Cycle GAN - 6조

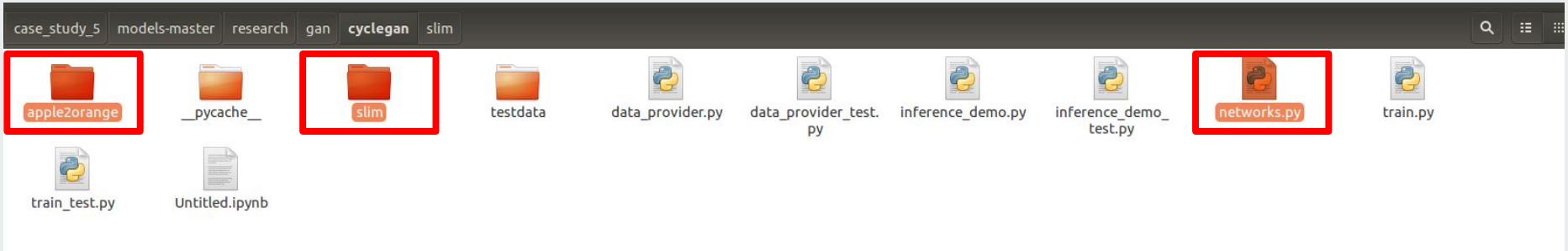
강재훈, 김민선, 송영훈

train.py를 실행했을 때 오류

```
root@681aead386be:/workspace/week5/week5/models-master/research/gan/cyclegan# python train.py
Traceback (most recent call last):
  File "train.py", line 26, in <module>
    import networks
ModuleNotFoundError: No module named 'networks'
```

```
root@681aead386be:/workspace/week5/week5/models-master/research/gan/cyclegan# python train.py
Traceback (most recent call last):
  File "train.py", line 26, in <module>
    import networks
  File "/workspace/week5/week5/models-master/research/gan/cyclegan/networks.py", line 23, in <module>
    from slim.nets import cyclegan
ModuleNotFoundError: No module named 'slim'
```

research로부터 networks.py, slim파일 복사



python - from XXX import ooo

__future__

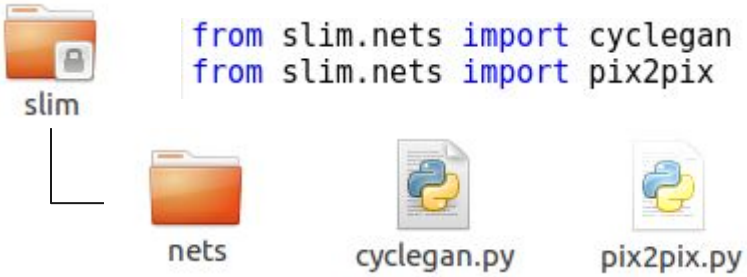
: 파이썬 2에서 파이썬3 기능을 일부 사용할 수 있게 해준다.

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
```

python - from XXX import ooo

디렉토리로부터 import

: 현재 디렉토리에서 slim/nets폴더에 있는 cyclegan, pix2pix를 import한다.




```
from slim.nets import cyclegan
from slim.nets import pix2pix
```

tensorflow flags

Tensorflow에서 제공하는 flags

객체를 사용하면, 고정값으로 되어있는 기본적인 데이터를 편리하게 사용할 수 있다.

```
# x file director 'None' -->
flags.DEFINE_string('image_set_x_file_pattern','apple2orange/trainA/*.jpg' ,'File pattern of images in image set X')
flags.DEFINE_string('image_set_y_file_pattern','apple2orange/trainB/*.jpg' ,'File pattern of images in image set Y')
```

```
In [10]: FLAGS.image_set_x_file_pattern
Out[10]: 'apple2orange/trainA/*.jpg'
```

define_model함수에서 에러

```
root@681aead386be:/workspace/week5/week5/models-master/research/gan/cyclegan# python train.py
/root/anaconda3/lib/python3.6/site-packages/absl/flags/_validators.py:359: UserWarning: Flag --image_
set_x_file_pattern has a non-None default value; therefore, mark_flag_as_required will pass even if f
lag is not specified in the command line!
  'command line!' % flag_name)
/root/anaconda3/lib/python3.6/site-packages/absl/flags/_validators.py:359: UserWarning: Flag --image_
set_y_file_pattern has a non-None default value; therefore, mark_flag_as_required will pass even if f
lag is not specified in the command line!
  'command line!' % flag_name)
Traceback (most recent call last):
  File "train.py", line 218, in <module>
    tf.app.run()
  File "/root/anaconda3/lib/python3.6/site-packages/tensorflow/python/platform/app.py", line 126, in
run
    _sys.exit(main(argv))
  File "train.py", line 182, in main
    cyclegan_model = _define_model(images_x, images_y)
  File "train.py", line 91, in _define_model
    cyclegan_model, num_comparisons=3, display_diffs=False)
  File "/root/anaconda3/lib/python3.6/site-packages/tensorflow/contrib/gan/python/eval/python/summari
es_impl.py", line 151, in add_image_comparison_summaries
    _assert_is_image(gan_model.generator_inputs)
AttributeError: 'CycleGANModel' object has no attribute 'generator_inputs'
```

train.py

```
#flags.DEFINE_string('image_set_x_file_pattern', None,
flags.DEFINE_string('image_set_x_file_pattern', '/workspace/case_study_5/models-master/research/gan/cyclegan/apple2orange/trainA/*.jpg',
                    'File pattern of images in image set X')

#flags.DEFINE_string('image_set_y_file_pattern', None,
flags.DEFINE_string('image_set_y_file_pattern', '/workspace/case_study_5/models-master/research/gan/cyclegan/apple2orange/trainB/*.jpg',
                    'File pattern of images in image set Y')
```

```python
def _define_model(images_x, images_y):
  """Defines a CycleGAN model that maps between images_x and images_y.

  Args:
    images_x: A 4D float `Tensor` of NHWC format.  Images in set X.
    images_y: A 4D float `Tensor` of NHWC format.  Images in set Y.

  Returns:
    A `CycleGANModel` namedtuple.
  """
  cyclegan_model = tfgan.cyclegan_model(
      generator_fn=networks.generator,
      discriminator_fn=networks.discriminator,
      data_x=images_x,
      data_y=images_y)

  # Add summaries for generated images.
  tfgan.eval.add_image_comparison_summaries(
      cyclegan_model, num_comparisons=3, display_diffs=False)
  tfgan.eval.add_gan_model_image_summaries(
      cyclegan_model, grid_size=int(np.sqrt(FLAGS.batch_size)))

  return cyclegan_model
```

```python
def _define_model(images_x, images_y):
  """Defines a CycleGAN model that maps between images_x and images_y.

  Args:
    images_x: A 4D float `Tensor` of NHWC format.  Images in set X.
    images_y: A 4D float `Tensor` of NHWC format.  Images in set Y.

  Returns:
    A `CycleGANModel` namedtuple.
  """
  cyclegan_model = tfgan.cyclegan_model(
      generator_fn=networks.generator,
      discriminator_fn=networks.discriminator,
      data_x=images_x,
      data_y=images_y)

  # Add summaries for generated images.
#  tfgan.eval.add_image_comparison_summaries(
#      cyclegan_model, num_comparisons=3, display_diffs=False)
#  tfgan.eval.add_gan_model_image_summaries(
# tfgan.eval.add_cyclegan_image_summaries(
#      cyclegan_model, grid_size=int(np.sqrt(FLAGS.batch_size)))
  tfgan.eval.add_cyclegan_image_summaries(cyclegan_model)
  return cyclegan_model
```

```
ValueError: `add_gan_model_image_summaries` does not take CycleGANModels. Please use `add_cyclegan_image_summaries`
 instead.
```

add_cyclegan_image_summaries

Adds image summaries for CycleGAN.

There are two summaries, one for each generator. The first image is the generator input, the second is the generator output, and the third is G(F(x)).

코드가 실행되는지는 확인(CPU버전)

```
ktai19@ktai19-Alienware-Aurora-R7: ~
Every 2.0s: nvidia-smi                                  Thu May  3 13:45:37 2018

Thu May  3 13:45:37 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.48                 Driver Version: 390.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1080    Off  | 00000000:01:00.0  On |                  N/A |
| 27%   34C    P8     9W / 180W |    851MiB /  8117MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1080    Off  | 00000000:02:00.0 Off |                  N/A |
| 27%   30C    P8     6W / 180W |      2MiB /  8119MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1176      G   /usr/lib/xorg/Xorg                           470MiB |
|    0      3329      G   compiz                                       178MiB |
|    0      4784      G   ...-token=189419AEC69931CF349E6BB75CFA4176   200MiB |
+-----------------------------------------------------------------------------+
```

```
  'command line!' % flag_name)
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/cyclegan/model.ckpt-1200
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 1201 into /tmp/cyclegan/model.ckpt.
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1200'
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1210' (5.089 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1221' (4.920 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1231' (4.915 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1241' (4.916 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1251' (4.930 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1261' (4.886 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
1271' (4.896 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step:
```

cuda 8.0 → cuda 9.0

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "train.py", line 23, in <module>
    import tensorflow as tf
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/__init__.py", line 24, in <module>
    from tensorflow.python import pywrap_tensorflow  # pylint: disable=unused-import
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/__init__.py", line 49, in <module>
    from tensorflow.python import pywrap_tensorflow
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow.py", line 74, in <module>
    raise ImportError(msg)
ImportError: Traceback (most recent call last):
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow.py", line 58, in <module>
    from tensorflow.python.pywrap_tensorflow_internal import *
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow_internal.py", line 28, in <module>
    _pywrap_tensorflow_internal = swig_import_helper()
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow_internal.py", line 24, in swig_import_helper
    _mod = imp.load_module('_pywrap_tensorflow_internal', fp, pathname, description)
  File "/opt/conda/lib/python3.6/imp.py", line 243, in load_module
    return load_dynamic(name, filename, file)
  File "/opt/conda/lib/python3.6/imp.py", line 343, in load_dynamic
    return _load(spec)
ImportError: libcublas.so.9.0: cannot open shared object file: No such file or directory
```

error

cuda 8.0 → cuda 9.0

```
#include "driver_types.h"
root@ktai17:/workspace/case_study_5/models-master/research/gan/cyclegan# nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri Sep  1 21:08:03 CDT 2017
Cuda compilation tools, release 9.0, V9.0.176
```

apt-get install cuda-9-0 : update 후,

nvcc --version : cuda version 확인

cudnn 6.0 → cudnn 7.0

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "train.py", line 23, in <module>
    import tensorflow as tf
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/__init__.py", line 24, in <module>
    from tensorflow.python import pywrap_tensorflow  # pylint: disable=unused-import
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/__init__.py", line 49, in <module>
    from tensorflow.python import pywrap_tensorflow
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow.py", line 74, in <module>
    raise ImportError(msg)
ImportError: Traceback (most recent call last):
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow.py", line 58, in <module>
    from tensorflow.python.pywrap_tensorflow_internal import *
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow_internal.py", line 28, in <module>
    _pywrap_tensorflow_internal = swig_import_helper()
  File "/opt/conda/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow_internal.py", line 24, in swig_import_helper
    _mod = imp.load_module('_pywrap_tensorflow_internal', fp, pathname, description)
  File "/opt/conda/lib/python3.6/imp.py", line 243, in load_module
    return load_dynamic(name, filename, file)
  File "/opt/conda/lib/python3.6/imp.py", line 343, in load_dynamic
    return _load(spec)
ImportError: libcudnn.so.7: cannot open shared object file: No such file or directory
```

error

cudnn 6.0 → cudnn 7.0

```
root@ktai17:/workspace/case_study_5/models-master/research/gan/cyclegan# cat /usr/include/cudnn.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR      6
#define CUDNN_MINOR      0
#define CUDNN_PATCHLEVEL 21
--
#define CUDNN_VERSION    (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)

#include "driver_types.h"
root@ktai17:/workspace/case_study_5/models-master/research/gan/cyclegan#
```

cat /usr/include/cudnn.h | grep CUDNN_MAJOR -A 2 : update 전, cudnn version 확인

cudnn 6.0 → cudnn 7.0

cudnn 7.0 install

cuDNN v7.1.3 Runtime Library for Ubuntu14.04 (Deb)

cuDNN v7.1.3 Developer Library for Ubuntu14.04 (Deb)

두 deb 파일 다운로드 후,

```
dpkg -i libcudnn7_7.1.3.16-1+cuda9.0_amd64.deb libcudnn7-dev_7.1.3.16-1+cuda9.0_amd64.deb
```

cudnn 6.0 → cudnn 7.0

```
#define CUDNN_MAJOR 7
#define CUDNN_MINOR 1
#define CUDNN_PATCHLEVEL 3
--
#define CUDNN_VERSION    (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)
```

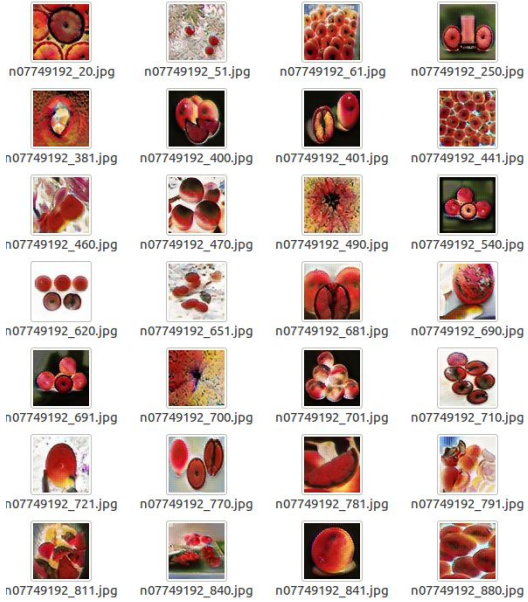
update 후, cudnn version 확인

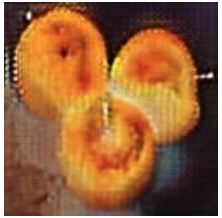
코드가 실행되는지는 확인(GPU버전)

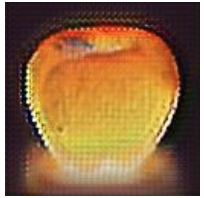
```
ktai17@ktai17: ~
Every 2.0s: nvidia-smi                                    Thu May  3 14:43:49 2018

Thu May  3 14:43:49 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.111                Driver Version: 384.111                   |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1080    Off  | 00000000:01:00.0  On |                  N/A |
| 54%   80C    P2   120W / 180W |   7800MiB /  8112MiB |     83%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1080    Off  | 00000000:02:00.0 Off |                  N/A |
| 27%   34C    P8     6W / 180W |   7718MiB /  8114MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1220      G   /usr/lib/xorg/Xorg                           261MiB |
|    0      3221      G   compiz                                       182MiB |
|    0      4673      G   ...-token=3471ADBDE097043FA9B652600213286D    52MiB |
|    0     16622      C   /opt/conda/bin/python                       7299MiB |
```

```
Console 1/A
INFO:tensorflow:global_step/sec: 9.85783
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38375' (0.983 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38385' (0.996 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38395' (1.014 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38405' (0.980 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38415' (1.004 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38425' (1.185 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38435' (0.985 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38445' (0.983 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38455' (0.971 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38465' (0.969 sec)
INFO:tensorflow:global_step/sec: 9.93491
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38475' (0.978 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38485' (1.004 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38495' (1.002 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38505' (0.982 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38515' (0.996 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38525' (1.166 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38535' (0.983 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38545' (0.998 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38555' (1.002 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38565' (1.000 sec)
INFO:tensorflow:global_step/sec: 9.87968
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38575' (0.990 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38585' (1.008 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38595' (1.004 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38605' (0.988 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38615' (0.984 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38625' (1.159 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38635' (0.973 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38645' (0.994 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38655' (1.002 sec)
INFO:tensorflow:Tensor("status_message:0", shape=(), dtype=string) = b'Starting train step: 38665' (0.985 sec)
INFO:tensorflow:global_step/sec: 9.91728
```

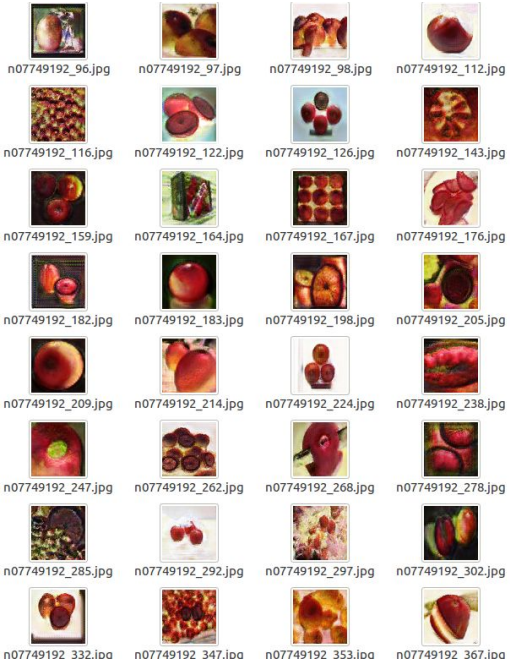
ckpt 11892로 사용한 결과

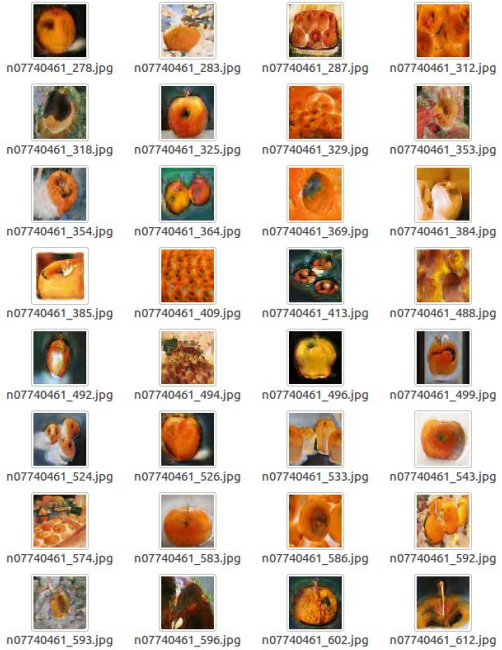






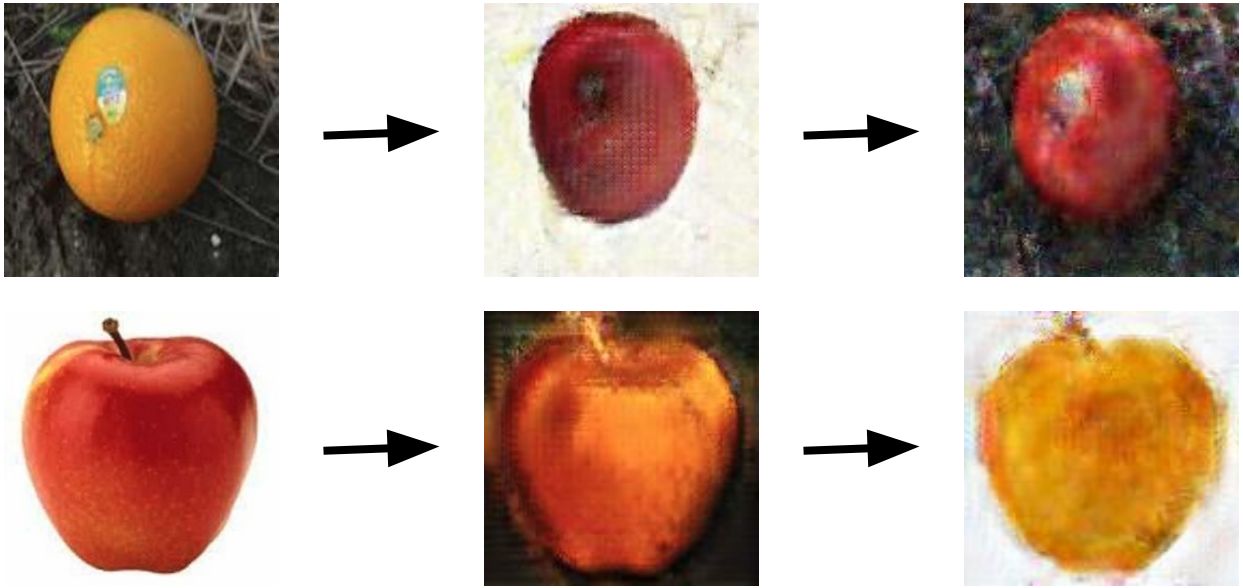
ckpt 75150으로 사용한 결과





ckpt 298970으로 사용한 결과

original → ckpt 75150 → ckpt 298970



Generator architectures We adapt our architectures from Johnson et al. [22]. We use 6 blocks for 128×128 training images, and 9 blocks for 256×256 or higher-resolution training images. Below, we follow the naming convention used in the Johnson el al.'s Github repository[5]

Let `c7s1-k` denote a 7×7 Convolution-InstanceNorm-ReLU layer with k filters and stride 1. `dk` denotes a 3×3 Convolution-InstanceNorm-ReLU layer with k filters, and stride 2. Reflection padding was used to reduce artifacts. `Rk` denotes a residual block that contains two 3×3 convolutional layers with the same number of filters on both layer. `uk` denotes a 3×3 fractional-strided-Convolution-InstanceNorm-ReLU layer with k filters, and stride $\frac{1}{2}$.

The network with 6 blocks consists of:

`c7s1-32,d64,d128,R128,R128,R128,`
`R128,R128,R128,u64,u32,c7s1-3`

The network with 9 blocks consists of:

`c7s1-32,d64,d128,R128,R128,R128,`
`R128,R128,R128,R128,R128,R128,u64,u32,c7s1-3`

```python
def generator_resnet(image, options, reuse=False, name="generator"):

    with tf.variable_scope(name):
        # image is 256 x 256 x input_c_dim
        if reuse:
            tf.get_variable_scope().reuse_variables()
        else:
            assert tf.get_variable_scope().reuse is False

        def residule_block(x, dim, ks=3, s=1, name='res'):
            p = int((ks - 1) / 2)
            y = tf.pad(x, [[0, 0], [p, p], [p, p], [0, 0]], "REFLECT")
            y = tf.instance_norm(tf.nn.conv2d(y, dim, ks, s, padding='VALID', name=name+'_c1'), name+'_bn1')
            y = tf.pad(tf.nn.relu(y), [[0, 0], [p, p], [p, p], [0, 0]], "REFLECT")
            y = tf.instance_norm(tf.nn.conv2d(y, dim, ks, s, padding='VALID', name=name+'_c2'), name+'_bn2')
            return y + x

        # Justin Johnson's model from https://github.com/jcjohnson/fast-neural-style/
        # The network with 9 blocks consists of: c7s1-32, d64, d128, R128, R128, R128,
        # R128, R128, R128, R128, R128, R128, u64, u32, c7s1-3
        c0 = tf.pad(image, [[0, 0], [3, 3], [3, 3], [0, 0]], "REFLECT")
        c1 = tf.nn.relu(tf.instance_norm(tf.nn.conv2d(c0, options.gf_dim, 7, 1, padding='VALID', name='g_e1_c'), 'g_e1_bn'))
        c2 = tf.nn.relu(tf.instance_norm(tf.nn.conv2d(c1, options.gf_dim*2, 3, 2, name='g_e2_c'), 'g_e2_bn'))
        c3 = tf.nn.relu(tf.instance_norm(tf.nn.conv2d(c2, options.gf_dim*4, 3, 2, name='g_e3_c'), 'g_e3_bn'))
        # define G network with 9 resnet blocks
        r1 = residule_block(c3, options.gf_dim*4, name='g_r1')
        r2 = residule_block(r1, options.gf_dim*4, name='g_r2')
        r3 = residule_block(r2, options.gf_dim*4, name='g_r3')
        r4 = residule_block(r3, options.gf_dim*4, name='g_r4')
        r5 = residule_block(r4, options.gf_dim*4, name='g_r5')
        r6 = residule_block(r5, options.gf_dim*4, name='g_r6')
        #r7 = residule_block(r6, options.gf_dim*4, name='g_r7')
        #r8 = residule_block(r7, options.gf_dim*4, name='g_r8')
        #r9 = residule_block(r8, options.gf_dim*4, name='g_r9')

        d1 = tf.nn.deconv2d(r6, options.gf_dim*2, 3, 2, name='g_d1_dc')
        d1 = tf.nn.relu(tf.instance_norm(d1, 'g_d1_bn'))
        d2 = tf.nn.deconv2d(d1, options.gf_dim, 3, 2, name='g_d2_dc')
        d2 = tf.nn.relu(tf.instance_norm(d2, 'g_d2_bn'))
        d2 = tf.pad(d2, [[0, 0], [3, 3], [3, 3], [0, 0]], "REFLECT")
        pred = tf.nn.tanh(tf.conv2d(d2, options.output_c_dim, 7, 1, padding='VALID', name='g_pred_c'))

        return pred
```

tf.pad

paddings : [n, 2] 의 shape이어야 한다.

paddings[D, 0] + tensor.dim_size(D) + paddings[D, 1]

```
# 't'는 [[1, 2, 3], [4, 5, 6]].
# 'paddings'는 [[1, 1,], [2, 2]].
# 't'의 랭크(rank)는 2.
```

```
pad(t, paddings, "CONSTANT") ==> [[0, 0, 0, 0, 0, 0, 0],
                                  [0, 0, 1, 2, 3, 0, 0],
                                  [0, 0, 4, 5, 6, 0, 0],
                                  [0, 0, 0, 0, 0, 0, 0]]

pad(t, paddings, "REFLECT") ==> [[6, 5, 4, 5, 6, 5, 4],
                                 [3, 2, 1, 2, 3, 2, 1],
                                 [6, 5, 4, 5, 6, 5, 4],
                                 [3, 2, 1, 2, 3, 2, 1]]

pad(t, paddings, "SYMMETRIC") ==> [[2, 1, 1, 2, 3, 3, 2],
                                   [2, 1, 1, 2, 3, 3, 2],
                                   [5, 4, 4, 5, 6, 6, 5],
                                   [5, 4, 4, 5, 6, 6, 5]]
```

tf.pad

```
pad(t, paddings, "CONSTANT") ==> [[0, 0, 0, 0, 0, 0, 0],
                                  [0, 0, 1, 2, 3, 0, 0],
                                  [0, 0, 4, 5, 6, 0, 0],
                                  [0, 0, 0, 0, 0, 0, 0]]

pad(t, paddings, "REFLECT") ==> [[6, 5, 4, 5, 6, 5, 4],
                                 [3, 2, 1, 2, 3, 2, 1],
                                 [6, 5, 4, 5, 6, 5, 4],
                                 [3, 2, 1, 2, 3, 2, 1]]

pad(t, paddings, "SYMMETRIC") ==> [[2, 1, 1, 2, 3, 3, 2],
                                   [2, 1, 1, 2, 3, 3, 2],
                                   [5, 4, 4, 5, 6, 6, 5],
                                   [5, 4, 4, 5, 6, 6, 5]]
```

instance normalization

Batch normalization

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \mu_i = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_i)^2.$$

Instance normalization

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2.$$

Discriminator architectures For discriminator networks, we use 70×70 PatchGAN [21]. Let `Ck` denote a 4×4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. After the last layer, we apply a convolution to produce a 1 dimensional output. We do not use InstanceNorm for the first `C64` layer. We use leaky ReLUs with slope 0.2. The discriminator architecture is:

`C64-C128-C256-C512`

```python
def discriminator_tf(image, options, reuse=False, name="discriminator"):

    with tf.variable_scope(name):
        # image is 256 x 256 x input_c_dim
        if reuse:
            tf.get_variable_scope().reuse_variables()
        else:
            assert tf.get_variable_scope().reuse is False

        h0 = tf.lrelu(tf.nn.conv2d(image, options.df_dim, name='d_h0_conv'))
        # h0 is (128 x 128 x self.df_dim)
        h1 = tf.lrelu(tf.instance_norm(tf.nn.conv2d(h0, options.df_dim*2, name='d_h1_conv'), 'd_bn1'))
        # h1 is (64 x 64 x self.df_dim*2)
        h2 = tf.lrelu(tf.instance_norm(tf.nn.conv2d(h1, options.df_dim*4, name='d_h2_conv'), 'd_bn2'))
        # h2 is (32x 32 x self.df_dim*4)
        h3 = tf.lrelu(tf.instance_norm(tf.nn.conv2d(h2, options.df_dim*8, s=1, name='d_h3_conv'), 'd_bn3'))
        # h3 is (32 x 32 x self.df_dim*8)
        h4 = tf.nn.conv2d(h3, 1, s=1, name='d_h3_pred')
        # h4 is (32 x 32 x 1)
        return h4
```

END