In [2]:

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from keras.layers import Input, Reshape, Dropout, Dense, Flatten
from keras.layers import BatchNormalization, Activation, ZeroPadding2D
from keras.layers.advanced_activations import LeakyReLU
from keras.layers.convolutional import UpSampling2D, Conv2D
from keras.models import Sequential, Model, load_model
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
```

```
Using TensorFlow backend.
/home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeon/lib/python3.7/site-pack
ages/pandas/compat/__init__.py:84: UserWarning: Could not import the lzma module.
Your installed Python is incomplete. Attempting to use lzma compression will resul
t in a RuntimeError.
  warnings.warn(msg)
/home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeon/lib/python3.7/site-pack
ages/pandas/compat/__init__.py:84: UserWarning: Could not import the lzma module.
Your installed Python is incomplete. Attempting to use lzma compression will resul
t in a RuntimeError.
  warnings.warn(msg)
```

In [3]:

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

from PIL import Image
from tqdm import tqdm

import os
```

In [7]:

```
class SongpyeonGenerator:
    def __init__(self,image_width,image_height,channels):
        self.image_width = image_width
        self.image_height = image_height

        self.channels = channels

        self.image_shape = (self.image_width,self.image_height,self.channels)

        #Amount of randomly generated numbers for the first layer of the generator.
        self.random_noise_dimension = 100

        #Just 10 times higher learning rate would result in generator loss being stuck at 0.
        optimizer = Adam(0.0002,0.5)

        self.discriminator = self.build_discriminator()
        self.discriminator.compile(loss="binary_crossentropy",optimizer=optimizer,metrics=["ac
curacy"])
        self.generator = self.build_generator()

        #A placeholder for the generator input.
        random_input = Input(shape=(self.random_noise_dimension,))

        #Generator generates images from random noise.
        generated_image = self.generator(random_input)

        # For the combined model we will only train the generator
        self.discriminator.trainable = False

        #Discriminator attempts to determine if image is real or generated
        validity = self.discriminator(generated_image)

        #Combined model = generator and discriminator combined.
        #1. Takes random noise as an input.
        #2. Generates an image.
        #3. Attempts to determine if image is real or generated.
        self.combined = Model(random_input,validity)
        self.combined.compile(loss="binary_crossentropy",optimizer=optimizer)

    def get_training_data(self,datafolder):
        print("Loading training data...")

        training_data = []
        #Finds all files in datafolder
        filenames = os.listdir(datafolder)
        for filename in tqdm(filenames):
            #Combines folder name and file name.
            path = os.path.join(datafolder,filename)
            #Opens an image as an Image object.
            image = Image.open(path).convert('RGB')
            #Resizes to a desired size.
            image = image.resize((self.image_width,self.image_height),Image.ANTIALIAS)
            #Creates an array of pixel values from the image.
            pixel_array = np.asarray(image)

            training_data.append(pixel_array)

        #training_data is converted to a numpy array
        training_data = np.reshape(training_data,(-1,self.image_width,self.image_height,self.c
```

```
hannels))
        return training_data



    def build_generator(self):
        #Generator attempts to fool discriminator by generating new images.
        model = Sequential()

        gf = 64
        model.add(Dense(16 * gf * self.image_width // 16 * self.image_height // 16,activation=
"relu",input_dim=self.random_noise_dimension))
        model.add(Reshape((self.image_width // 16, self.image_height // 16, gf * 16)))

        #Four layers of upsampling, convolution, batch normalization and activation.
        # 1. Upsampling: Input data is repeated. Default is (2,2). In that case a 4x4x256 arra
y becomes an 8x8x256 array.
        # 2. Convolution: If you are not familiar, you should watch this video: https://www.yo
utube.com/watch?v=FTr3n7uBIuE
        # 3. Normalization normalizes outputs from convolution.
        # 4. Relu activation:  f(x) = max(0,x). If x < 0, then f(x) = 0.


        model.add(UpSampling2D())
        model.add(Conv2D(gf * 8,kernel_size=3,padding="same"))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Activation("relu"))

        model.add(UpSampling2D())
        model.add(Conv2D(gf * 4,kernel_size=3,padding="same"))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Activation("relu"))

        model.add(UpSampling2D())
        model.add(Conv2D(gf,kernel_size=3,padding="same"))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Activation("relu"))

        model.add(UpSampling2D())
        model.add(Conv2D(gf,kernel_size=3,padding="same"))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Activation("relu"))


        # Last convolutional layer outputs as many featuremaps as channels in the final image.
        model.add(Conv2D(self.channels,kernel_size=3,padding="same"))
        # tanh maps everything to a range between -1 and 1.
        model.add(Activation("tanh"))

        # show the summary of the model architecture
        model.summary()

        # Placeholder for the random noise input
        input = Input(shape=(self.random_noise_dimension,))
        #Model output
        generated_image = model(input)

        #Change the model type from Sequential to Model (functional API) More at: https://kera
s.io/models/model/.
        return Model(input,generated_image)
```

```
    def build_discriminator(self):
        #Discriminator attempts to classify real and generated images
        model = Sequential()

        model.add(Conv2D(32, kernel_size=3, strides=2, input_shape=self.image_shape, padding=
"same"))
        #Leaky relu is similar to usual relu. If x < 0 then f(x) = x * alpha, otherwise f(x) =
x.
        model.add(LeakyReLU(alpha=0.2))

        #Dropout blocks some connections randomly. This help the model to generalize better.
        #0.25 means that every connection has a 25% chance of being blocked.
        model.add(Dropout(0.25))
        model.add(Conv2D(64, kernel_size=3, strides=2, padding="same"))
        #Zero padding adds additional rows and columns to the image. Those rows and columns ar
e made of zeros.
        model.add(ZeroPadding2D(padding=((0,1),(0,1))))
        model.add(BatchNormalization(momentum=0.8))
        model.add(LeakyReLU(alpha=0.2))

        model.add(Dropout(0.25))
        model.add(Conv2D(128, kernel_size=3, strides=2, padding="same"))
        model.add(BatchNormalization(momentum=0.8))
        model.add(LeakyReLU(alpha=0.2))

        model.add(Dropout(0.25))
        model.add(Conv2D(256, kernel_size=3, strides=1, padding="same"))
        model.add(BatchNormalization(momentum=0.8))
        model.add(LeakyReLU(alpha=0.2))

        model.add(Dropout(0.25))
        model.add(Conv2D(512, kernel_size=3, strides=1, padding="same"))
        model.add(BatchNormalization(momentum=0.8))
        model.add(LeakyReLU(alpha=0.2))

        model.add(Dropout(0.25))
        #Flatten layer flattens the output of the previous layer to a single dimension.
        model.add(Flatten())
        #Outputs a value between 0 and 1 that predicts whether image is real or generated. 0 =
generated, 1 = real.
        model.add(Dense(1, activation='sigmoid'))

        model.summary()

        input_image = Input(shape=self.image_shape)

        #Model output given an image.
        validity = model(input_image)

        return Model(input_image, validity)

    def train(self, datafolder ,epochs,batch_size,save_images_interval, checkpoint_path, isLoa
d = False):
        checkpoint_dir = os.path.dirname(checkpoint_path)
        generator_checkpoint_path = checkpoint_path+'/generator.ckpt'
        discriminator_checkpoint_path = checkpoint_path+'/discriminator.ckpt'
        #Get the real images
        training_data = self.get_training_data(datafolder)

        #Map all values to a range between -1 and 1.
```

```
        training_data = training_data / 127.5 - 1.

        #Two arrays of labels. Labels for real images: [1,1,1 ... 1,1,1], labels for generated
images: [0,0,0 ... 0,0,0]
        labels_for_real_images = np.ones((batch_size,1))
        labels_for_generated_images = np.zeros((batch_size,1))

        if isLoad:
            self.generator.load_weights(generator_checkpoint_path)
            self.discriminator.load_weights(discriminator_checkpoint_path)
        for epoch in range(epochs):
            # Select a random half of images
            indices = np.random.randint(0,training_data.shape[0],batch_size)
            real_images = training_data[indices]

            #Generate random noise for a whole batch.
            random_noise = np.random.normal(0,1,(batch_size,self.random_noise_dimension))
            #Generate a batch of new images.
            generated_images = self.generator.predict(random_noise)

            #Train the discriminator on real images.
            discriminator_loss_real = self.discriminator.train_on_batch(real_images,labels_for
_real_images)
            #Train the discriminator on generated images.
            discriminator_loss_generated = self.discriminator.train_on_batch(generated_images,
labels_for_generated_images)
            #Calculate the average discriminator loss.
            discriminator_loss = 0.5 * np.add(discriminator_loss_real,discriminator_loss_gener
ated)

            #Train the generator using the combined model. Generator tries to trick discrimina
tor into mistaking generated images as real.
            generator_loss = self.combined.train_on_batch(random_noise,labels_for_real_images)
            print ("%d [Discriminator loss: %f, acc.: %.2f%%] [Generator loss: %f]" % (epoch,
discriminator_loss[0], 100*discriminator_loss[1], generator_loss))

            if epoch % save_images_interval == 0:
                self.save_images(epoch)
                self.generator.save_weights(generator_checkpoint_path)
                self.discriminator.save_weights(discriminator_checkpoint_path)

        #Save the model for a later use
        self.generator.save("saved_models/songpyeongenerator.h5")


    def save_images(self,epoch):
        #Save 25 generated images for demonstration purposes using matplotlib.pyplot.
        rows, columns = 5, 5
        noise = np.random.normal(0, 1, (rows * columns, self.random_noise_dimension))
        generated_images = self.generator.predict(noise)

        generated_images = 0.5 * generated_images + 0.5

        figure, axis = plt.subplots(rows, columns)
        image_count = 0
        for row in range(rows):
            for column in range(columns):
                axis[row,column].imshow(generated_images[image_count, :], cmap='spring')
                axis[row,column].axis('off')
                image_count += 1
        figure.savefig("generated_images/generated_%d.png" % epoch)
```

```
        plt.close()

    def generate_single_image(self,model_path,image_save_path):
        noise = np.random.normal(0,1,(1,self.random_noise_dimension))
        model = load_model(model_path)
        generated_image = model.predict(noise)
        #Normalized (-1 to 1) pixel values to the real (0 to 256) pixel values.
        generated_image = (generated_image+1)*127.5
        print(generated_image)
        #Drop the batch dimension. From (1,w,h,c) to (w,h,c)
        generated_image = np.reshape(generated_image,self.image_shape)

        image = Image.fromarray(generated_image,"RGB")
        image.save(image_save_path)
```

In [ ]:

```
songpyeongenerator = SongpyeonGenerator(256, 256, 3)
songpyeongenerator.train(datafolder="img", epochs = 2000, batch_size=32, save_images_interval=
100, checkpoint_path="checkpoint", isLoad= False)
songpyeongenerator.generate_single_image("saved_models/songpyeongenerator.h5", "test.png")
```

```
WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:66: The name tf.
get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instea
d.

WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:541: The name t
f.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:4432: The name t
f.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:148: The name t
f.placeholder_with_default is deprecated. Please use tf.compat.v1.placeholder_with
_default instead.

WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:3733: calling dr
opout (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_p
rob`.
WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:190: The name t
f.get_default_session is deprecated. Please use tf.compat.v1.get_default_session i
nstead.

WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:2041: The name t
f.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fused_batch_norm i
nstead.

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 32)      896
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 128, 128, 32)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 65, 65, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 65, 65, 64)        256
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 65, 65, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 65, 65, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 33, 33, 128)       73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 33, 33, 128)       512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 33, 33, 128)       0
_________________________________________________________________
```

```
dropout_3 (Dropout)          (None, 33, 33, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 33, 33, 256)       295168
_________________________________________________________________
batch_normalization_3 (Batch (None, 33, 33, 256)       1024
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 33, 33, 256)       0
_________________________________________________________________
dropout_4 (Dropout)          (None, 33, 33, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 33, 33, 512)       1180160
_________________________________________________________________
batch_normalization_4 (Batch (None, 33, 33, 512)       2048
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 33, 33, 512)       0
_________________________________________________________________
dropout_5 (Dropout)          (None, 33, 33, 512)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 557568)            0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 557569
=================================================================
Total params: 2,129,985
Trainable params: 2,128,065
Non-trainable params: 1,920
_________________________________________________________________
WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/optimizers.py:793: The name tf.train.Optimizer
is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/tensorflow/python/ops/nn_impl.py:180: add_dispatch_s
upport.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and w
ill be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeo
n/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:2239: The name t
f.image.resize_nearest_neighbor is deprecated. Please use tf.compat.v1.image.resiz
e_nearest_neighbor instead.

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 262144)            26476544
_________________________________________________________________
reshape_1 (Reshape)          (None, 16, 16, 1024)      0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 32, 32, 1024)      0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 512)       4719104
_________________________________________________________________
batch_normalization_5 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 512)       0
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 64, 64, 512)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 64, 64, 256)       1179904
_________________________________________________________________
```

```
batch_normalization_6 (Batch (None, 64, 64, 256)       1024
_________________________________________________________________
activation_2 (Activation)    (None, 64, 64, 256)       0
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 128, 128, 256)     0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 128, 128, 64)      147520
_________________________________________________________________
batch_normalization_7 (Batch (None, 128, 128, 64)      256
_________________________________________________________________
activation_3 (Activation)    (None, 128, 128, 64)      0
_________________________________________________________________
up_sampling2d_4 (UpSampling2 (None, 256, 256, 64)      0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 256, 256, 64)      36928
_________________________________________________________________
batch_normalization_8 (Batch (None, 256, 256, 64)      256
_________________________________________________________________
activation_4 (Activation)    (None, 256, 256, 64)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 256, 256, 3)       1731
_________________________________________________________________
activation_5 (Activation)    (None, 256, 256, 3)       0
=================================================================
Total params: 32,565,315
Trainable params: 32,563,523
Non-trainable params: 1,792
_________________________________________________________________

  0%|          | 0/364 [00:00<?, ?it/s]

Loading training data...

100%|██████████| 364/364 [00:07<00:00, 47.80it/s]
/home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeon/lib/python3.7/site-pack
ages/keras/engine/training.py:493: UserWarning: Discrepancy between trainable weig
hts and collected trainable weights, did you set `model.trainable` without calling
`model.compile` after ?
  'Discrepancy between trainable weights and collected trainable'
/home/lmu/.anyenv/envs/pyenv/versions/3.7.4/envs/songpyeon/lib/python3.7/site-pack
ages/keras/engine/training.py:493: UserWarning: Discrepancy between trainable weig
hts and collected trainable weights, did you set `model.trainable` without calling
`model.compile` after ?
  'Discrepancy between trainable weights and collected trainable'
```

In [ ]:

In [ ]: