up         2622465   (?, 512, 16, 16)    (3, 3, 512, 512)
G_synthesis/16x16/Conv1            2622465   (?, 512, 16, 16)    (3, 3, 512, 512)
G_synthesis/16x16/Upsample         -         (?, 3, 16, 16)      -
G_synthesis/16x16/ToRGB            264195    (?, 3, 16, 16)      (1, 1, 512, 3)
G_synthesis/32x32/Conv0_up         2622465   (?, 512, 32, 32)    (3, 3, 512, 512)
G_synthesis/32x32/Conv1            2622465   (?, 512, 32, 32)    (3, 3, 512, 512)
G_synthesis/32x32/Upsample         -         (?, 3, 32, 32)      -
G_synthesis/32x32/ToRGB            264195    (?, 3, 32, 32)      (1, 1, 512, 3)
G_synthesis/64x64/Conv0_up         2622465   (?, 512, 64, 64)    (3, 3, 512, 512)
G_synthesis/64x64/Conv1            2622465   (?, 512, 64, 64)    (3, 3, 512, 512)
G_synthesis/64x64/Upsample         -         (?, 3, 64, 64)      -
G_synthesis/64x64/ToRGB            264195    (?, 3, 64, 64)      (1, 1, 512, 3)
G_synthesis/128x128/Conv0_up       1442561   (?, 256, 128, 128)  (3, 3, 512, 256)
G_synthesis/128x128/Conv1          721409    (?, 256, 128, 128)  (3, 3, 256, 256)
G_synthesis/128x128/Upsample       -         (?, 3, 128, 128)    -
G_synthesis/128x128/ToRGB          132099    (?, 3, 128, 128)    (1, 1, 256, 3)
G_synthesis/256x256/Conv0_up       426369    (?, 128, 256, 256)  (3, 3, 256, 128)
G_synthesis/256x256/Conv1          213249    (?, 128, 256, 256)  (3, 3, 128, 128)
G_synthesis/256x256/Upsample       -         (?, 3, 256, 256)    -
G_synthesis/256x256/ToRGB          66051     (?, 3, 256, 256)    (1, 1, 128, 3)
G_synthesis/images_out             -         (?, 3, 256, 256)    -
G_synthesis/noise0                 -         (1, 1, 4, 4)        -
G_synthesis/noise1                 -         (1, 1, 8, 8)        -
G_synthesis/noise2                 -         (1, 1, 8, 8)        -
G_synthesis/noise3                 -         (1, 1, 16, 16)      -
G_synthesis/noise4                 -         (1, 1, 16, 16)      -
G_synthesis/noise5                 -         (1, 1, 32, 32)      -
G_synthesis/noise6                 -         (1, 1, 32, 32)      -
G_synthesis/noise7                 -         (1, 1, 64, 64)      -
G_synthesis/noise8                 -         (1, 1, 64, 64)      -
G_synthesis/noise9                 -         (1, 1, 128, 128)    -
G_synthesis/noise10                -         (1, 1, 128, 128)    -
G_synthesis/noise11                -         (1, 1, 256, 256)    -
G_synthesis/noise12                -         (1, 1, 256, 256)    -
images_out                         -         (?, 3, 256, 256)    -
---                                ---       ---                 ---
Total                              30034338
```

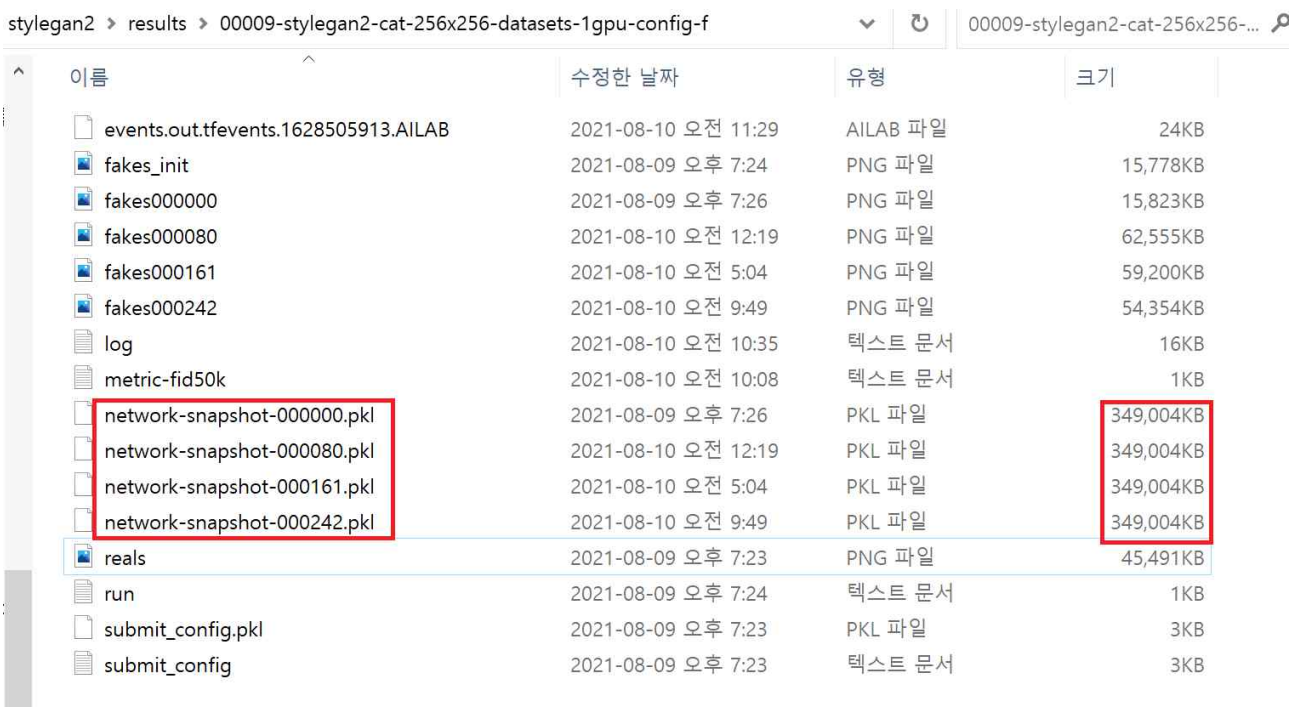
D	Params	OutputShape	WeightShape
images_in	-	(?, 3, 256, 256)	-
labels_in	-	(?, 0)	-
256x256/FromRGB	512	(?, 128, 256, 256)	(1, 1, 3, 128)
256x256/Conv0	147584	(?, 128, 256, 256)	(3, 3, 128, 128)
256x256/Conv1_down	295168	(?, 256, 128, 128)	(3, 3, 128, 256)
256x256/Skip	32768	(?, 256, 128, 128)	(1, 1, 128, 256)
128x128/Conv0	590080	(?, 256, 128, 128)	(3, 3, 256, 256)
128x128/Conv1_down	1180160	(?, 512, 64, 64)	(3, 3, 256, 512)
128x128/Skip	131072	(?, 512, 64, 64)	(1, 1, 256, 512)
64x64/Conv0	2359808	(?, 512, 64, 64)	(3, 3, 512, 512)
64x64/Conv1_down	2359808	(?, 512, 32, 32)	(3, 3, 512, 512)
64x64/Skip	262144	(?, 512, 32, 32)	(1, 1, 512, 512)
32x32/Conv0	2359808	(?, 512, 32, 32)	(3, 3, 512, 512)
32x32/Conv1_down	2359808	(?, 512, 16, 16)	(3, 3, 512, 512)
32x32/Skip	262144	(?, 512, 16, 16)	(1, 1, 512, 512)
16x16/Conv0	2359808	(?, 512, 16, 16)	(3, 3, 512, 512)
16x16/Conv1_down	2359808	(?, 512, 8, 8)	(3, 3, 512, 512)
16x16/Skip	262144	(?, 512, 8, 8)	(1, 1, 512, 512)
8x8/Conv0	2359808	(?, 512, 8, 8)	(3, 3, 512, 512)
8x8/Conv1_down	2359808	(?, 512, 4, 4)	(3, 3, 512, 512)
8x8/Skip	262144	(?, 512, 4, 4)	(1, 1, 512, 512)
4x4/MinibatchStddev	-	(?, 513, 4, 4)	-
4x4/Conv	2364416	(?, 512, 4, 4)	(3, 3, 513, 512)
4x4/Dense0	4194816	(?, 512)	(8192, 512)
Output	513	(?, 1)	(512, 1)
scores_out	-	(?, 1)	-
---	---	---	---
Total	28864129		

학습과정 예

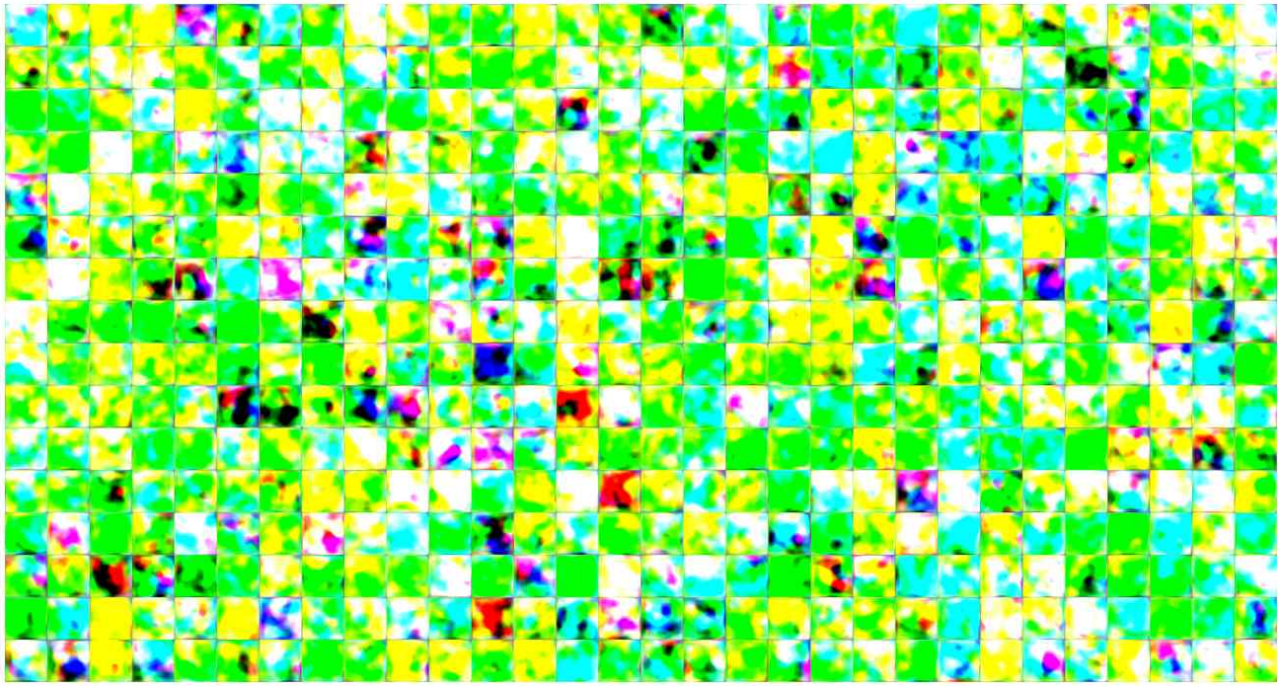
```
Building TensorFlow graph...
Initializing logs...
Training for 25000 kimg...

tick 0     kimg 0.1      lod 0.00  minibatch 32  time 38s          sec/tick 38.4    sec/kimg 300.13  maintenance 0.0    gpumem 6.1
network-snapshot-000000        time 19m 04s      fid50k 336.6543
tick 1     kimg 8.2      lod 0.00  minibatch 32  time 51m 11s      sec/tick 1855.9  sec/kimg 230.14  maintenance 1176.7 gpumem 6.1
tick 2     kimg 16.3     lod 0.00  minibatch 32  time 1h 21m 22s   sec/tick 1810.8  sec/kimg 224.55  maintenance 0.0    gpumem 6.1
tick 3     kimg 24.3     lod 0.00  minibatch 32  time 1h 47m 57s   sec/tick 1595.1  sec/kimg 197.81  maintenance 0.0    gpumem 6.1
tick 4     kimg 32.4     lod 0.00  minibatch 32  time 2h 14m 33s   sec/tick 1595.9  sec/kimg 197.90  maintenance 0.0    gpumem 6.1
tick 5     kimg 40.4     lod 0.00  minibatch 32  time 2h 41m 07s   sec/tick 1594.4  sec/kimg 197.72  maintenance 0.0    gpumem 6.1
tick 6     kimg 48.5     lod 0.00  minibatch 32  time 3h 07m 44s   sec/tick 1596.7  sec/kimg 198.00  maintenance 0.0    gpumem 6.1
tick 7     kimg 56.6     lod 0.00  minibatch 32  time 3h 34m 21s   sec/tick 1597.2  sec/kimg 198.07  maintenance 0.0    gpumem 6.1
tick 8     kimg 64.6     lod 0.00  minibatch 32  time 4h 00m 58s   sec/tick 1597.2  sec/kimg 198.07  maintenance 0.0    gpumem 6.1
tick 9     kimg 72.7     lod 0.00  minibatch 32  time 4h 27m 33s   sec/tick 1594.8  sec/kimg 197.77  maintenance 0.0    gpumem 6.1
tick 10    kimg 80.8     lod 0.00  minibatch 32  time 4h 54m 11s   sec/tick 1597.5  sec/kimg 198.10  maintenance 0.0    gpumem 6.1
network-snapshot-000080        time 18m 47s      fid50k 416.4822
tick 11    kimg 88.8     lod 0.00  minibatch 32  time 5h 39m 57s   sec/tick 1596.4  sec/kimg 197.97  maintenance 1149.4 gpumem 6.1
tick 12    kimg 96.9     lod 0.00  minibatch 32  time 6h 06m 34s   sec/tick 1597.8  sec/kimg 198.14  maintenance 0.0    gpumem 6.1
tick 13    kimg 105.0    lod 0.00  minibatch 32  time 6h 33m 09s   sec/tick 1594.9  sec/kimg 197.78  maintenance 0.0    gpumem 6.1
tick 14    kimg 113.0    lod 0.00  minibatch 32  time 6h 59m 46s   sec/tick 1596.7  sec/kimg 198.01  maintenance 0.0    gpumem 6.1
tick 15    kimg 121.1    lod 0.00  minibatch 32  time 7h 26m 23s   sec/tick 1597.1  sec/kimg 198.06  maintenance 0.0    gpumem 6.1
tick 16    kimg 129.2    lod 0.00  minibatch 32  time 7h 53m 00s   sec/tick 1596.7  sec/kimg 198.01  maintenance 0.0    gpumem 6.1
tick 17    kimg 137.2    lod 0.00  minibatch 32  time 8h 19m 33s   sec/tick 1592.8  sec/kimg 197.52  maintenance 0.0    gpumem 6.1
tick 18    kimg 145.3    lod 0.00  minibatch 32  time 8h 46m 09s   sec/tick 1595.8  sec/kimg 197.89  maintenance 0.0    gpumem 6.1
tick 19    kimg 153.3    lod 0.00  minibatch 32  time 9h 12m 45s   sec/tick 1596.2  sec/kimg 197.94  maintenance 0.0    gpumem 6.1
tick 20    kimg 161.4    lod 0.00  minibatch 32  time 9h 39m 21s   sec/tick 1596.2  sec/kimg 197.94  maintenance 0.0    gpumem 6.1
network-snapshot-000161        time 18m 47s      fid50k 373.4701
tick 21    kimg 169.5    lod 0.00  minibatch 32  time 10h 25m 06s  sec/tick 1594.1  sec/kimg 197.68  maintenance 1150.7 gpumem 6.1
tick 22    kimg 177.5    lod 0.00  minibatch 32  time 10h 51m 42s  sec/tick 1595.9  sec/kimg 197.90  maintenance 0.0    gpumem 6.1
tick 23    kimg 185.6    lod 0.00  minibatch 32  time 11h 18m 17s  sec/tick 1595.6  sec/kimg 197.86  maintenance 0.0    gpumem 6.1
tick 24    kimg 193.7    lod 0.00  minibatch 32  time 11h 44m 54s  sec/tick 1596.5  sec/kimg 197.97  maintenance 0.0    gpumem 6.1
tick 25    kimg 201.7    lod 0.00  minibatch 32  time 12h 11m 34s  sec/tick 1600.1  sec/kimg 198.43  maintenance 0.0    gpumem 6.1
tick 26    kimg 209.8    lod 0.00  minibatch 32  time 12h 38m 10s  sec/tick 1596.5  sec/kimg 197.97  maintenance 0.0    gpumem 6.1
tick 27    kimg 217.9    lod 0.00  minibatch 32  time 13h 04m 47s  sec/tick 1597.1  sec/kimg 198.05  maintenance 0.0    gpumem 6.1
tick 28    kimg 225.9    lod 0.00  minibatch 32  time 13h 31m 23s  sec/tick 1595.6  sec/kimg 197.87  maintenance 0.0    gpumem 6.1
tick 29    kimg 234.0    lod 0.00  minibatch 32  time 13h 57m 58s  sec/tick 1594.5  sec/kimg 197.73  maintenance 0.0    gpumem 6.1
tick 30    kimg 242.0    lod 0.00  minibatch 32  time 14h 24m 35s  sec/tick 1597.7  sec/kimg 198.12  maintenance 0.0    gpumem 6.1
network-snapshot-000242        time 18m 50s      fid50k 191.5806
tick 31    kimg 250.1    lod 0.00  minibatch 32  time 15h 10m 29s  sec/tick 1599.4  sec/kimg 198.34  maintenance 1154.1 gpumem 6.1
tick 32    kimg 258.2    lod 0.00  minibatch 32  time 15h 37m 23s  sec/tick 1614.1  sec/kimg 200.17  maintenance 0.0    gpumem 6.1
tick 33    kimg 266.2    lod 0.00  minibatch 32  time 16h 04m 19s  sec/tick 1616.1  sec/kimg 200.41  maintenance 0.0    gpumem 6.1
```

results 폴더에 반영되는 학습과정의 예




stylegan2 › results › 00009-stylegan2-cat-256x256-datasets-1gpu-config-f
00009-stylegan2-cat-256x256-...
이름
수정한 날짜
유형
크기
events.out.tfevents.1628505913.AILAB 2021-08-10 오전 11:29 AILAB 파일 24KB
fakes_init 2021-08-09 오후 7:24 PNG 파일 15,778KB
fakes000000 2021-08-09 오후 7:26 PNG 파일 15,823KB
fakes000080 2021-08-10 오전 12:19 PNG 파일 62,555KB
fakes000161 2021-08-10 오전 5:04 PNG 파일 59,200KB
fakes000242 2021-08-10 오전 9:49 PNG 파일 54,354KB
log 2021-08-10 오전 10:35 텍스트 문서 16KB
metric-fid50k 2021-08-10 오전 10:08 텍스트 문서 1KB
network-snapshot-000000.pkl 2021-08-09 오후 7:26 PKL 파일 349,004KB
network-snapshot-000080.pkl 2021-08-10 오전 12:19 PKL 파일 349,004KB
network-snapshot-000161.pkl 2021-08-10 오전 5:04 PKL 파일 349,004KB
network-snapshot-000242.pkl 2021-08-10 오전 9:49 PKL 파일 349,004KB
reals 2021-08-09 오후 7:23 PNG 파일 45,491KB
run 2021-08-09 오후 7:24 텍스트 문서 1KB
submit_config.pkl 2021-08-09 오후 7:23 PKL 파일 3KB
submit_config 2021-08-09 오후 7:23 텍스트 문서 3KB

학습중 fake image 확인 파일: (http://ailab.silla.ac.kr/lec/pipaai/gan/stylegan2/fakes000000.png)



학습중 fake image 확인 파일: (http://ailab.silla.ac.kr/lec/pipaai/gan/stylegan2/fakes000080.png)

학습중 fake image 확인 파일: (http://ailab.silla.ac.kr/lec/pipaai/gan/stylegan2/fakes000242.png)

real image 확인 파일: (http://ailab.silla.ac.kr/lec/pipaai/gan/stylegan2/reals.png)

[8] 이미지 생성 (1000개)

python run_generator.py generate-images --seeds=0-999 --truncation-psi=1.0 \
  --network=results/0000??-stylegan2-ffhq-1gpu-config-f/networks-final.pkl

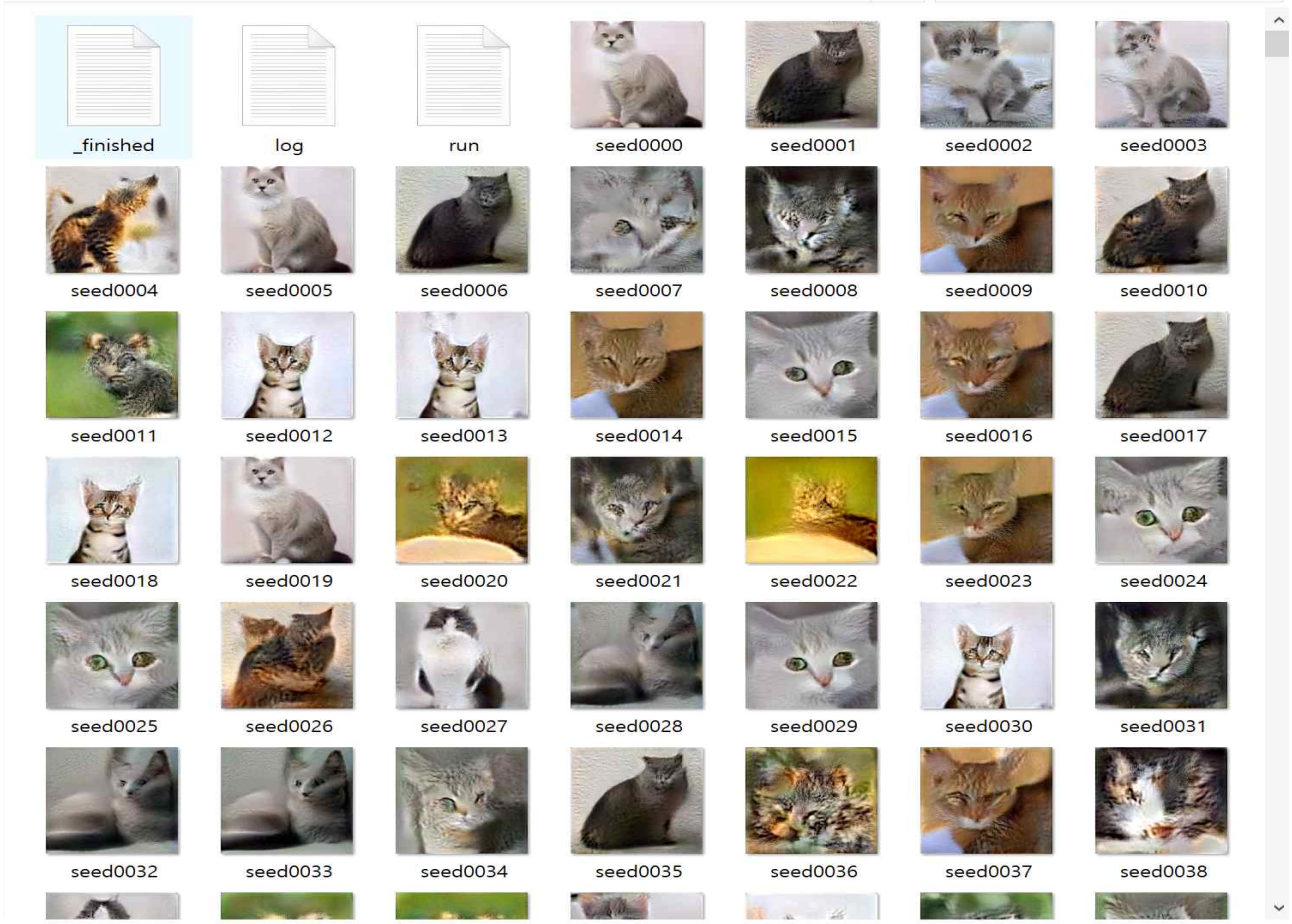
python run_generator.py generate-images --seeds=0-999 —truncation-psi=1.0 --network=results/00009-stylegan2-cat-256x256-datasets-1gpu-config-f/network-snapshot-000242.pkl

```
Local submit - run_dir: results#00012-generate-images
dnnlib: Running run_generator.generate_images() on localhost...
Loading networks from "results/00009-stylegan2-cat-256x256-datasets-1gpu-config-f/network-snapshot-000242.pkl"...
Setting up TensorFlow plugin "fused_bias_act.cu": Preprocessing... Loading... Done.
Setting up TensorFlow plugin "upfirdn_2d.cu": Preprocessing... Loading... Done.
Generating image for seed 0 (0/1000) ...
Generating image for seed 1 (1/1000) ...
Generating image for seed 2 (2/1000) ...
Generating image for seed 3 (3/1000) ...
Generating image for seed 4 (4/1000) ...
Generating image for seed 5 (5/1000) ...
Generating image for seed 6 (6/1000) ...
Generating image for seed 7 (7/1000) ...
Generating image for seed 8 (8/1000) ...
Generating image for seed 9 (9/1000) ...
Generating image for seed 10 (10/1000) ...
Generating image for seed 11 (11/1000) ...
```

```
Generating image for seed 991 (991/1000) ...
Generating image for seed 992 (992/1000) ...
Generating image for seed 993 (993/1000) ...
Generating image for seed 994 (994/1000) ...
Generating image for seed 995 (995/1000) ...
Generating image for seed 996 (996/1000) ...
Generating image for seed 997 (997/1000) ...
Generating image for seed 998 (998/1000) ...
Generating image for seed 999 (999/1000) ...
dnnlib: Finished run_generator.generate_images() in 1m 10s.
```

컴 디스크 (D:) > myprojects > stylegan2 > results > 00012-generate-images

00012-generate-images 검색



고려할 사항

데이터 준비

- 보통 몇만~몇십만개의 데이터 필요
- 정방형 데이터 크기 : 256x256, 512x512, 1024x1024, 하드웨어 환경 고려
- Data Augmentation : Mirroring 여부

I experimented mainly with two “fashion” datasets

- dresses (~30k packshot images)
- footwear (~140k packshot images)

Real images samples from the footwear and dresses datasets

(https://towardsdatascience.com/stylegan-v2-notes-on-training-and-latent-space-exploration-e51cf96584b3?gi=6980a11c5852)

Configuration : 신경망 모델의 구조와 특성을 결정

: config-a 부터 weight demodulation, lazy regularization, path length regularization 등을 추가 하거나 조정하여 config-f 까지 있음

config-a - the original SyleGAN v1

~

~

config-e - the full updated setup for v2

config-f - a larger network version of config-e

생성된 영상의 품질 평가 지표 (FID : Frechet Inception Distance)

: 생성된 영상의 품질을 평가, 영상 집합 사이의 거리를 측정 (낮은 숫자가 좋음)

: 방법 - Inception Network의 중간 layer에서의 feature 값의 평균과 공분산을 이용하여 계산

$$FID(x,g) = \left|\left|\mu_x - \mu_g\right|\right|_2^2 + Tr\left(\Sigma_x + \Sigma_g - 2\left(\Sigma_x\Sigma_g\right)^{\frac{1}{2}}\right)$$

※Tr: 대각 성분의 합

학습률 lr(Learning rate)

: lr 값은 크지 않게, G 와 D 각각 별도의 lr값 사용

: 큰 lr 값은 학습 시 발산하기 쉬움

StyleGan2-Ada (adaptive discriminator augmentation)

: Training Generative Adversarial Networks with Limited Data (June 11, 2020)
(https://github.com/NVlabs/stylegan2-ada)

- 적은 수(3만개 미만) 학습 데이터의 경우 Discriminator가 데이터에 과적합 되는 학습 발산 문제 개선
- ADA 이용: 3만개 미만의 CIFAR-10 이미지 학습해서 State-of-the-art 성능 보임
- Mixed-precision support: 1.6배 까지 학습 가속, 1.3배까지 추론 가속, 1.5배까지 GPU 메모리 소비 줄임
- Better hyperparameter defaults: 하이퍼파라미터 설정이 더 간편해짐
- Clean codebase: 코드 리펙토링, 단순화, 사용하기 더 쉬움
- Command line tools: 논문대로 학습 가능, 임의의 이미지에 대해서도 비디오로 projection 가능
- Network import: 학습된 stylegan2용 pkl파일을 네트워크상에서 다운로드 가능
- Augmentation pipeline: 초고화질 이미지 증강 기법을 GPU를 이용해서 구현

생성된 이미지 예

기 학습된 모델 이용 비디오 프로젝션 예 (https://tv.kakao.com/v/413242134)