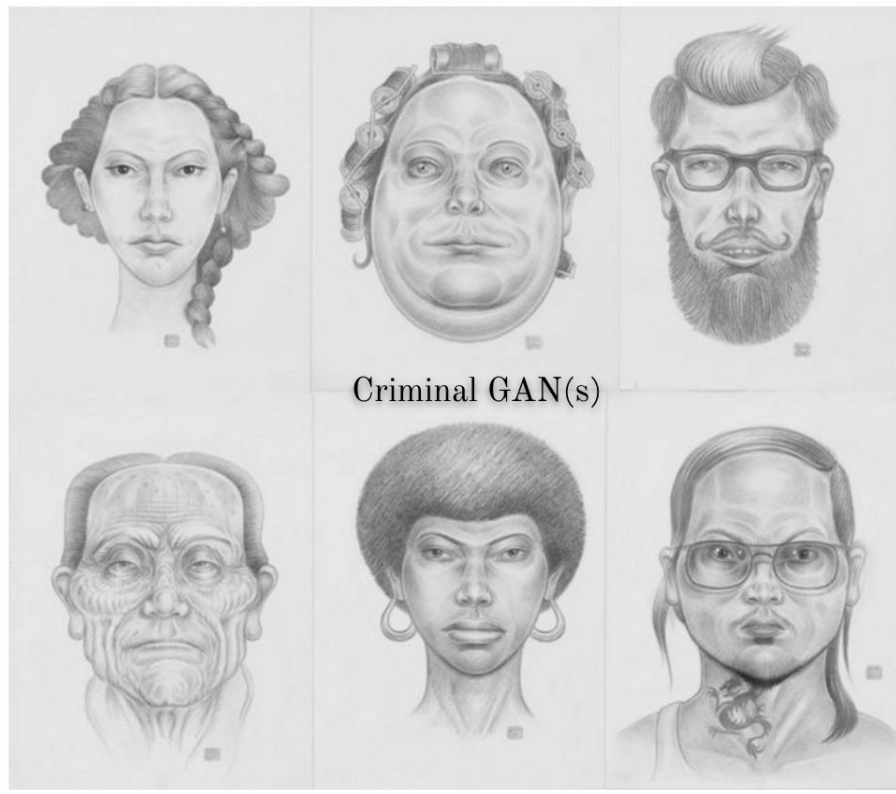


Criminal GAN(s)



Kasidech Tantipanichaphan

Description

The concept/goal of this project is to gather what we've explored in the context of this class and to help formulate a more efficient way in producing a facial composite of the criminal. There are two methods that I decided to use. The first method is using the CycleGAN to produce a better face composition of the criminal from the police sketches. Maybe the criminal face could not be thoroughly visualized. The CycleGAN will select a new loss function to generate more sharp and realistic images. Another element is to produce text to face the result. The idea is to use StackGAN and ProGAN to synthesize faces of the criminal based on the textual description from the victim. The reason why is because sometimes it takes time to sketch the faces of the criminal based on the victim's description. By using machines to generate the face could be an efficient way of doing so.

Concept

Nowadays, police sketches are nevertheless a routine part of law enforcement investigations. In other words, visualization plays an important role towards the criminal investigation. Because the picture of the criminal is not available, this requires us to draw a brief sketch. Clearly, this is not enough for the officers to speculate. Notice that sometimes we cannot clearly visualize the image of the criminal by the sketch that the sketch-artist has drawn. According to the *Physical Security* statistics, some researchers suggest that facial sketches or composite pictures of suspected criminals are useful less than 20% of the time. Other studies put the percentage even lower – maybe as low as 8%. That means that at least 80% of the time, the time and effort spent creating and circulating these pictures are worthless.

If this concept were to actually implement in real life, say if we have access to the data have all the people faces in the world, which can be obtained from the government embassy (could be the images from their passport or photo I.D.). Then, the use of these methods could be effective towards the future investigation.

Technique

There are two parts for my project, sketch-to-face and text-to-face. There are three techniques that I decided to use: StackGAN, ProGAN, and CycleGAN. In the first part, I synthesize the realistic face of the criminal based on the face sketch. The job of CycleGAN is to select a new loss function to generate more sharp and realistic images. In the second part I use StackGAN to synthesize the face of the criminal based on the given textual description. The textual description is encoded into a summary vector using an LSTM network. The concept of StackGAN is to sketch the primitive shapes as well as color of the object, which in this case is the criminal face.

ProGAN, was used to both speed the training up as well as stabilize it. To explain the concept of ProGAN, it has two networks, which are generator and discriminator. The generator

creates “fakes”, and the discriminator attempts to distinguish these “fake” samples from the “real” ones. Both networks start out performing quite poorly at their tasks, but when training goes well, they improve in tandem until the generator is producing convincing fakes. This was used to both kind of speed the training up as well as stabilize it.

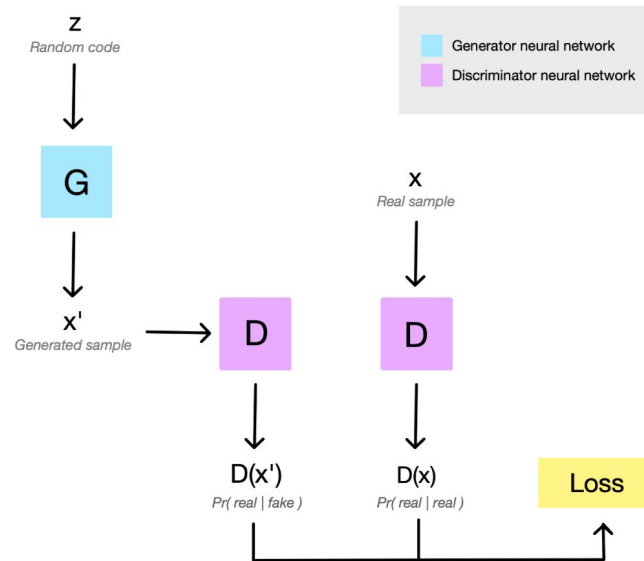


Figure 1

There are three main datasets that I used for the first part of the project: FERET dataset, CUHK dataset, and MUCT Face database. According to the website, FERET database is a dataset used for facial recognition system evaluation as part of the Face Recognition Technology (FERET) program. The database contains a total of ~14,125 images pertaining to 1199 individuals along with 365 duplicate sets of images that were taken on a different day. The FERET dataset is needed to be requested from the National Institute of Standards and Technology (.gov) website. Similarly, the MUCT Face database also contains lots of people's faces, with the exception that it contains color images instead of gray-scale. Another dataset that I use is called CUHK Face Sketch dataset. Basically, this dataset is for research on face sketch synthesis. This includes 1194 persons, with different lighting variation and a sketch with shape exaggeration drawn by an artist. The second part of the project I use Face2Text dataset which contains 400 facial images and textual captions for each of them.

Process

In this implementation, I use the CycleGAN model to generate the results. After we got the FERET and CUHK dataset. It is time to use cycleGAN to train these images. The trainA is the sketches images from CUHK dataset and the trainB is the actual face images from FERET dataset. The color FERET dataset was requested to the National Institute of Standards and Technology (.gov), but the organization does not reply back. Therefore, I used the MUCT Database instead.

Added more images, so that the data becomes more and more accurate (roughly about 1000+). During the experiment, I adjusted several parameters as well as the number of epochs. The epochs that are greater 100 achieve more satisfying results. The cycle took about 2-3 hours to achieve the best result.

The second part of the project, I use StackGAN and ProGAN to synthesize faces from textual descriptions. This process took 3-4 hours to train the model. This process requires PyTorch framework. I installed PyTorch version 0.4.0 by executed the following command

```
!pip install http://download.pytorch.org/whl/cu80/torch-0.4.1-cp36-cp36m-linux_x86_64.whl
```

Also, execute `!pip install torchvision`. This is the prerequisite so that I will be able to generate my results, which sketches the primitive shape and colors of the object based on the given textual description. The textual description is encoded into a summary vector using an LSTM network. The dataset that I used is called *Face2Text* dataset, however note that this project is performing a reverse. Just to give a brief description *Face2Text* dataset which contains 400 facial images and textual captions for each of them.

The following is the coding organization:

`configs`: contains the configuration files for training the network.

`data_processing`: package containing data processing and loading modules

`networks`: package contains network implementation

`train_network.py`: script for running the training the network

`processed_annotations`: directory stores output of running `process_text_annotations.py` script

`process_text_annotations.py`: processes the captions and stores output in `processed_annotations/` directory.

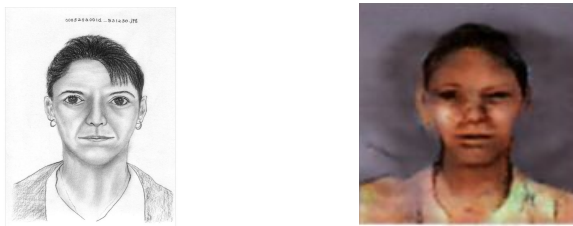
Sketch-to-Face Results (Method 1)

Synthesize the face of the criminal based on the police face sketches

Train/Generate Using Faret Dataset



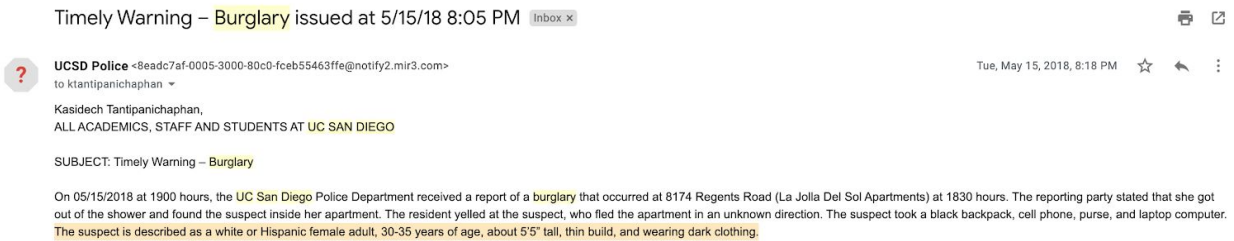
Train/Generate Using MUCT dataset



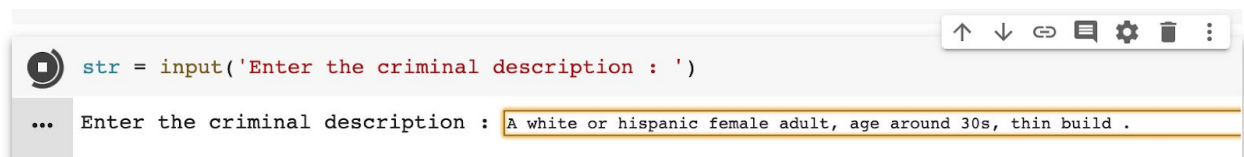
Text-to-face Results (Method 2)

Synthesize the face of the criminal based on the textual description given by the victims.

Example from UCSD Timely Warning Report Email



The highlighted sentence is texts that I want to generate and synthesize the face of the criminal. Now, I remove the unrelated information, such as the person's height and what clothes they are wearing, because they are not the facial component. The following is the input string that I used:




After that, below is the generated result.






More Example from UCSD Timely Warning Report Email




[APB-L] Timely Warning - Burglary



UCSD Police Notice <apb-l@announce.ucsd.edu>
to apb-l@ucsd.edu

Sun, Jan 14, 2018, 7:26 AM





UC SAN DIEGO POLICE DEPARTMENT

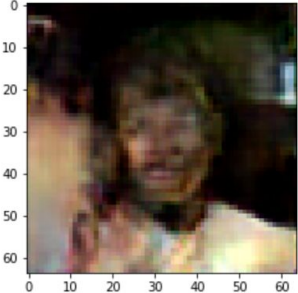
January 14, 2018

ALL ACADEMICS, STAFF AND STUDENTS AT UC SAN DIEGO

SUBJECT: Timely Warning – Burglary

On 01/13/2018 at 8:34 pm, the UC San Diego Police Department received a report of a burglary that had just occurred at Bonner Hall. The reporting party stated that a male had just fled the building with his laptop computer and ran west, toward Urey Hall. The victim described the suspect as a black male adult with short curly hair, approximately 5'6" tall, weighing approximately 145 pounds, wearing a blue sweatshirt with white lettering, a red shirt under the sweatshirt and whitewashed style blue jeans.

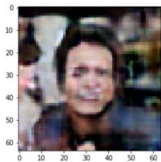
```
> /usr/local/lib/python3.6/dist-packages/torch/nn/modules/upsampling.py:129: UserWarning: nn.Upsample is deprecated.
  warnings.warn("nn.{0} is deprecated. Use nn.functional.interpolate instead.".format(self.name))
<matplotlib.image.AxesImage at 0x7fa51ddef320>
```



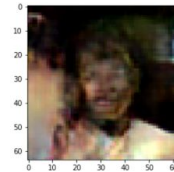
More generated examples can be examined on the next page.

More generated examples

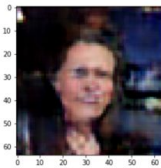
In this section, I include the relevant information such as their race, age, gender, hairstyles, and some of them I also include the emotion as well. And below are my generated outputs based on the input strings.



A white male , age around 30s , short black hair , pale complexion .



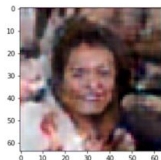
A black male , age around 40s , short hair .



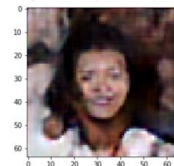
A white female , age around 40s , long hair , smiley face .



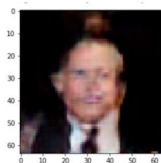
A white female , age around 30s , short blonde hair , happy face .



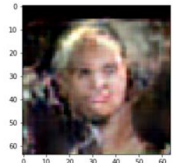
A latino female , age around 40s , short brown hair .



A black female , age around 40s .



A white male , age around 50s , blue eyes , receding hairlines .



A white male , age around 20s , bold hair , looks upset .

Reflection

In conclusion, if there's a future opportunity for this type of project, I would definitely want to improve more on it since the time span that we currently have to complete the project is quite limited. However, this is to give an idea how we could use these GAN(s) to generate the face of the criminal in multiple ways. Overall, this class has taught me a lot about how machine learning could be fun as well as creative. We often see concepts such as object detection, facial recognition, random prediction, etc. However, this class has shown that we could create something that is more than these general concepts.

REFERENCE

- [1] CycleGAN: <https://junyanz.github.io/CycleGAN/>
- [2] StackGAN: <https://github.com/hanzhanggit/StackGAN>
- [3] ProGAN: https://github.com/akanimax/pro_gan_pytorch

CODE: <https://github.com/ucsd-ml-arts/ml-art-final-kasidech-t-1>