생성적 적대 신경망 (GAN)

양승우

GAN 훈련

라이브러리 임포트

In [1]:
```
import os
import matplotlib.pyplot as plt

from models.GAN import GAN
from utils.loaders import load_safari
```

```
Using TensorFlow backend.
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:526: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:527: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:528: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:529: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:530: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:535: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:
```
# run params
SECTION = 'gan'
RUN_ID = '0001'
DATA_NAME = 'camel'
RUN_FOLDER = 'run/{}/'.format(SECTION)
RUN_FOLDER += '_'.join([RUN_ID, DATA_NAME])

if not os.path.exists(RUN_FOLDER):
    os.mkdir(RUN_FOLDER)
    os.mkdir(os.path.join(RUN_FOLDER, 'viz'))
    os.mkdir(os.path.join(RUN_FOLDER, 'images'))
    os.mkdir(os.path.join(RUN_FOLDER, 'weights'))

mode =  'build' #'load' #
```

여기서 아래와 같은 오류가 발생할 수 있는데, (다음 슬라이드 참고)

이는 data 파일이 없어서 발생하는 오류로 GDL_code 안에 다음과 같은 파일들을 복사 후에

현재 생성한 .ipynb과 같은 경로에 둡니다.

데이터 적재

깃허브에 camel 데이터셋이 포함되어 있으므로 별도로 다운로드 받을 필요가 없습니다.

In [3]:
```
(x_train, y_train) = load_safari(DATA_NAME)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-3-595832a29a8f> in <module>
----> 1 (x_train, y_train) = load_safari(DATA_NAME)

~₩GAN₩utils₩loaders.py in load_safari(folder)
    188                 break
    189
--> 190     slice_train = int(80000/len(txt_name_list))  ###Setting value to be 80000 for the final dataset
    191     i = 0
    192     seed = np.random.randint(1, 10e6)

ZeroDivisionError: division by zero
```

데이터 적재

깃허브에 camel 데이터셋이 포함되어 있으므로 별도로 다운로드 받을 필요가 없습니다.

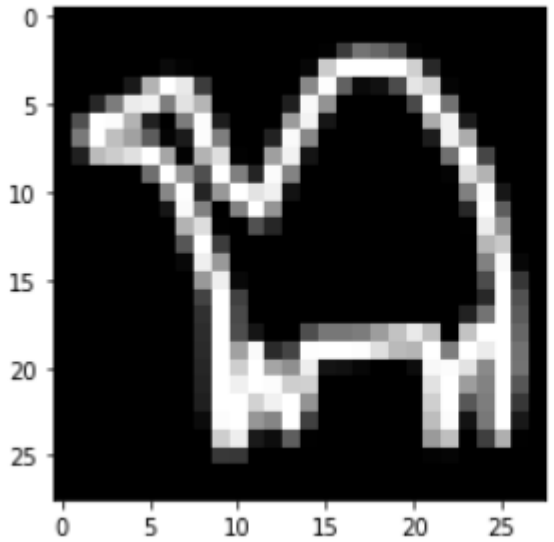
In [3]:
```
(x_train, y_train) = load_safari(DATA_NAME)
```

In [4]:
```
x_train.shape
```

Out[4]:
```
(80000, 28, 28, 1)
```

In [5]:
```
plt.imshow(x_train[200,:,:,0], cmap = 'gray')
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x206006b2668>
```



모델 만들기

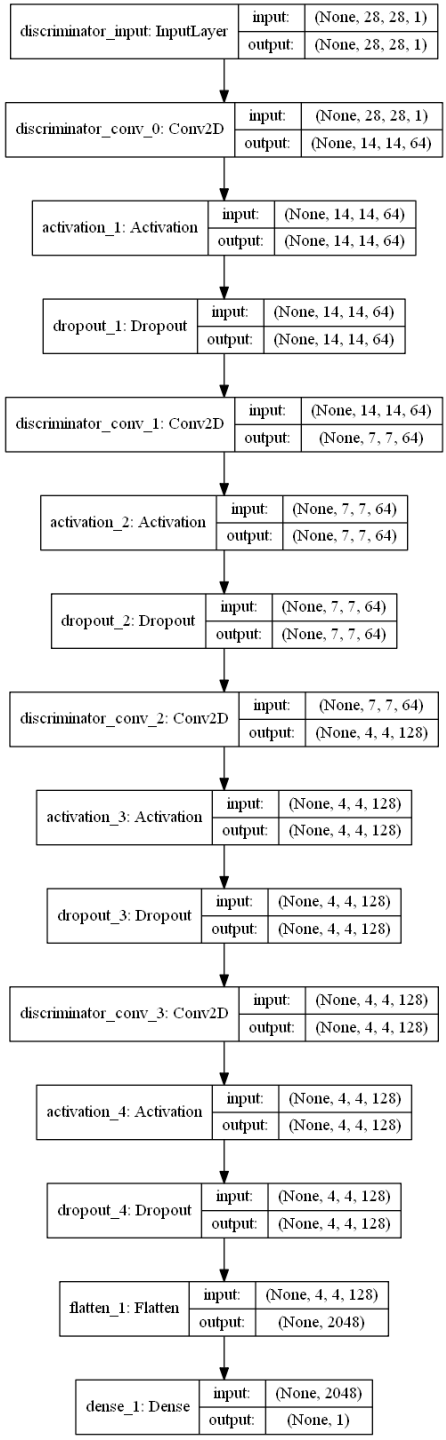
In [6]:
```
gan = GAN(input_dim = (28,28,1)
        , discriminator_conv_filters = [64,64,128,128]
        , discriminator_conv_kernel_size = [5,5,5,5]
        , discriminator_conv_strides = [2,2,2,1]
        , discriminator_batch_norm_momentum = None
        , discriminator_activation = 'relu'
        , discriminator_dropout_rate = 0.4
        , discriminator_learning_rate = 0.0008
        , generator_initial_dense_layer_size = (7, 7, 64)
        , generator_upsample = [2,2, 1, 1]
        , generator_conv_filters = [128,64, 64,1]
        , generator_conv_kernel_size = [5,5,5,5]
        , generator_conv_strides = [1,1, 1, 1]
        , generator_batch_norm_momentum = 0.9
        , generator_activation = 'relu'
        , generator_dropout_rate = None
        , generator_learning_rate = 0.0004
        , optimiser = 'rmsprop'
        , z_dim = 100
        )

if mode == 'build':
    gan.save(RUN_FOLDER)
else:
    gan.load_weights(os.path.join(RUN_FOLDER, 'weights/weights.h5'))
```

```
WARNING:tensorflow:From /home/luxmaris16/anaconda3/envs/testGPU/lib/python3.6/site-packages/tensorflow/python/framework/op_def_library.
py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /home/luxmaris16/anaconda3/envs/testGPU/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:3445: c
alling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

GAN discriminator

In [7]:
```
gan.discriminator.summary()
```

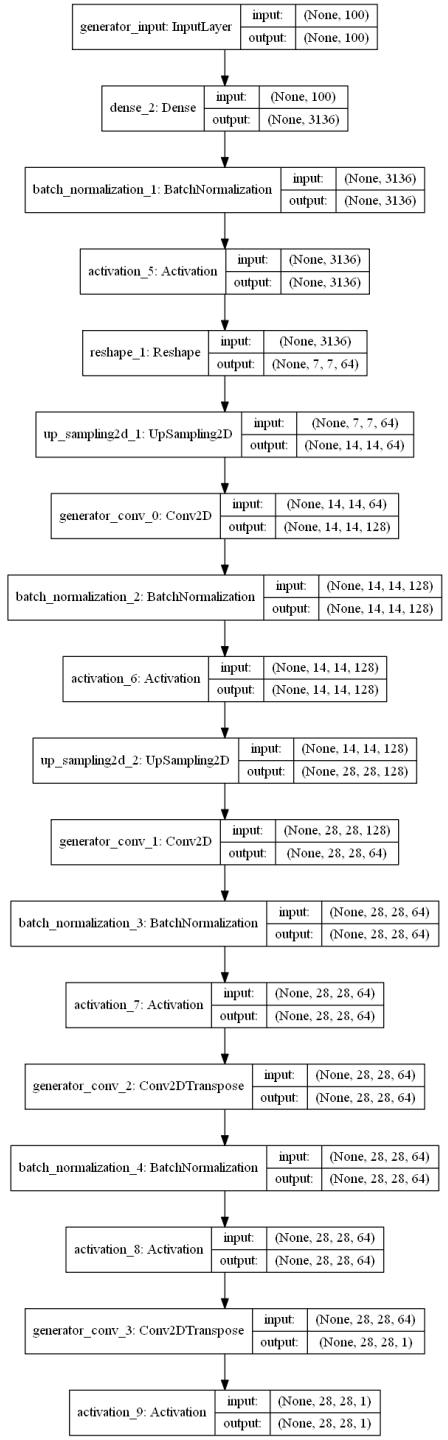
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
discriminator_input (InputLa (None, 28, 28, 1)         0
_________________________________________________________________
discriminator_conv_0 (Conv2D (None, 14, 14, 64)        1664
_________________________________________________________________
activation_1 (Activation)    (None, 14, 14, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
discriminator_conv_1 (Conv2D (None, 7, 7, 64)          102464
_________________________________________________________________
activation_2 (Activation)    (None, 7, 7, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
discriminator_conv_2 (Conv2D (None, 4, 4, 128)         204928
_________________________________________________________________
activation_3 (Activation)    (None, 4, 4, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
discriminator_conv_3 (Conv2D (None, 4, 4, 128)         409728
_________________________________________________________________
activation_4 (Activation)    (None, 4, 4, 128)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2049
=================================================================
Total params: 720,833
Trainable params: 720,833
Non-trainable params: 0
_________________________________________________________________
```



GAN generator

In [8]:
```
gan.generator.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
generator_input (InputLayer) (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3136)              316736
_________________________________________________________________
batch_normalization_1 (Batch (None, 3136)              12544
_________________________________________________________________
activation_5 (Activation)    (None, 3136)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 7, 7, 64)          0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 14, 14, 64)        0
_________________________________________________________________
generator_conv_0 (Conv2D)    (None, 14, 14, 128)       204928
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 128)       512
_________________________________________________________________
activation_6 (Activation)    (None, 14, 14, 128)       0
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 28, 28, 128)       0
_________________________________________________________________
generator_conv_1 (Conv2D)    (None, 28, 28, 64)        204864
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 64)        256
_________________________________________________________________
activation_7 (Activation)    (None, 28, 28, 64)        0
_________________________________________________________________
generator_conv_2 (Conv2DTran (None, 28, 28, 64)        102464
_________________________________________________________________
batch_normalization_4 (Batch (None, 28, 28, 64)        256
_________________________________________________________________
activation_8 (Activation)    (None, 28, 28, 64)        0
_________________________________________________________________
generator_conv_3 (Conv2DTran (None, 28, 28, 1)         1601
_________________________________________________________________
activation_9 (Activation)    (None, 28, 28, 1)         0
=================================================================
Total params: 844,161
Trainable params: 837,377
Non-trainable params: 6,784
_________________________________________________________________
```



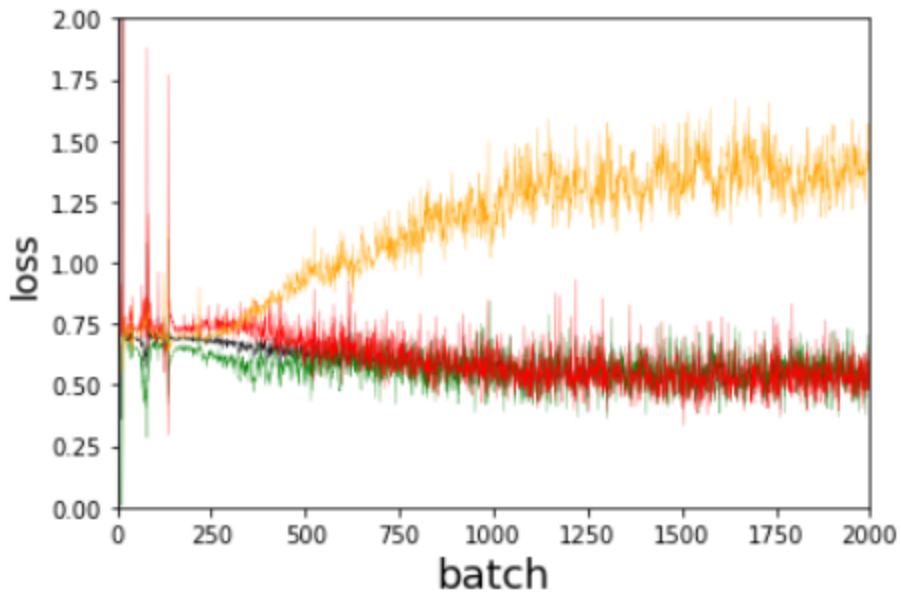
모델 훈련

In [9]:
```
BATCH_SIZE = 64
EPOCHS = 6000
PRINT_EVERY_N_BATCHES = 5
```

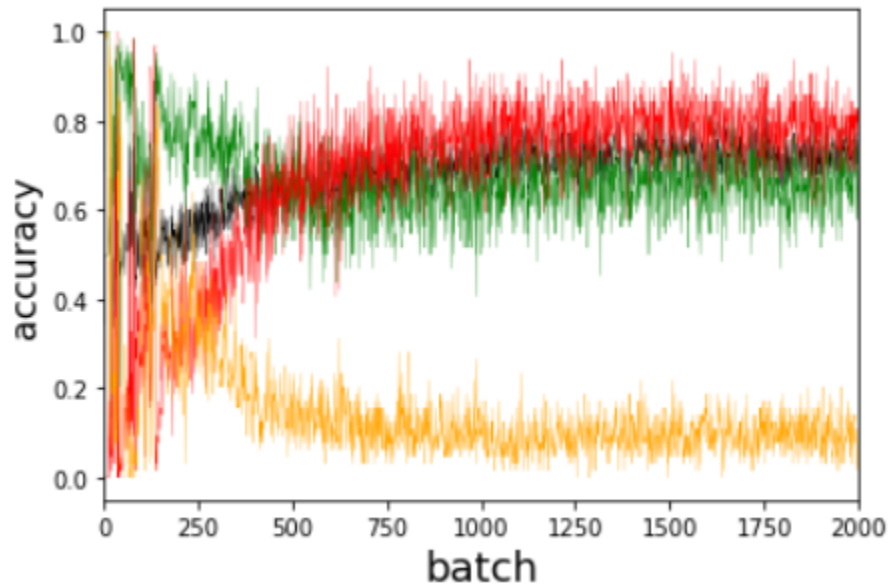
In [10]:
```
gan.train(
    x_train
    , batch_size = BATCH_SIZE
    , epochs = EPOCHS
    , run_folder = RUN_FOLDER
    , print_every_n_batches = PRINT_EVERY_N_BATCHES
)
```

```
WARNING:tensorflow:From /home/luxmaris16/anaconda3/envs/testGPU/lib/python3.6/site-packages/tensorflow/python/ops/math_ops.py:3066:
to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

/home/luxmaris16/anaconda3/envs/testGPU/lib/python3.6/site-packages/keras/engine/training.py:490: UserWarning: Discrepancy between t
rainable weights and collected trainable weights, did you set `model.trainable` without calling `model.compile` after ?
  'Discrepancy between trainable weights and collected trainable'

0 [D loss: (0.743)(R 0.689, F 0.798)] [D acc: (0.344)(0.688, 0.000)] [G loss: 0.678] [G acc: 1.000]
1 [D loss: (0.969)(R 0.661, F 1.277)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.689] [G acc: 1.000]
2 [D loss: (0.694)(R 0.687, F 0.700)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.689] [G acc: 1.000]
3 [D loss: (0.693)(R 0.686, F 0.699)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.689] [G acc: 1.000]
4 [D loss: (0.693)(R 0.686, F 0.700)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.688] [G acc: 1.000]
5 [D loss: (0.691)(R 0.683, F 0.700)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.689] [G acc: 1.000]
6 [D loss: (0.690)(R 0.677, F 0.703)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.690] [G acc: 0.984]
7 [D loss: (0.688)(R 0.667, F 0.709)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.689] [G acc: 0.984]
8 [D loss: (0.682)(R 0.644, F 0.720)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.688] [G acc: 1.000]
9 [D loss: (0.656)(R 0.598, F 0.715)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.688] [G acc: 1.000]
10 [D loss: (0.656)(R 0.399, F 0.912)] [D acc: (0.500)(1.000, 0.000)] [G loss: 0.689] [G acc: 1.000]
```

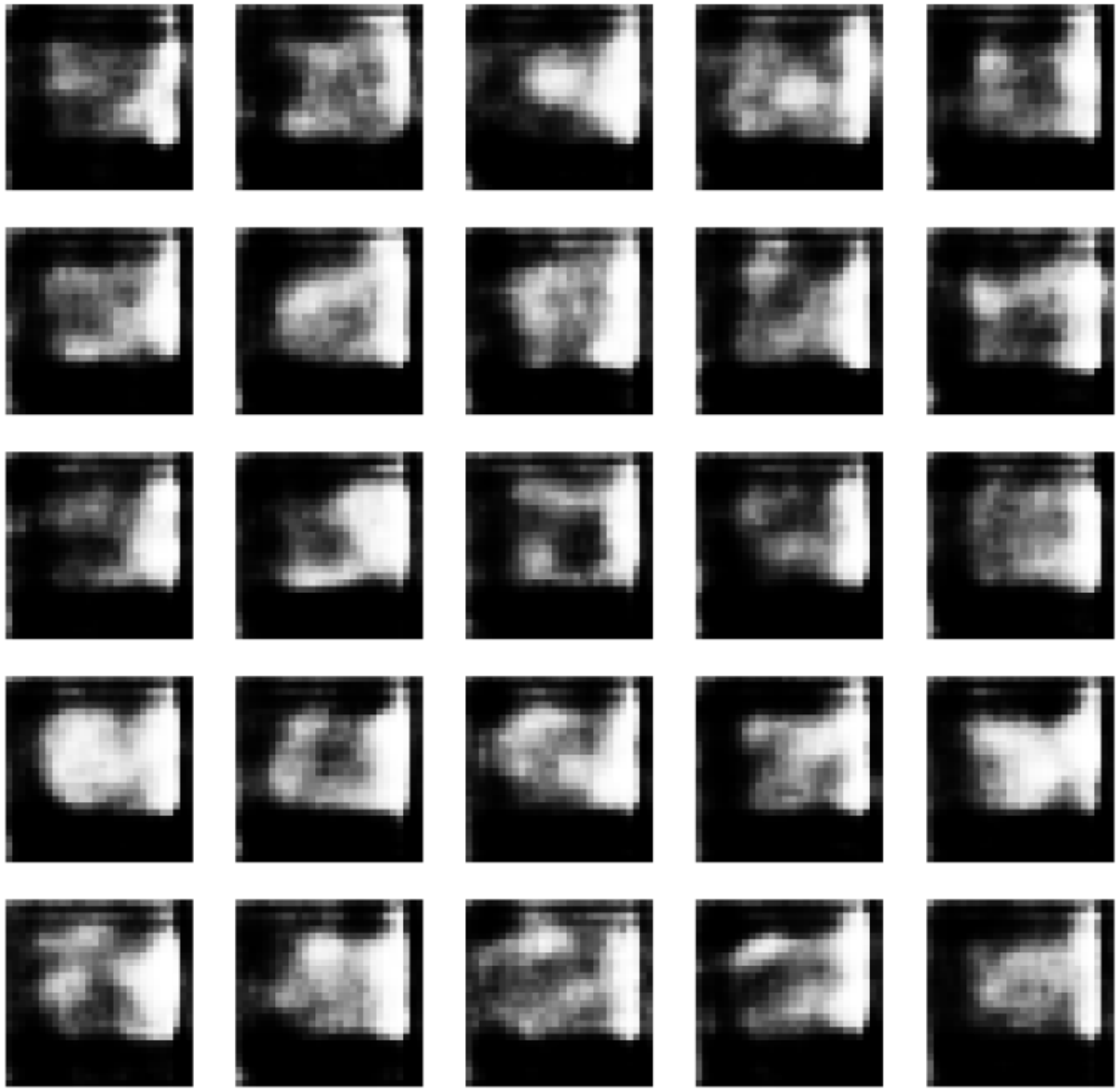
In [11]:
```
fig = plt.figure()
plt.plot([x[0] for x in gan.d_losses], color='black', linewidth=0.25)

plt.plot([x[1] for x in gan.d_losses], color='green', linewidth=0.25)
plt.plot([x[2] for x in gan.d_losses], color='red', linewidth=0.25)
plt.plot([x[0] for x in gan.g_losses], color='orange', linewidth=0.25)

plt.xlabel('batch', fontsize=18)
plt.ylabel('loss', fontsize=16)

plt.xlim(0, 2000)
plt.ylim(0, 2)

plt.show()
```



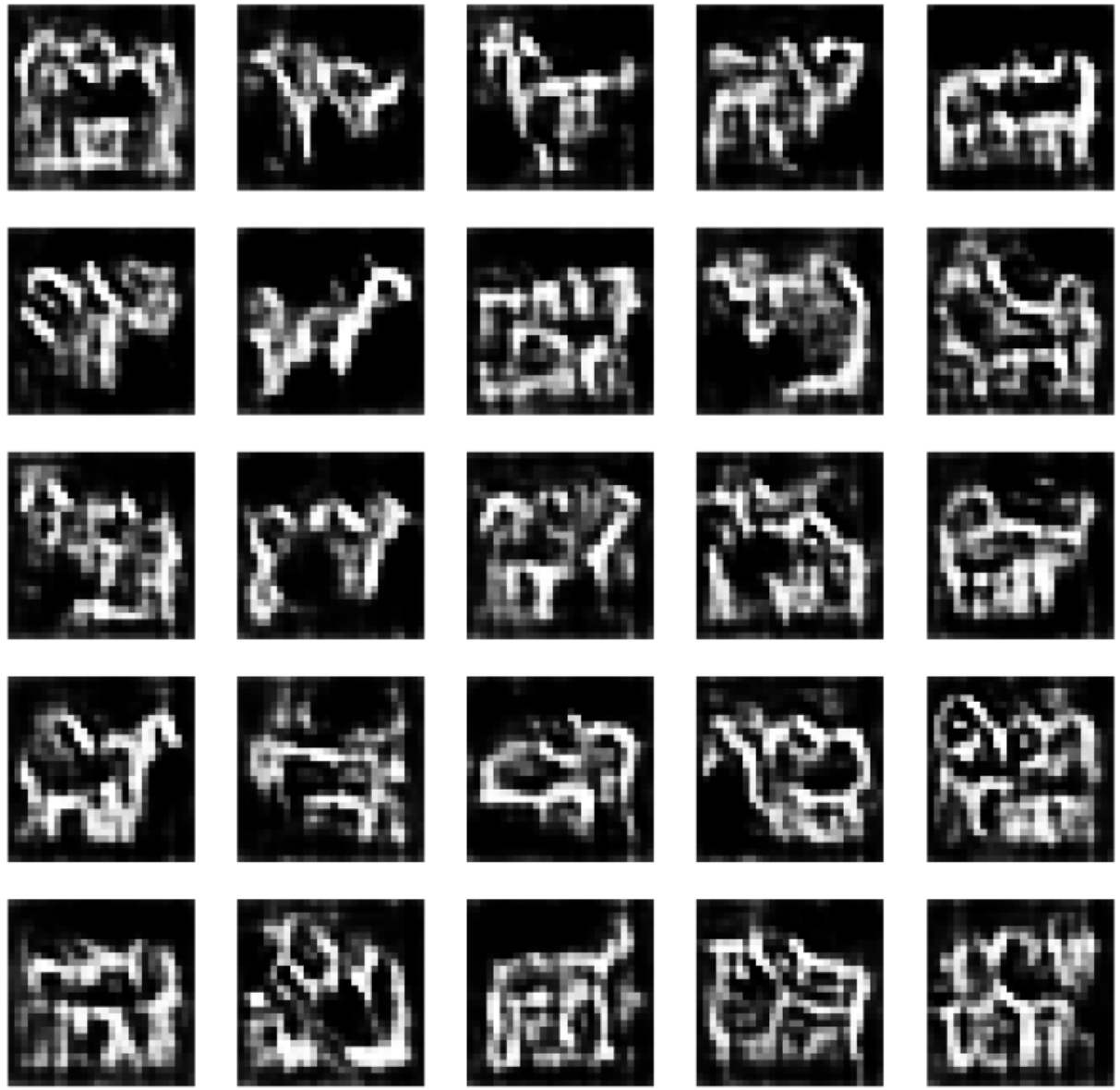
In [12]:
```
fig = plt.figure()
plt.plot([x[3] for x in gan.d_losses], color='black', linewidth=0.25)
plt.plot([x[4] for x in gan.d_losses], color='green', linewidth=0.25)
plt.plot([x[5] for x in gan.d_losses], color='red', linewidth=0.25)
plt.plot([x[1] for x in gan.g_losses], color='orange', linewidth=0.25)

plt.xlabel('batch', fontsize=18)
plt.ylabel('accuracy', fontsize=16)

plt.xlim(0, 2000)

plt.show()
```



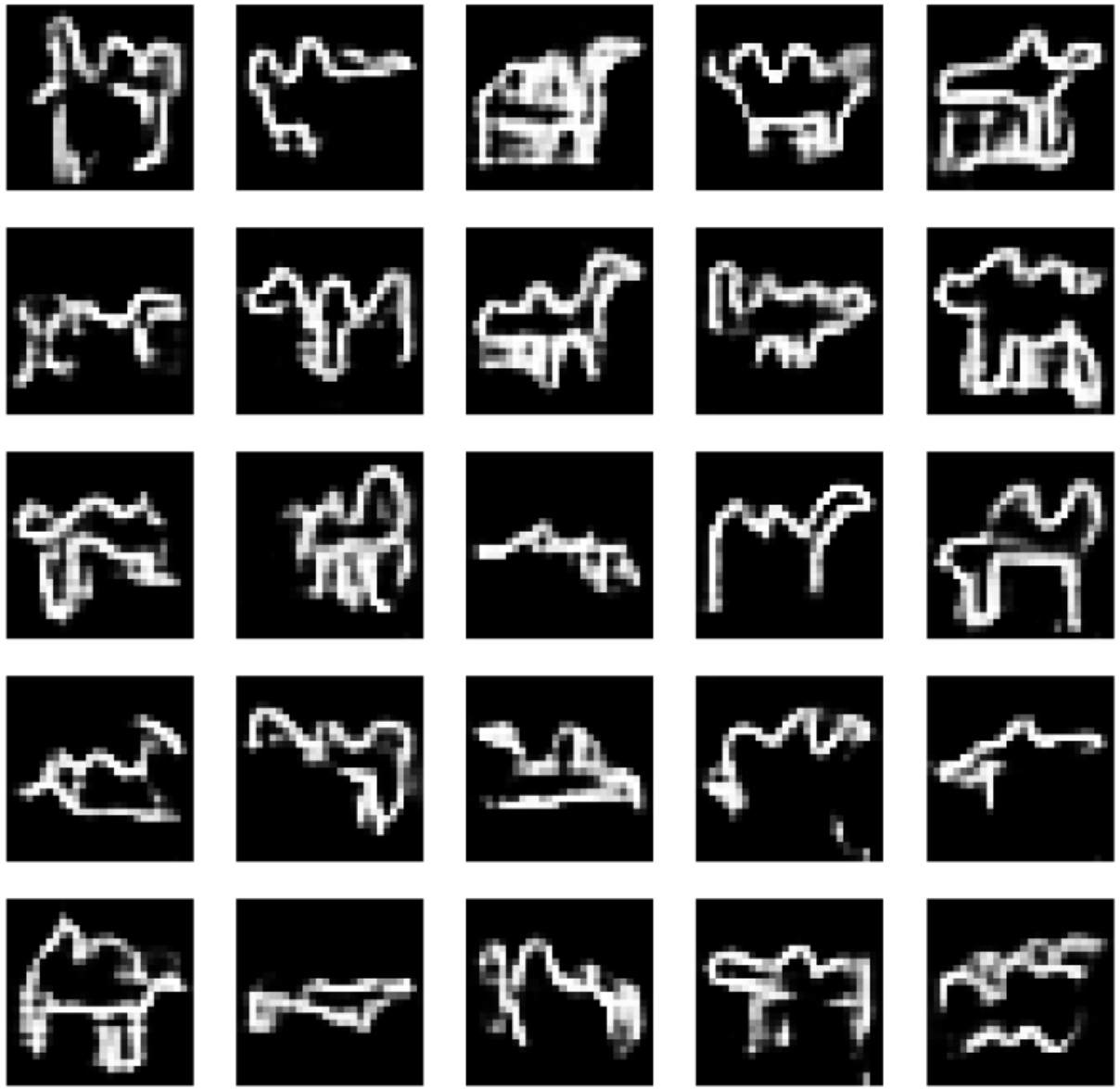
에포크 20



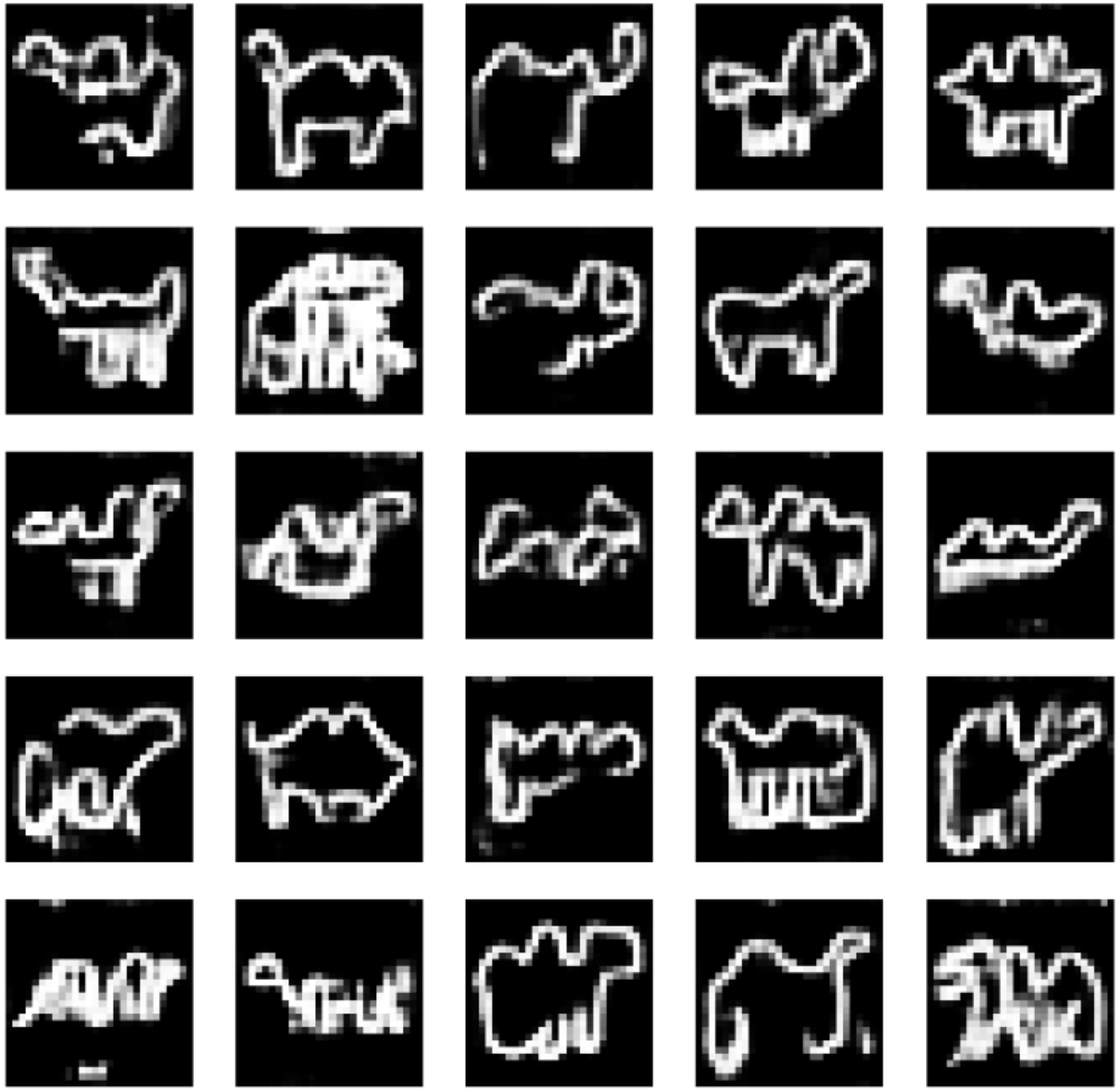
에포크 200



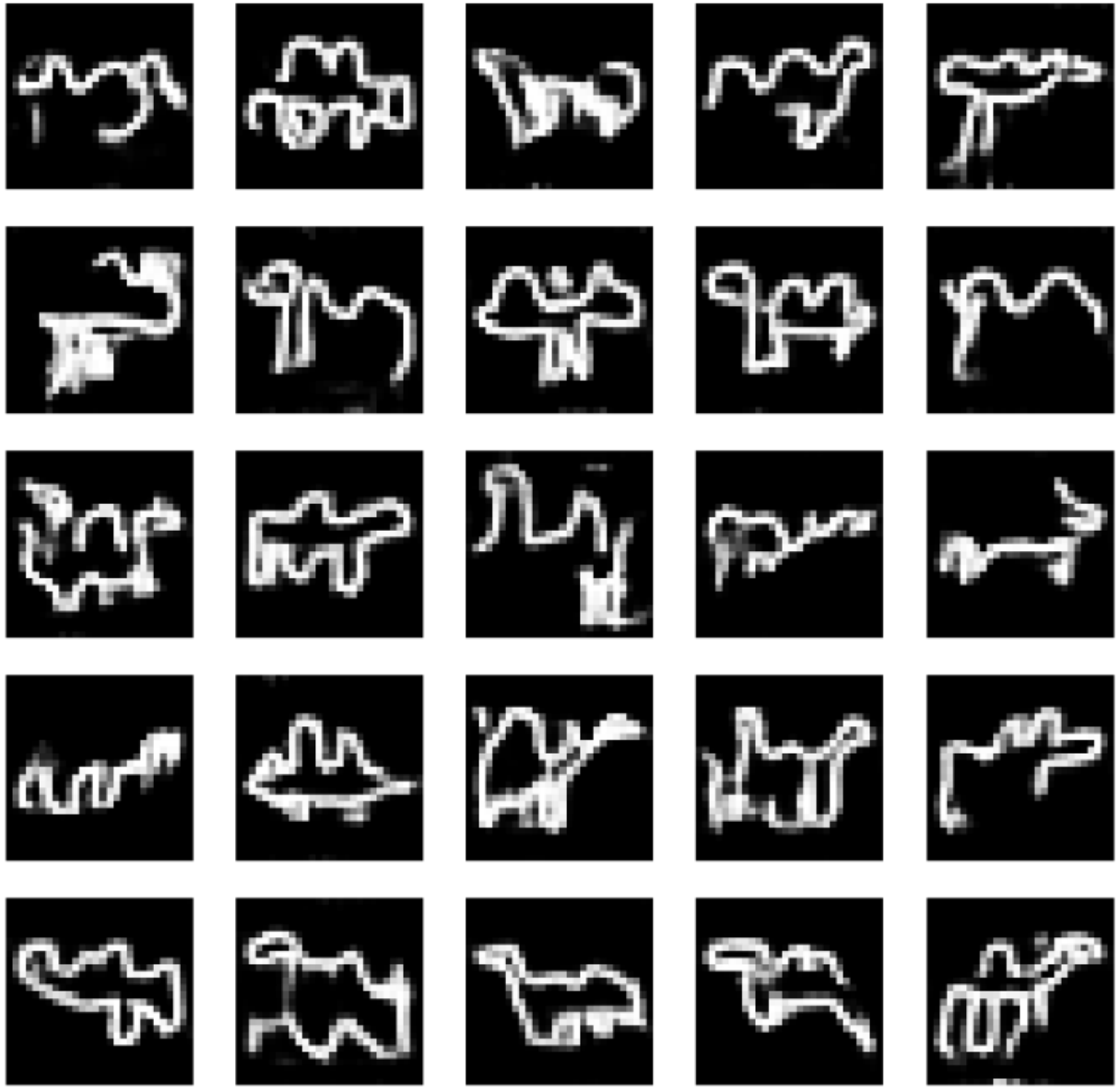
에포크 1000



에포크 2000



에포크 5995



WGAN 훈련

라이브러리 임포트

In [1]:
```
%matplotlib inline

import os
import numpy as np
import matplotlib.pyplot as plt

from models.WGAN import WGAN
from utils.loaders import load_cifar
```

```
Using TensorFlow backend.
/home/luxmaris16/anaconda3/envs/testGAN/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/luxmaris16/anaconda3/envs/testGAN/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:527: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/luxmaris16/anaconda3/envs/testGAN/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:528: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/luxmaris16/anaconda3/envs/testGAN/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:529: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/luxmaris16/anaconda3/envs/testGAN/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:530: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/luxmaris16/anaconda3/envs/testGAN/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:535: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:
```
# run params
SECTION = 'gan'
RUN_ID = '0002'
DATA_NAME = 'dog'
RUN_FOLDER = 'run/{}/'.format(SECTION)
RUN_FOLDER += '_'.join([RUN_ID, DATA_NAME])

if not os.path.exists(RUN_FOLDER):
    os.mkdir(RUN_FOLDER)
    os.mkdir(os.path.join(RUN_FOLDER, 'viz'))
    os.mkdir(os.path.join(RUN_FOLDER, 'images'))
    os.mkdir(os.path.join(RUN_FOLDER, 'weights'))

mode =  'build' #'load' #
```

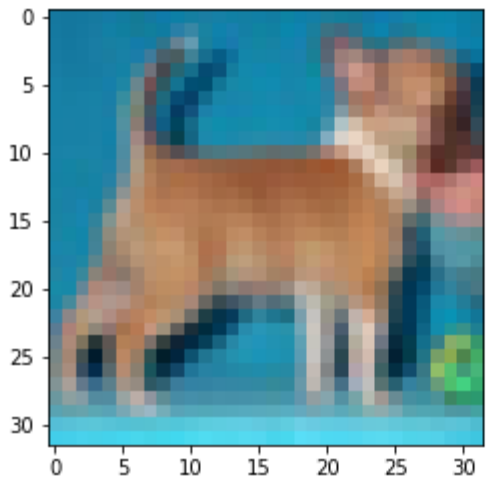
데이터 적재

In [3]:
```
if DATA_NAME == 'cars':
    label = 1
elif DATA_NAME == 'dog':
    label = 5
(x_train, y_train) = load_cifar(label, 10)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 3s 0us/step
```

In [4]:
```
plt.imshow((x_train[150,:,:,:]+1)/2)
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x7f578fdd60f0>
```



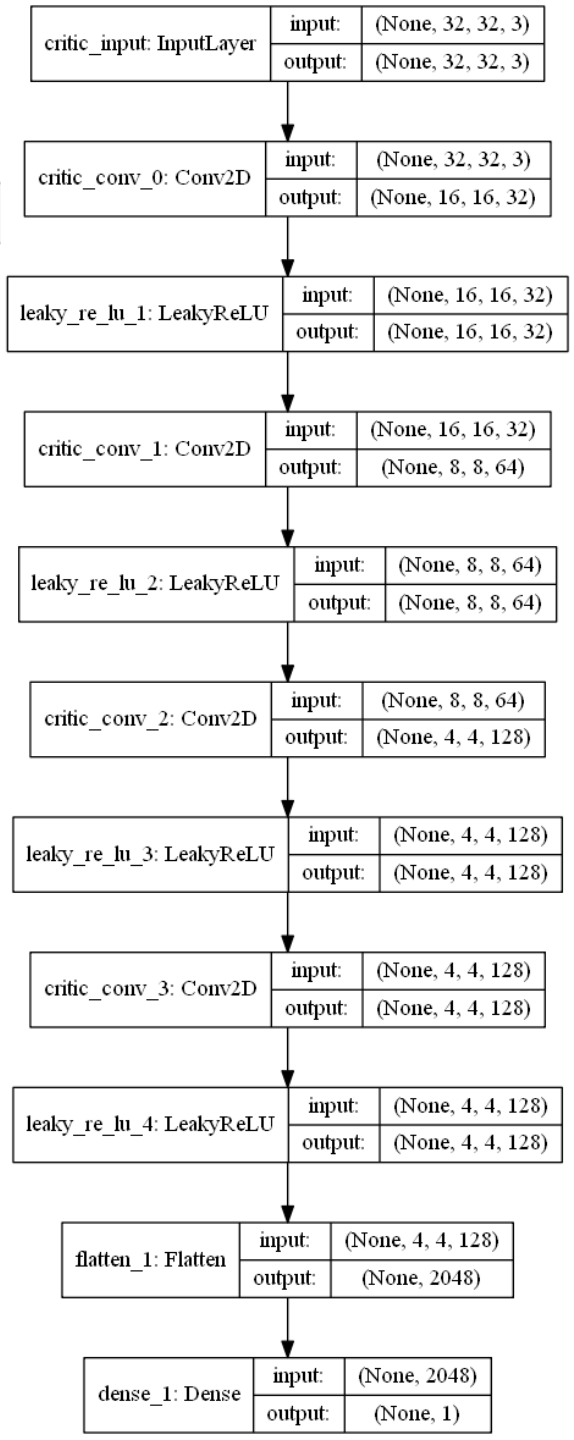
모델 생성

In [5]:
```
if mode == 'build':

    gan = WGAN(input_dim = (32,32,3)
            , critic_conv_filters = [32,64,128,128]
            , critic_conv_kernel_size = [5,5,5,5]
            , critic_conv_strides = [2,2,2,1]
            , critic_batch_norm_momentum = None
            , critic_activation = 'leaky_relu'
            , critic_dropout_rate = None
            , critic_learning_rate = 0.00005
            , generator_initial_dense_layer_size = (4, 4, 128)
            , generator_upsample = [2,2, 2,1]
            , generator_conv_filters = [128,64,32,3]
            , generator_conv_kernel_size = [5,5,5,5]
            , generator_conv_strides = [1,1, 1,1]
            , generator_batch_norm_momentum = 0.8
            , generator_activation = 'leaky_relu'
            , generator_dropout_rate = None
            , generator_learning_rate = 0.00005
            , optimiser = 'rmsprop'
            , z_dim = 100
            )
    gan.save(RUN_FOLDER)

else:
    gan.load_weights(os.path.join(RUN_FOLDER, 'weights/weights.h5'))
```

```
WARNING:tensorflow:From /home/haesun/github/GDL_code/env/lib/python3.7/site-packages/tensorflow/python/framework/op_def_library.py:263: col
ocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [6]:
```
gan.critic.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
critic_input (InputLayer)    (None, 32, 32, 3)         0
_________________________________________________________________
critic_conv_0 (Conv2D)       (None, 16, 16, 32)        2432
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 16, 16, 32)        0
_________________________________________________________________
critic_conv_1 (Conv2D)       (None, 8, 8, 64)          51264
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 8, 8, 64)          0
_________________________________________________________________
critic_conv_2 (Conv2D)       (None, 4, 4, 128)         204928
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 4, 4, 128)         0
_________________________________________________________________
critic_conv_3 (Conv2D)       (None, 4, 4, 128)         409728
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2049
=================================================================
Total params: 670,401
Trainable params: 670,401
Non-trainable params: 0
_________________________________________________________________
```



In [7]:
```
gan.generator.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
generator_input (InputLayer) (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2048)              206848
_________________________________________________________________
batch_normalization_1 (Batch (None, 2048)              8192
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 2048)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 128)         0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 8, 8, 128)         0
_________________________________________________________________
generator_conv_0 (Conv2D)    (None, 8, 8, 128)         409728
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 8, 8, 128)         0
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 128)       0
_________________________________________________________________
generator_conv_1 (Conv2D)    (None, 16, 16, 64)        204864
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 32, 32, 64)        0
_________________________________________________________________
generator_conv_2 (Conv2D)    (None, 32, 32, 32)        51232
_________________________________________________________________
batch_normalization_4 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 32, 32, 32)        0
_________________________________________________________________
generator_conv_3 (Conv2DTran (None, 32, 32, 3)         2403
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 3)         0
=================================================================
Total params: 884,163
Trainable params: 879,619
Non-trainable params: 4,544
_________________________________________________________________
```

모델 훈련

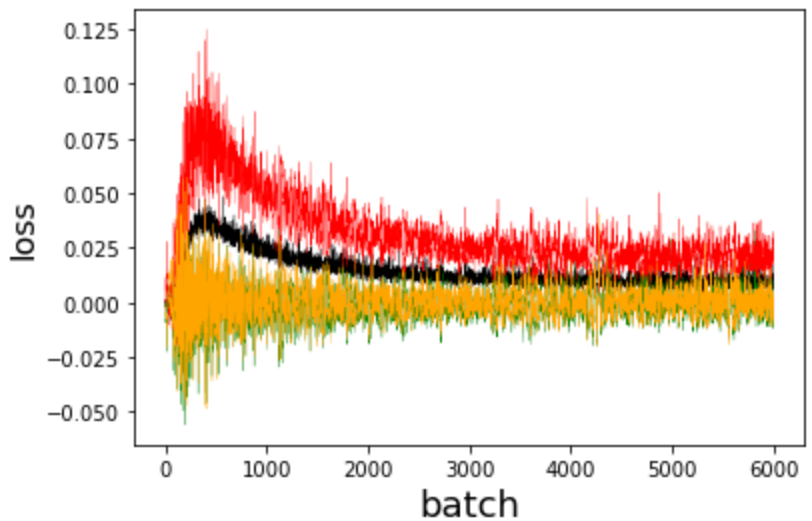
In [8]:
```
BATCH_SIZE = 128
EPOCHS = 6000
PRINT_EVERY_N_BATCHES = 5
N_CRITIC = 5
CLIP_THRESHOLD = 0.01
```

In [9]:
```
gan.train(
    x_train
    , batch_size = BATCH_SIZE
    , epochs = EPOCHS
    , run_folder = RUN_FOLDER
    , print_every_n_batches = PRINT_EVERY_N_BATCHES
    , n_critic = N_CRITIC
    , clip_threshold = CLIP_THRESHOLD
)
```

```
5981 [D loss: (0.009)(R -0.011, F 0.029)]  [G loss: -0.000]
5982 [D loss: (0.010)(R -0.009, F 0.030)]  [G loss: -0.006]
5983 [D loss: (0.010)(R -0.008, F 0.028)]  [G loss: -0.011]
5984 [D loss: (0.009)(R -0.008, F 0.027)]  [G loss: -0.005]
5985 [D loss: (0.013)(R -0.002, F 0.028)]  [G loss: -0.005]
5986 [D loss: (0.005)(R -0.012, F 0.023)]  [G loss: -0.003]
5987 [D loss: (0.011)(R -0.009, F 0.032)]  [G loss: -0.004]
5988 [D loss: (0.011)(R -0.002, F 0.023)]  [G loss: -0.002]
5989 [D loss: (0.008)(R -0.002, F 0.017)]  [G loss: 0.000]
5990 [D loss: (0.008)(R -0.002, F 0.018)]  [G loss: 0.005]
5991 [D loss: (0.017)(R 0.010, F 0.025)]  [G loss: -0.001]
5992 [D loss: (0.009)(R -0.006, F 0.025)]  [G loss: -0.008]
5993 [D loss: (0.009)(R -0.001, F 0.019)]  [G loss: 0.005]
5994 [D loss: (0.006)(R -0.008, F 0.020)]  [G loss: -0.002]
5995 [D loss: (0.010)(R -0.001, F 0.021)]  [G loss: -0.002]
5996 [D loss: (0.008)(R -0.005, F 0.022)]  [G loss: 0.003]
5997 [D loss: (0.011)(R -0.001, F 0.022)]  [G loss: 0.001]
5998 [D loss: (0.009)(R -0.001, F 0.019)]  [G loss: 0.001]
5999 [D loss: (0.008)(R -0.002, F 0.017)]  [G loss: -0.000]
```

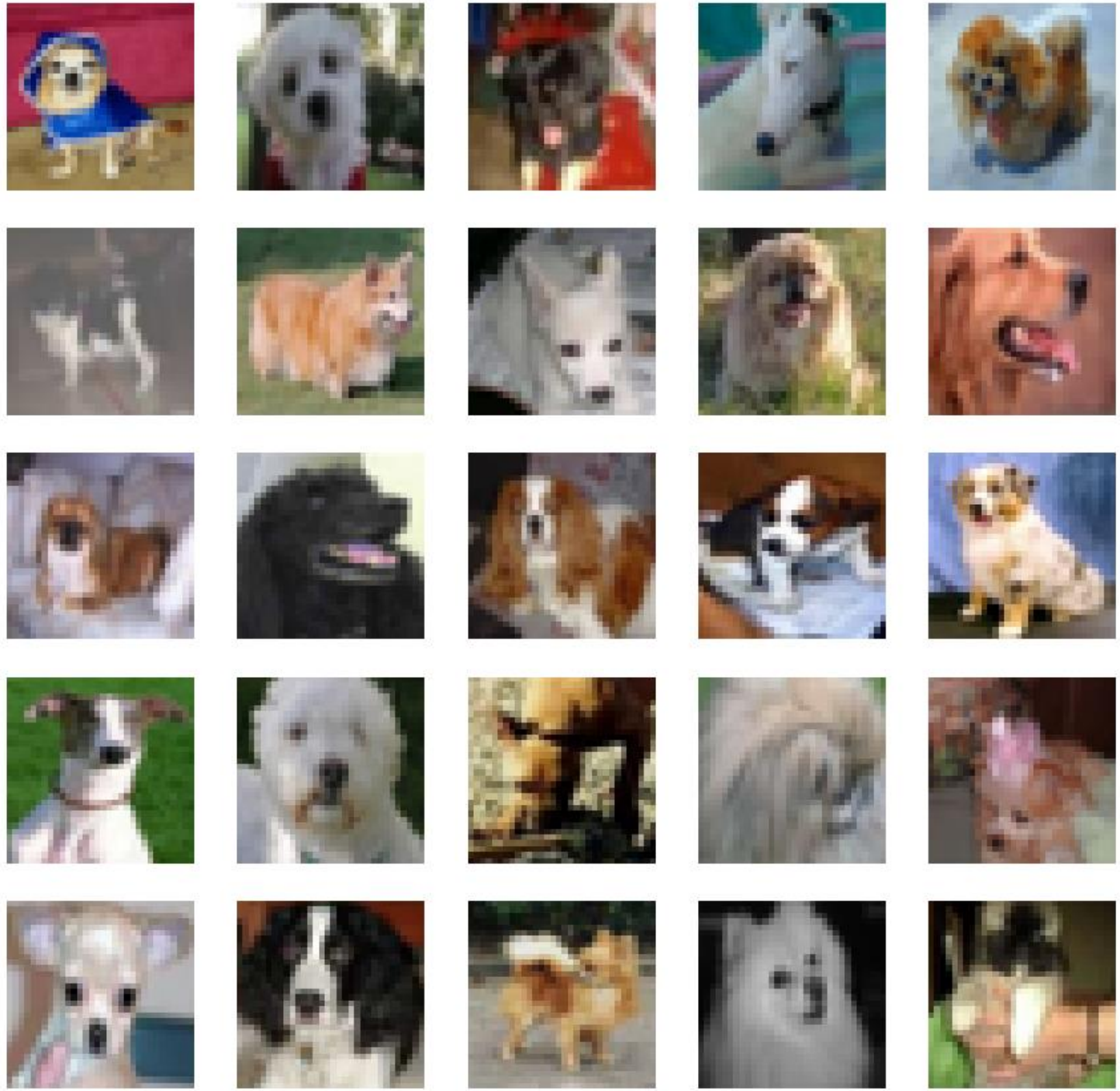
In [10]:
```
gan.sample_images(RUN_FOLDER)
```

In [11]:
```
fig = plt.figure()
plt.plot([x[0] for x in gan.d_losses], color='black', linewidth=0.25)

plt.plot([x[1] for x in gan.d_losses], color='green', linewidth=0.25)
plt.plot([x[2] for x in gan.d_losses], color='red', linewidth=0.25)
plt.plot(gan.g_losses, color='orange', linewidth=0.25)

plt.xlabel('batch', fontsize=18)
plt.ylabel('loss', fontsize=16)

# plt.xlim(0, 2000)
# plt.ylim(0, 2)

plt.show()
```

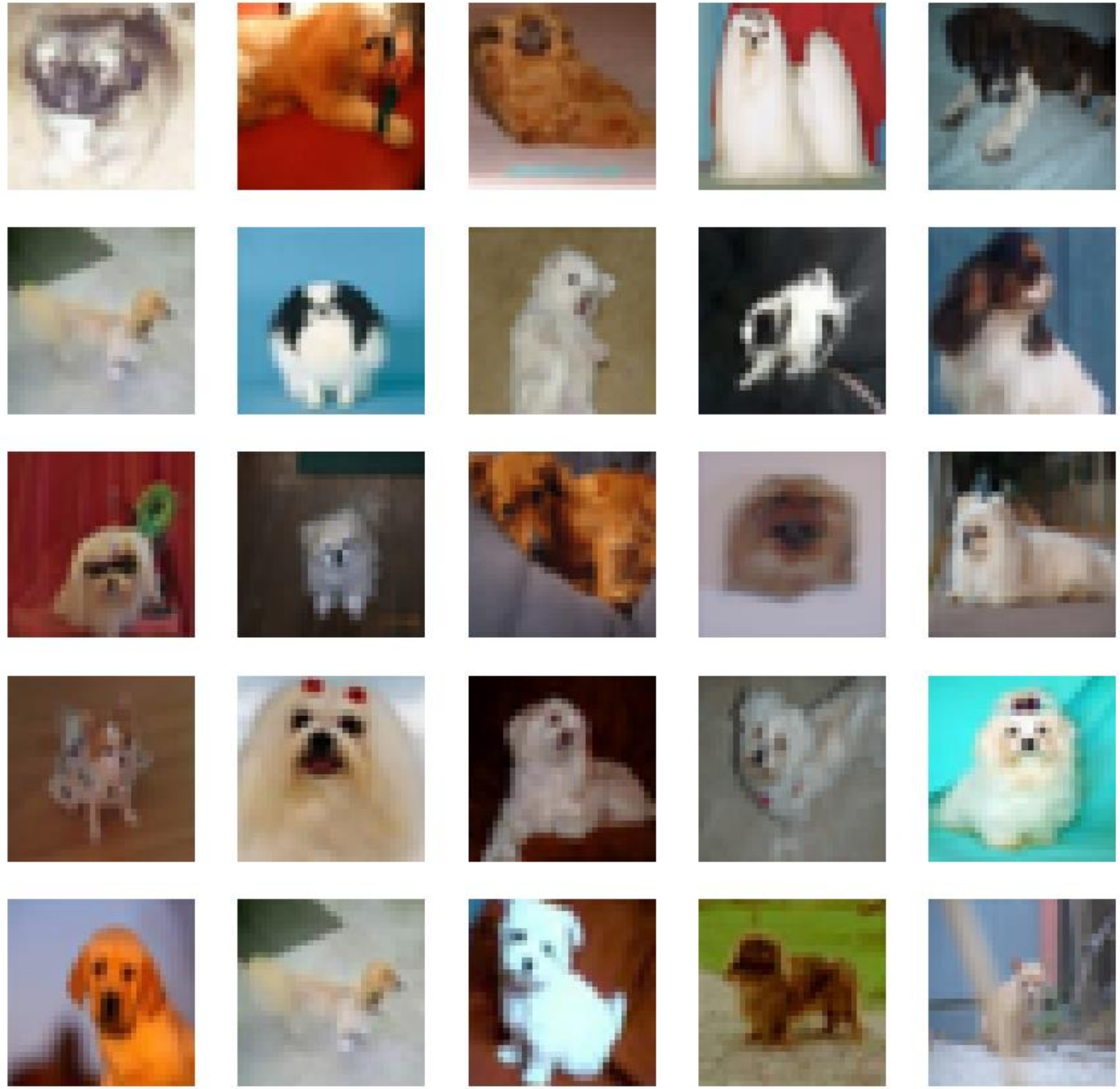


In [12]:
```
def compare_images(img1, img2):
    return np.mean(np.abs(img1 - img2))
```

In [15]:
```
r, c = 5, 5

idx = np.random.randint(0, x_train.shape[0], BATCH_SIZE)
true_imgs = (x_train[idx] + 1) *0.5

fig, axs = plt.subplots(r, c, figsize=(15,15))
cnt = 0

for i in range(r):
    for j in range(c):
        axs[i,j].imshow(true_imgs[cnt], cmap = 'gray_r')
        axs[i,j].axis('off')
        cnt += 1
fig.savefig(os.path.join(RUN_FOLDER, "images/real.png"))
plt.show()
```



In [16]:
```
r, c = 5, 5
noise = np.random.normal(0, 1, (r * c, gan.z_dim))
gen_imgs = gan.generator.predict(noise)

#Rescale images 0 - 1

gen_imgs = 0.5 * (gen_imgs + 1)
# gen_imgs = np.clip(gen_imgs, 0, 1)

fig, axs = plt.subplots(r, c, figsize=(15,15))
cnt = 0

for i in range(r):
    for j in range(c):
        axs[i,j].imshow(np.squeeze(gen_imgs[cnt, :,:,:]), cmap = 'gray_r')
        axs[i,j].axis('off')
        cnt += 1
fig.savefig(os.path.join(RUN_FOLDER, "images/sample.png"))
plt.close()


fig, axs = plt.subplots(r, c, figsize=(15,15))
cnt = 0

for i in range(r):
    for j in range(c):
        c_diff = 99999
        c_img = None
        for k_idx, k in enumerate((x_train + 1) * 0.5):

            diff = compare_images(gen_imgs[cnt, :,:,:], k)
            if diff < c_diff:
                c_img = np.copy(k)
                c_diff = diff
        axs[i,j].imshow(c_img, cmap = 'gray_r')
        axs[i,j].axis('off')
        cnt += 1

fig.savefig(os.path.join(RUN_FOLDER, "images/sample_closest.png"))
plt.show()
```



WGAN-GP 훈련

라이브러리 임포트

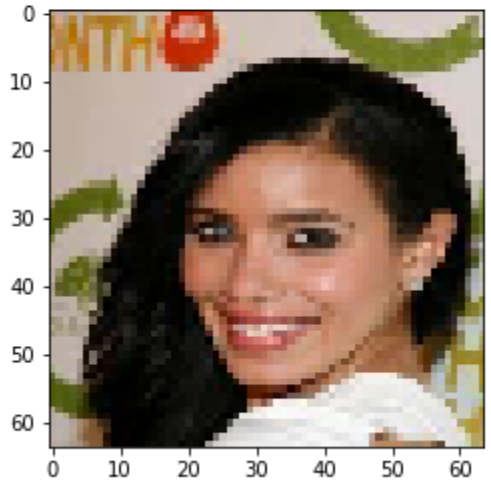
In [1]:
```
%matplotlib inline

import os
import matplotlib.pyplot as plt

from models.WGANGP import WGANGP
from utils.loaders import load_celeb

import pickle
```

```
Using TensorFlow backend.
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:526: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:527: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:528: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:529: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:530: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:₩Users₩User₩anaconda3₩envs₩testGAN₩lib₩site-packages₩tensorflow₩python₩framework₩dtypes.py:535: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:
```
# run params
SECTION = 'gan'
RUN_ID = '0003'
DATA_NAME = 'celeb'
RUN_FOLDER = 'run/{}/'.format(SECTION)
RUN_FOLDER += '_'.join([RUN_ID, DATA_NAME])

if not os.path.exists(RUN_FOLDER):
    os.mkdir(RUN_FOLDER)
    os.mkdir(os.path.join(RUN_FOLDER, 'viz'))
    os.mkdir(os.path.join(RUN_FOLDER, 'images'))
    os.mkdir(os.path.join(RUN_FOLDER, 'weights'))

mode =  'build' #'load' #
```

In [6]:
```
plt.imshow((x_train[0][0][0]+1)/2)
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x7f4cf948a400>
```



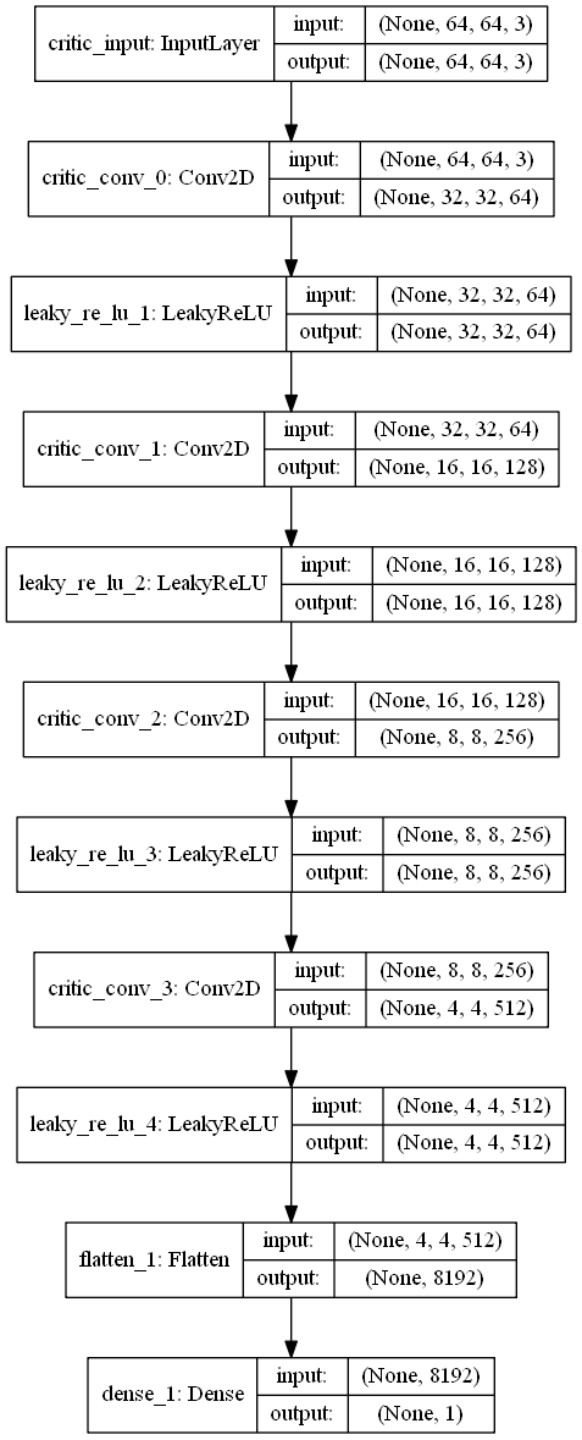
모델 생성

In [7]:
```
gan = WGANGP(input_dim = (IMAGE_SIZE,IMAGE_SIZE,3)
        , critic_conv_filters = [64,128,256,512]
        , critic_conv_kernel_size = [5,5,5,5]
        , critic_conv_strides = [2,2,2,2]
        , critic_batch_norm_momentum = None
        , critic_activation = 'leaky_relu'
        , critic_dropout_rate = None
        , critic_learning_rate = 0.0002
        , generator_initial_dense_layer_size = (4, 4, 512)
        , generator_upsample = [1,1,1,1]
        , generator_conv_filters = [256,128,64,3]
        , generator_conv_kernel_size = [5,5,5,5]
        , generator_conv_strides = [2,2,2,2]
        , generator_batch_norm_momentum = 0.9
        , generator_activation = 'leaky_relu'
        , generator_dropout_rate = None
        , generator_learning_rate = 0.0002
        , optimiser = 'adam'
        , grad_weight = 10
        , z_dim = 100
        , batch_size = BATCH_SIZE
        )

if mode == 'build':
    gan.save(RUN_FOLDER)

else:
    gan.load_weights(os.path.join(RUN_FOLDER, 'weights/weights.h5'))
```

```
WARNING:tensorflow:From /home/haesun/github/GDL_code/env/lib/python3.7/site-packages/tensorflow/python/framework/op_def_library.py:263: col
ocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [8]:
```
gan.critic.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
critic_input (InputLayer)    (None, 64, 64, 3)         0
_________________________________________________________________
critic_conv_0 (Conv2D)       (None, 32, 32, 64)        4864
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
critic_conv_1 (Conv2D)       (None, 16, 16, 128)       204928
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
critic_conv_2 (Conv2D)       (None, 8, 8, 256)         819456
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 8, 8, 256)         0
_________________________________________________________________
critic_conv_3 (Conv2D)       (None, 4, 4, 512)         3277312
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 4, 4, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 8193
=================================================================
Total params: 4,314,753
Trainable params: 4,314,753
Non-trainable params: 0
_________________________________________________________________
```



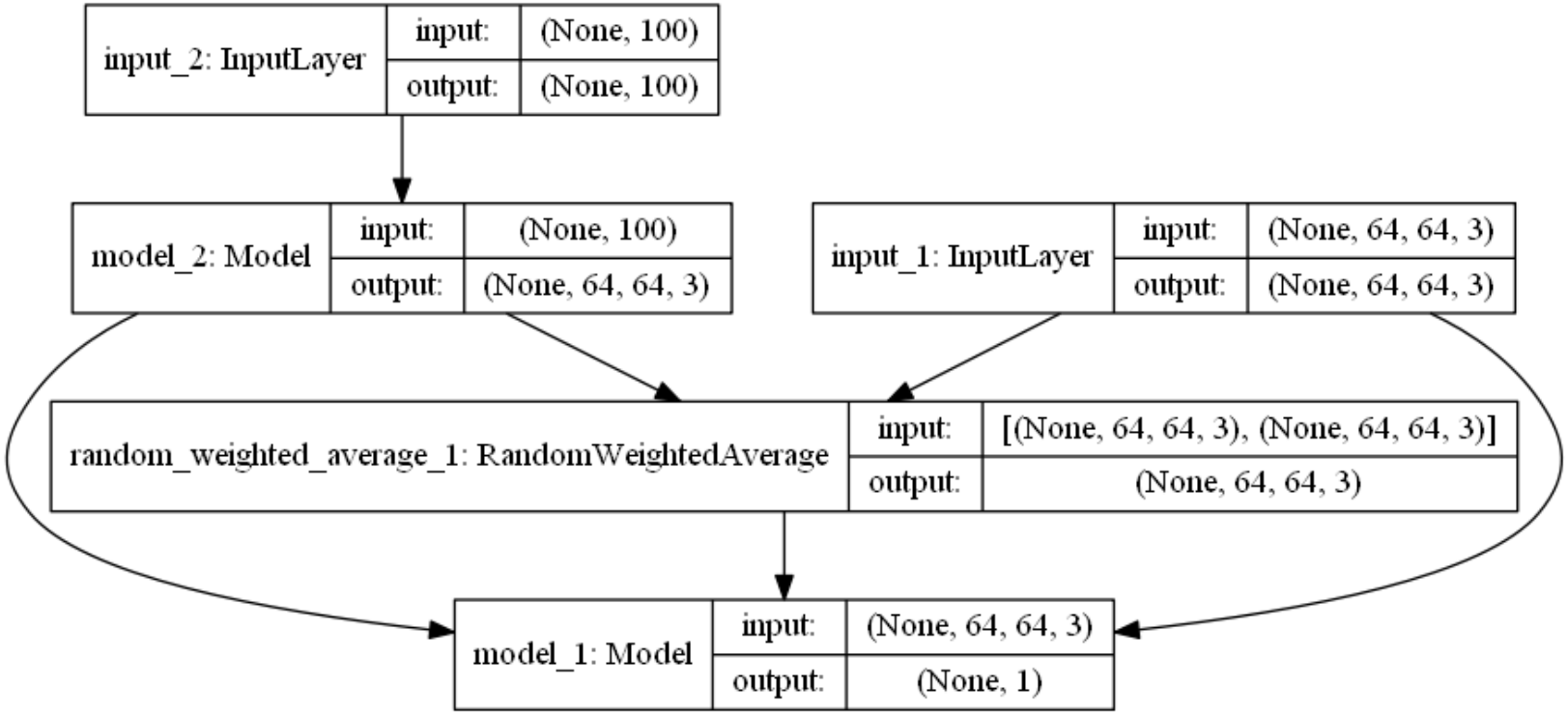
In [9]:
```
gan.generator.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
generator_input (InputLayer) (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 8192)              827392
_________________________________________________________________
batch_normalization_1 (Batch (None, 8192)              32768
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 8192)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 512)         0
_________________________________________________________________
generator_conv_0 (Conv2DTran (None, 8, 8, 256)         3277056
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 8, 8, 256)         0
_________________________________________________________________
generator_conv_1 (Conv2DTran (None, 16, 16, 128)       819328
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
generator_conv_2 (Conv2DTran (None, 32, 32, 64)        204864
_________________________________________________________________
batch_normalization_4 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
generator_conv_3 (Conv2DTran (None, 64, 64, 3)         4803
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 3)         0
=================================================================
Total params: 5,168,003
Trainable params: 5,150,723
Non-trainable params: 17,280
_________________________________________________________________
```

In [10]:
```
gan.critic_model.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 100)          0
__________________________________________________________________________________________________
input_1 (InputLayer)            (None, 64, 64, 3)    0
__________________________________________________________________________________________________
model_2 (Model)                 (None, 64, 64, 3)    5168003     input_2[0][0]
__________________________________________________________________________________________________
random_weighted_average_1 (Rand (None, 64, 64, 3)    0           input_1[0][0]
                                                                 model_2[1][0]
__________________________________________________________________________________________________
model_1 (Model)                 (None, 1)            4314753     model_2[1][0]
                                                                 input_1[0][0]
                                                                 random_weighted_average_1[0][0]
==================================================================================================
Total params: 4,332,033
Trainable params: 4,314,753
Non-trainable params: 17,280
__________________________________________________
```

```
/home/haesun/github/GDL_code/env/lib/python3.7/site-pa
ts and collected trainable weights, did you set `mode
  'Discrepancy between trainable weights and collecte
```



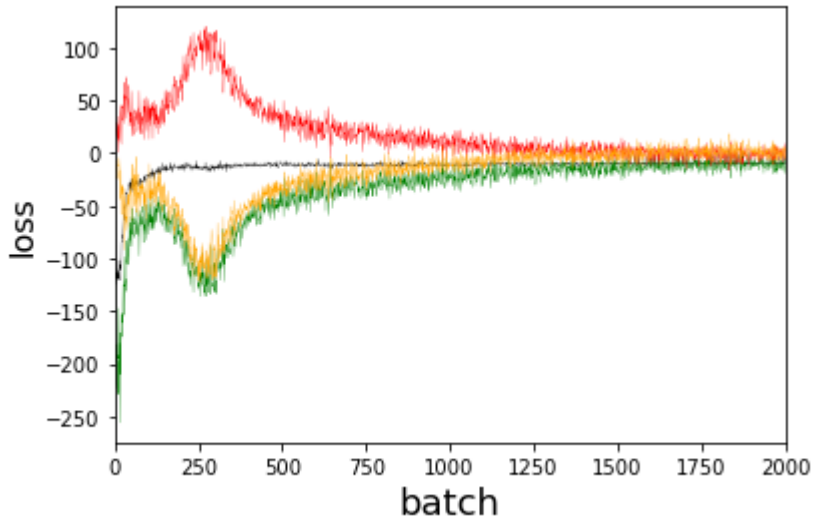
모델 훈련

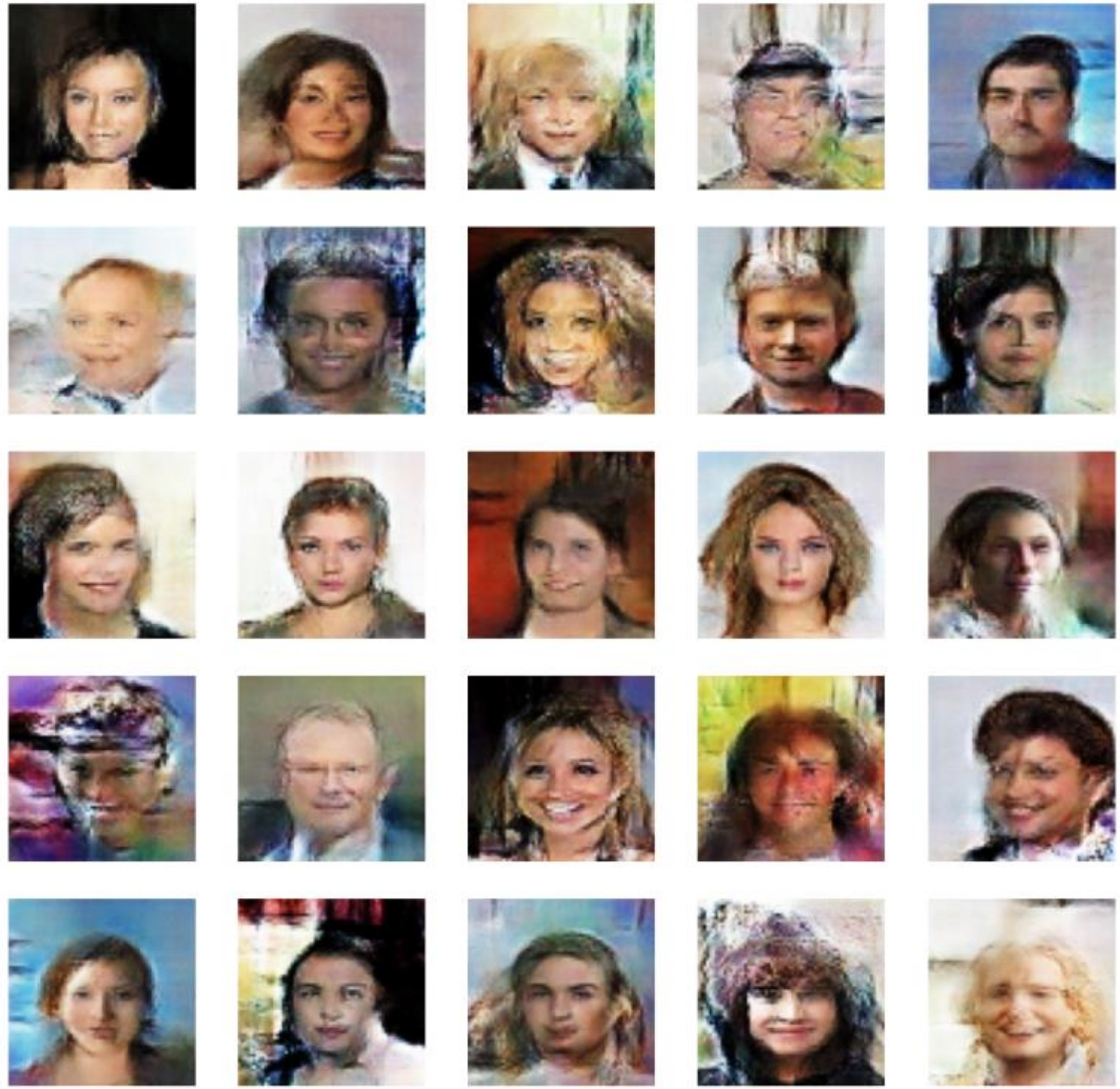
In [10]:
```
EPOCHS = 6000
PRINT_EVERY_N_BATCHES = 5
N_CRITIC = 5
BATCH_SIZE = 64
```

In [11]:
```
gan.train(
    x_train
    , batch_size = BATCH_SIZE
    , epochs = EPOCHS
    , run_folder = RUN_FOLDER
    , print_every_n_batches = PRINT_EVERY_N_BATCHES
    , n_critic = N_CRITIC
    , using_generator = True
)
```

```
WARNING:tensorflow:From /home/haesun/github/GDL_code/env/lib/python3.7/site-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (f
rom tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

/home/haesun/github/GDL_code/env/lib/python3.7/site-packages/keras/engine/training.py:490: UserWarning: Discrepancy between trainable we
ights and collected trainable weights, did you set `model.trainable` without calling `model.compile` after ?
  'Discrepancy between trainable weights and collected trainable'

0 (5, 1) [D loss: (1.0)(R -3.5, F -1.3, G 0.6)] [G loss: 2.5]
1 (5, 1) [D loss: (-66.7)(R -87.1, F -10.1, G 3.1)] [G loss: 6.4]
2 (5, 1) [D loss: (-122.9)(R -206.2, F 15.1, G 6.8)] [G loss: -14.2]
3 (5, 1) [D loss: (-120.7)(R -217.9, F 16.6, G 8.1)] [G loss: -11.7]
4 (5, 1) [D loss: (-135.9)(R -195.2, F 6.5, G 5.3)] [G loss: -18.7]
5 (5, 1) [D loss: (-130.0)(R -204.2, F 11.5, G 6.3)] [G loss: -16.2]
6 (5, 1) [D loss: (-141.2)(R -219.0, F 5.5, G 7.2)] [G loss: -29.8]
7 (5, 1) [D loss: (-140.4)(R -214.9, F 7.2, G 6.7)] [G loss: -20.4]
8 (5, 1) [D loss: (-136.3)(R -210.3, F 14.8, G 5.9)] [G loss: -24.1]
9 (5, 1) [D loss: (-130.8)(R -218.7, F 0.6, G 8.7)] [G loss: -5.2]
10 (5, 1) [D loss: (-114.4)(R -176.0, F 4.9, G 5.7)] [G loss: -8.8]
```

In [10]:
```
fig = plt.figure()
plt.plot([x[0] for x in gan.d_losses], color='black', linewidth=0.25)

plt.plot([x[1] for x in gan.d_losses], color='green', linewidth=0.25)
plt.plot([x[2] for x in gan.d_losses], color='red', linewidth=0.25)
plt.plot(gan.g_losses, color='orange', linewidth=0.25)

plt.xlabel('batch', fontsize=18)
plt.ylabel('loss', fontsize=16)

plt.xlim(0, 2000)
# plt.ylim(0, 2)

plt.show()
```





수고하셨습니다.