Tutorial

Healthc Inform Res. 2019 October;25(4):344-349.
https://doi.org/10.4258/hir.2019.25.4.344
pISSN 2093-3681 • eISSN 2093-369X

HIR
Healthcare Informatics Research

Augmentation of Doppler Radar Data Using Generative Adversarial Network for Human Motion Analysis

Ibrahim Alnujaim, Youngwook Kim
Department of Electrical and Computer Engineering, California State University, Fresno, CA, USA

Objectives: Human motion analysis can be applied to the diagnosis of musculoskeletal diseases, rehabilitation therapies, fall detection, and estimation of energy expenditure. To analyze human motion with micro-Doppler signatures measured by radar, a deep learning algorithm is one of the most effective approaches. Because deep learning requires a large data set, the high cost involved in measuring large amounts of human data is an intrinsic problem. The objective of this study is to augment human motion micro-Doppler data employing generative adversarial networks (GANs) to improve the accuracy of human motion classification. **Methods:** To test data augmentation provided by GANs, authentic data for 7 human activities were collected using micro-Doppler radar. Each motion yielded 144 data samples. Software including GPU driver, CUDA library, cuDNN library, and Anaconda were installed to train the GANs. Keras-GPU, SciPy, Pillow, OpenCV, Matplotlib, and Git were used to create an Anaconda environment. The data produced by GANs were saved every 300 epochs, and the training was stopped at 3,000 epochs. The images generated from each epoch were evaluated, and the best images were selected. **Results:** Each data set of the micro-Doppler signatures, consisting of 144 data samples, was augmented to produce 1,472 synthesized spectrograms of 64 × 64. Using the augmented spectrograms, the deep neural network was trained, increasing the accuracy of human motion classification. **Conclusions:** Data augmentation to increase the amount of training data was successfully conducted through the use of GANs. Thus, augmented micro-Doppler data can contribute to improving the accuracy of human motion recognition.

Keywords: Motion Perception, Data Visualization, Deep Learning, Big Data, Supervised Machine Learning

Submitted: July 16, 2019
Revised: October 7, 2019
Accepted: October 21, 2019

Corresponding Author
Youngwook Kim
Department of Electrical and Computer Engineering, California State University, 2320 E. San Ramon Ave, MS EE 94 Fresno, CA 93740-8030, USA. Tel: +1-559-278-4629, E-mail: youngkim@csufresno.edu (https://orcid.org/0000-0002-4067-6254)

I. Introduction

Human motion analysis has diverse applications in medicine, healthcare, rehabilitation, game engineering, surveillance, search and rescue, and defense. Human motion analysis can be used for the diagnosis of motion-related diseases, such as cumulative trauma disorders, psychosomatic disorders, and autism spectrum disorders [1]. Energy expenditure can be estimated by the class of human motion [2]. In addition, human gait analysis is essential for the evaluation of the degree of rehabilitation. Radar offers a unique opportunity for monitoring human motion remotely. In particular, micro-Doppler signatures produced by human limb motion contain information pertaining to such motion. Because micro-Doppler signatures are represented as a spectrogram in the form of an image, human motions can be recognized through analysis of spectrogram images.

Due to the advancement of deep learning, the image recognition/classification problem can be effectively addressed by the use of deep convolutional neural networks (DCNN). To train a DCNN effectively to achieve high classification accuracy requires a large amount of image data. In the case of radar data, such an effort is challenging due to a lack of historical records as well as the high costs of collecting a large data set. Therefore, it is necessary to augment the radar data set to fully explore the capability of a DCNN. Recently, generative adversarial networks (GANs) have been successfully used to address the radar data augmentation problem [3].

A GAN is a machine learning algorithm designed to produce large amounts of synthesized data that have similar distributions to that of the original data. Owing to this capability, GANs have many applications, such as image synthesis, image de-noising, and image-to-image translation. A GAN consists of two networks, a generative network and a discriminative network, that compete against each other during the training process. The generative network generates synthesized images, and the discriminative network evaluates the generated images. During training, the cost function is defined such that the generative network decreases the classification rate of the discriminative network, while the discriminative network is trained to increase the classification accuracy. Over the course of training, each of the networks contributes to improve the appearance of generated images [4].

This tutorial will describe the process of setting up environments for GANs through the installation of the GPU driver, cuDNN library, CUDA library, and Anaconda along with the training of GANs using a measured data set. Finally, we will apply this approach to augment a human motion data set measured by Doppler radar to investigate whether the augmented data are effective in the training of a DCNN.

II. Methods

In this study, the data set included 7 activities that were recorded using 12 human subjects for 12 iterations; the total number of data points was 1,008. Figure 1 shows the measurement setup. The 7 activities included boxing while moving forward, boxing while standing in place, crawling, running, sitting still, walking, and walking low while holding a stick. The data were organized in a MATLAB .mat file named *Seven_activity*. The .mat file has a structure file named *activity*. The structure has three fields. The first field, *name*, is a string containing the activity name; the second field, *human_number*, is a numerical number with a datatype double; and the third field, *data*, has a matrix sized 600 × 140.

The GAN we designed consists of two neural networks. The generative network takes an input of a noise vector and tries to produce a synthesized image, while the discriminative network tries to classify the data correctly as synthesized or real data. To train networks, an Adam optimizer is employed to reduce the loss function. The loss function is defined using the error from the discriminative network. For the activation, a sigmoid function is used. To initialize the weight and biases in the neural networks, Xavier initialization is used. The steps and processes are described in below.

This tutorial was completed on Windows 7 with an i7 CPU and an NVIDIA GTX 770 GPU. Since Anaconda and GPU drivers are available on MacOS and Ubuntu, this should be easy to replicate on any OS and any NVIDIA graphic card with cuDNN support and the same software as that used in our work. Training a GAN requires significant memory space to complete the process; a GPU is preferred because of the large amount of memory available. To use the GPU, the right drivers must be installed correctly, starting with the graphic card driver and followed by the CUDA library and the cuDNN library. To download the CUDA library, go to website (https://developer.nvidia.com/cuda-toolkit), click on 'Download now', then choose the appropriate operating system and follow the installation instructions [5]. To download the cuDNN library, go to website (https://developer.nvidia.com/rdp/form/cudnn-download-survey). A membership must be created to download the file. It is necessary to download the latest library with the appropriate CUDA version and then follow the installation process [6]. In a Windows environment, the following path variable must be added:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin;

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\

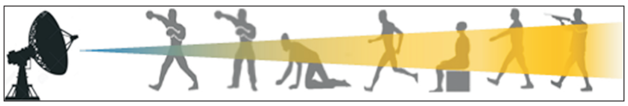


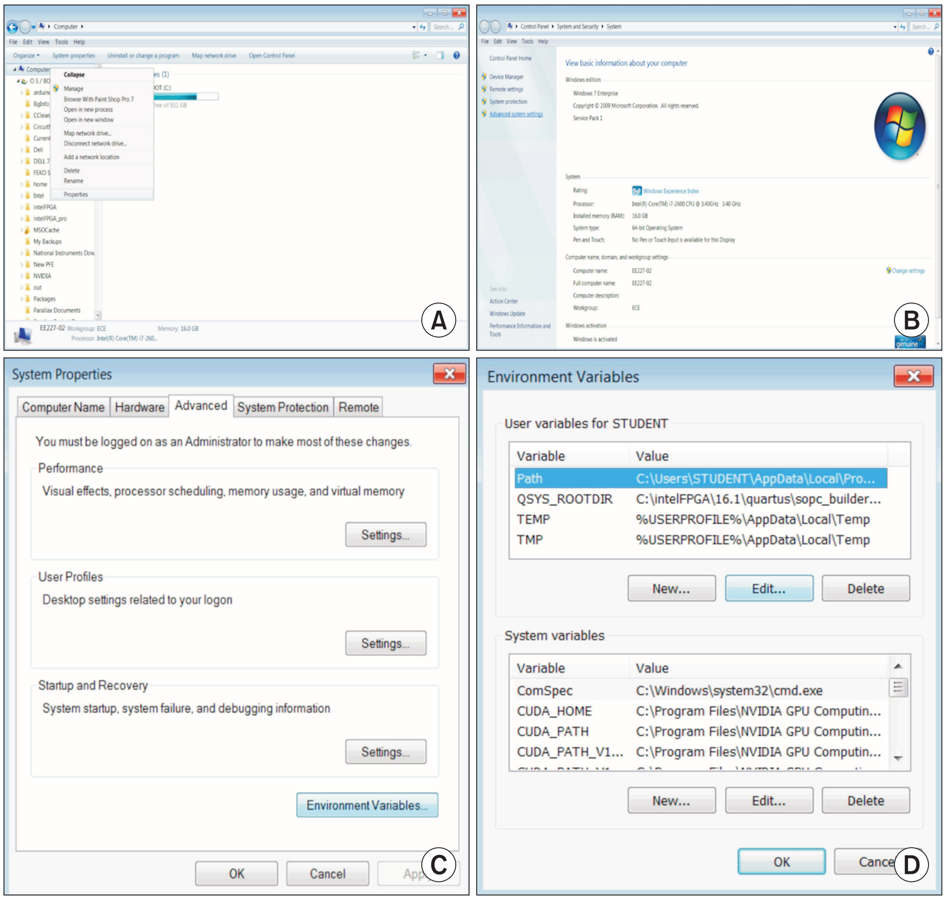
Figure 1. Setup for the seven human motion measurement using Doppler radar.

v10.1\libnvvp.

The procedure to set the path variable is shown in Figure 2. To use TensorFlow, Anaconda version 4.3 with Python 2 must be installed. This Anaconda version can be found at https://repo.continuum.io/archive/. Download 'Anaconda2-4.3.1-Windows-x86.exe' for a 32-bit system or 'Anaconda2-4.3.1-Windows-x86_64.exe' for a 64-bit system. Then follow the installation process. After installation of Anaconda, the following path variable must be added:

C:\Program Files\Anaconda2;
C:\Program Files\Anaconda2\Scripts;
C:\Program Files\Anaconda2\Library\bin.

After installing Anaconda and setting the path variable, open the Anaconda Prompt and create an environment using the following command:



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your logon

Startup and Recovery
System startup, system failure, and debugging information

Environment Variables

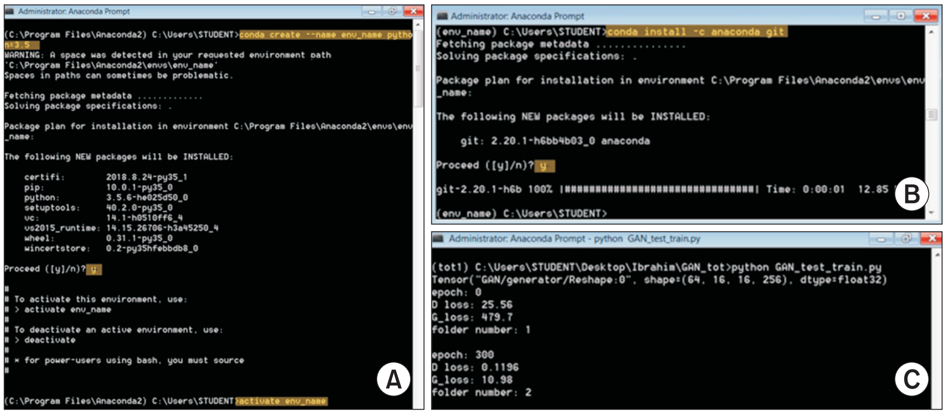
User variables for STUDENT

Variable	Value
Path	C:\Users\STUDENT\AppData\Local\Pro...
QSYS_ROOTDIR	C:\intelFPGA\16.1\quartus\sopc_builder...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
CUDA_HOME	C:\Program Files\NVIDIA GPU Computin...
CUDA_PATH	C:\Program Files\NVIDIA GPU Computin...
CUDA_PATH_V1...	C:\Program Files\NVIDIA GPU Computin...

Figure 2. Procedure for setting up a path: (A) open Windows Explorer then choose Properties, (B) open advanced system setting, (C) setting up environment variables, and (D) select path then click edit.



```
Administrator: Anaconda Prompt

(C:\Program Files\Anaconda2) C:\Users\STUDENT>conda create --name env_name pytho
n=3.5
WARNING: A space was detected in your requested environment path
'C:\Program Files\Anaconda2\envs\env_name'
Spaces in paths can sometimes be problematic.

Fetching package metadata .............
Solving package specifications: .

Package plan for installation in environment C:\Program Files\Anaconda2\envs\env
_name:

The following NEW packages will be INSTALLED:

    certifi:        2018.8.24-py35_1
    pip:            10.0.1-py35_0
    python:         3.5.6-he025d50_0
    setuptools:     40.2.0-py35_0
    vc:             14.1-h0510ff6_4
    vs2015_runtime: 14.15.26706-h3a45250_4
    wheel:          0.31.1-py35_0
    wincertstore:   0.2-py35hfebbdb8_0

Proceed ([y]/n)? y

#
# To activate this environment, use:
# > activate env_name
#
# To deactivate an active environment, use:
# > deactivate
#
# * for power-users using bash, you must source
#


(C:\Program Files\Anaconda2) C:\Users\STUDENT>activate env_name
```

```
Administrator: Anaconda Prompt

(env_name) C:\Users\STUDENT>conda install -c anaconda git
Fetching package metadata ...............
Solving package specifications: .

Package plan for installation in environment C:\Program Files\Anaconda2\envs\env
_name:

The following NEW packages will be INSTALLED:

    git: 2.20.1-h6bb4b03_0 anaconda

Proceed ([y]/n)? y

git-2.20.1-h6b 100% |###################################| Time: 0:00:01  12.85

(env_name) C:\Users\STUDENT>
```

```
Administrator: Anaconda Prompt - python  GAN_test_train.py

(tot1) C:\Users\STUDENT\Desktop\Ibrahim\GAN_tot>python GAN_test_train.py
Tensor("GAN/generator/Reshape:0", shape=(64, 16, 16, 256), dtype=float32)
epoch: 0
D loss: 25.56
G_loss: 479.7
folder number: 1

epoch: 300
D loss: 0.1196
G_loss: 10.98
folder number: 2
```

Figure 3. Anaconda environment and training process: (A) creating and activating an Anaconda environment, (B) installing a package in Anaconda, and (C) running a training session.

```
conda create --name env_name python = 3.5.
```

The name of the environment in this case is env_name. The Python version must be set to 3.5 because this is what TensorFlow uses. The environment can be activated using the following command line:

```
activate env_name.
```

Figure 3A shows a screenshot of an example for creating and activating an environment. The following packages must then be installed—Keras-GPU, SciPy, Pillow, OpenCV, Matplotlib, and Git—in the env_name environment [7-12]. These packages can easily be installed by typing the following command lines:

```
conda install -c conda-forge keras-gpu
conda install -c anaconda scipy
conda install -c anaconda pillow
conda install -c conda-forge opencv
conda install -c conda-forge matplotlib
conda install -c anaconda git
```

Each line must be executed in sequence. See Figure 3B for an example of a package installation using Anaconda Prompt.

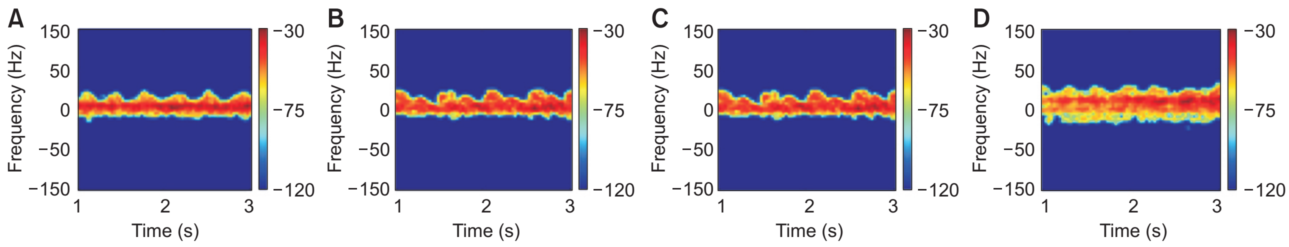
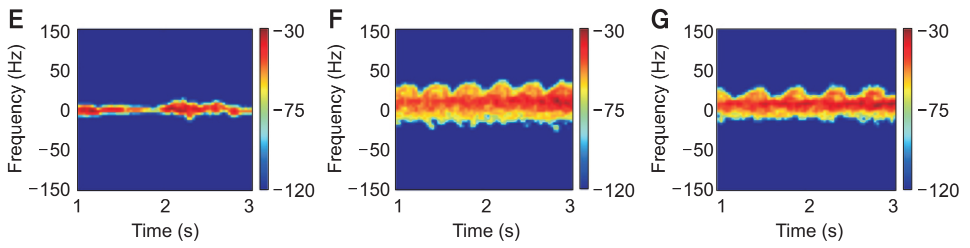



Figure 4. Original micro-Doppler image of the following seven activities: (A) boxing while moving forward, (B) boxing while standing in place, (C) crawling, (D) running, (E) sitting still, (F) walking, and (G) walking hunched over while holding a stick.

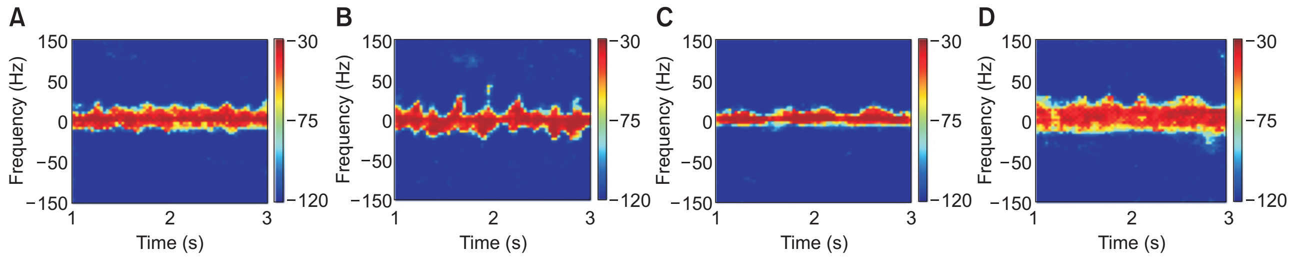
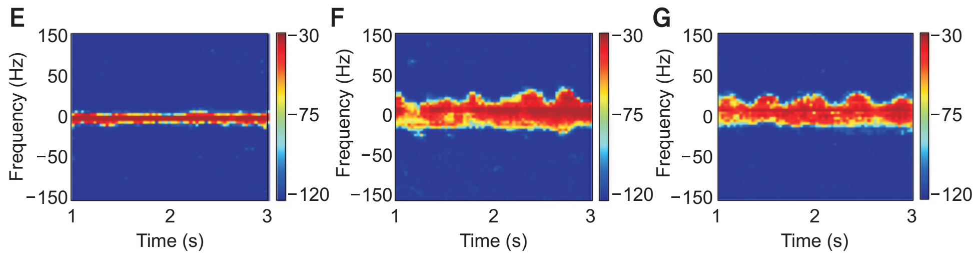



Figure 5. Augmented micro-Doppler image using generative adversarial networks: (A) boxing while moving forward, (B) boxing while standing in place, (C) crawling, (D) running, (E) sitting still, (F) walking, and (G) walking hunched over while holding a stick.

Once everything is installed, the code must be downloaded to start the training process. The recommendation is to create a folder into which the code can be saved. Next, the folder should be set as the current directory by using the command cd. The code can be downloaded by using the following command [13]:

```
git clone https://github.com/isn350/e_hir_GAN.git
```

Open *read.py* to edit the path to the data set in line:

```
data_file = 'C:/Users/STUDENT/Desktop/Ibrahim/GAN_tot'
```

The size of the data used in this study was 600 × 140, but if the size of data used is different, this can be modified in read.py as in the following lines:

```
original_image_dim_x = 600
original_image_dim_y = 140
```

The number of data points in the data set must be set in *read.py* in the following line:

```
if x % 144 == 0:
```

The function *read.py* has three objectives, namely, reading the data, preparing the data for GANs training, and visualization of the data. The function reads a .mat file and resizes the images to 64 × 64 to input to the GANs, while the size of the input image can be any in the .mat file. For example, the original data size in our case was 600 × 140. Once the codes are saved, to run the code, run the following command:

```
python GAN_train.py
```

Running the above command only produces a GAN image for boxing while moving forward. To change the activity, open *GAN_train.py* using Notepad, find the line below to change the activity, *activity* = '*boxingmoving*', and change the name between single quotation marks. The code in *GAN_train.py* initiates the training process of GANs. In the code, the directory of the input data and output data from GANs is determined. Figure 3C shows an example of running the code and the output lines.

III. Results

After visual inspection, augmented images were produced at 2,700 epochs. Images of the original data are shown in Figure 4, and Figure 5 presents the augmented images. As seen in Figures 4 and 5, the synthesized images from the GANs show a similar distribution to that of the original images. With the combination of original data and synthesized data, the DCNN is designed and trained. The number of layers of the DCNN structure is selected heuristically until the classification accuracy becomes saturated. The DCNN we designed has 6 layers including 3 convolutional layers and 3 fully connected layers. The numbers of filters in the convolutional layers are 16, 32, and 64, while the numbers of nodes in the fully connected layers are 124, 124 and 7. In the convolutional layer, batch normalization, a rectified linear unit, and max pooling are employed. The convolutional filter size is 2 × 2. We have only considered the motion classification accuracy of original data because the classification of synthesized data is meaningless even though they are used in the training process. The results reveal that the use of GANs can improve the recognition of human motion from 90% to 94% when the same DCNN structure is used.

IV. Discussion

This paper presented the overall process of preparing an environment for GANs and training them. In particular, we have presented an example of augmenting micro-Doppler radar data of human motion measured by Doppler radar. Owing to the augmented data set, deeper neural networks can be constructed and effectively trained, resulting in better classification accuracy. This preliminary research on the automatic recognition of human motion has the potential to contribute to diverse applications in healthcare and rehabilitation, such as human gait analysis or energy expenditure estimation.

It should be noted that the current use of GANs presents challenges as it is an emerging and advancing technology. First, no standard currently exists to evaluate the quality of GANs outputs. The number of epochs should be determined by visual inspection, which can be subjective. Therefore, it is not easy to quantify the success of GANs training. Second, GANs occasionally have a mode-collapsing issue that limits the production of outputs with diverse characteristics. In addition, improperly trained GANs produce only very similar images. These issues should be addressed in the future to enable the wider use of GANs in radar image processing.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

ORCID

Ibrahim Alnujaim (http://orcid.org/0000-0001-5610-0631)
Youngwook Kim (http://orcid.org/0000-0002-4067-6254)

References

1. Taylor JL, Gotham KO. Cumulative life events, traumatic experiences, and psychiatric symptomatology in transition-aged youth with autism spectrum disorder. J Neurodev Disord 2016;8:28.
2. Dongwoo K, Kim HC. Activity energy expenditure assessment system based on activity classification using multi-site triaxial accelerometers. Conf Proc IEEE Eng Med Biol Soc 2007;2007:2285-7.
3. Alnujaim I, Oh D, Kim Y. Generative adversarial networks for classification of micro-doppler signatures of human activity. IEEE Geosci Remote Sens Lett 2019 [Epub]. http://10.1109/LGRS.2019.2919770.
4. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Adv Neural Inf Process Syst 2014;27;2672-80.
5. NVIDIA CUDA Toolkit [Internet]. Santa Clara (CA): NVIDIA Corp.; 2019 [cited at 2019 June 18]. Available from: https://developer.nvidia.com/cuda-toolkit.
6. NVIDIA cuDNN [Internet]. Santa Clara (CA): NVIDIA Corp.; 2019 [cited at 2019 June 18]. Available from: https://developer.nvidia.com/cudnn.
7. Anaconda Keras-GPU [Internet]. Austin (TX): Anaconda Inc.; 2019 [cited at 2019 June 18]. Available from: https://anaconda.org/anaconda/keras-gpu.
8. Anaconda SciPy [Internet]. Austin (TX): Anaconda Inc.; 2019 [cited at 2019 June 18]. Available from: https://anaconda.org/anaconda/scipy.
9. Anaconda Pillow [Internet]. Austin (TX): Anaconda Inc.; 2019 [cited at 2019 June 18]. Available from: https://anaconda.org/conda-forge/pillow.
10. Anaconda OpenCv [Internet]. Austin (TX): Anaconda Inc.; 2019 [cited at 2019 June 18]. Available from: https://anaconda.org/conda-forge/opencv.
11. Anaconda matplotlib [Internet]. Austin (TX): Anaconda Inc.; 2019 [cited at 2019 June 18]. Available from: https://anaconda.org/conda-forge/matplotlib.
12. Anaconda git [Internet]. Austin (TX): Anaconda Inc.; 2019 [cited at 2019 June 18]. Available from: https://anaconda.org/anaconda/git.
13. e-hir GAN Tutorial [Internet]. [place unknown]: github.com; c2019 [cited at 2019 June 18]. Available from: https://github.com/isn350/e_hir_GAN.