[DeepLearning] Final Report

I Seob, Kim(20111635)
Hyun Seo, Nam(20121165)
Jeong taek, Lim(20131504)

June 20 2017

Abstract

From 2014, GANs(Generative Adversial Networks) has been popular, and even in 2016, DCGAN(Depp convolutional GAN) has been published so that GANs has been more famous in deep learning area. And there is another type of GAN, which is CGAN(Conditional GAN). We had been finding GAN examples and we found a pix2pix example, edges2shoes demo(https://affinelayer.com/pixsrv/). In this project, we used cGAN with image-to-image translation, and made a similar application that is implemented in the cGAN paper[3]. . We studied how this work and tried to apply for our fried chicken images. Before that, we already collected fried chicken images with crawling, and then convert chicken images to edge images. And we implement training with the images.

1 Introduction

The image-to-image deep learning has been a popular research topic in recent years. Mostly using CNN(Convolutional Neural Network), combine two pictures in one picture, or using GAN to make fake photo from sample data. As we were searching for interesting image deep learning, we found the image-to-image translation with cGAN paper. In this paper, there were explanation of Conditional Generative Adeversial Networks, and several interesting experiments. One of the examples is 'edge to photo', which transform simple sketch with only lines to photo. We were interested that algorithm, and example. So we decided to study cGAN, implement the algorithm, and test the similar experiment. And we, as live in dormitory, loved chicken, so we wanted to transform chicken sketch to chicken photo. The goal of this project is to colorize input image based on conditional Generative Adversarial Network(cGAN) model. We'll extract 'chicken' dataset from Google image and Instagram, then we'll learn this image set on this cGAN model [3]. In the case of a model in which a chicken image is learned, the real chicken image will be formed by inputting a uncolored chicken image.

Figure 1: Detected edge handbags, compared to ground truth.[1]

This model will translate the input image to real image that the discriminator can't recognize whether it is real or fake. The idea of discriminator and generating the real image is follows GAN model [1], and the idea of translation of image to image follows cGAN model [3].

2 Related Work

1. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. **Generative adversarial nets.** In NIPS, 2014.

2. 1. Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with **Deep Convolutional Generative Adversarial Networks** ICLR, 2016.

3. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with **conditional adversarial networks**. In CVPR, 2017.

Our project will proceed based on the paper [3], "Image-to-image translation with conditional adversarial networks", research that based on GAN and cGAN. GAN is deep learning model based on game theory, generate pixels and learn the weights by minimax function (maximize the probability of discriminating fake and minimize the probability of not being identified as fake). cGAN model receives input image conditionally. (the approximate shape) We'll follows this work.

3 Pix2Pix cGAN

We uses conditional generative adversarial network (cGAN). The cGAN is based on GAN [1]. The cGAN is conditional version of GAN, the generator and

discriminator are constrained by some condition. Feed additional input layer with the condition into generator and discriminator and the input are mixed as hidden representation.

3.1 GAN

The GAN is consist of two parts, the Generator and the Discriminator. The generator produces real-like images and try to decieve Discriminator and Discriminator is try to catch whether the image is real or generated image. The Generator want to know the distibution of datas to decieve Discriminator effectly, while the Discriminator estimate to check whether the input image for D is real or generated. We denote a function D(x), x is input for D, D(x)=1 when x is real image amd D(x) is 0 if not real image. and we denote G(z), z is noise for z and G(z) is generated image with noise z. So, If the GAN works properly, The Generator try to make D(G(z)) = 1, and the Discriminator try to make D(G(z)) = 0.

3.2 Generator

The generator has structure of encoder and decoder. The generator takes input image and make new image in order to generate target image. The output will be colorized image. The input is image with 256x256 arrays and has 3 RGB color channels and In each step, the input-output images is compressed and we want to make it smaller version of representation. Each encoder takes input and implement convolution and batch normalization and activation function (in this case, ReLU). "U-net"[4] for this process rather than encoder-decoder is better to improve the image to image performance. The skip connection make difference actually other is same. When skip connection used, it work differently. To train the generator, it generate an output image. After the discriminator guess whether the image is real or not, the weight of the generator is updated. The weights are adjusted based on the difference between the output and target image and the out putof the discriminator.

3.3 Discriminator

The discriminator takes two kind of images, first is input image and second is target image or generated image, and the discriminator will conclude whether the second image is generated image or not. The discriminator has concat part which concatenate two images in order to deal the two differenct images at once, and has encoders.The final output is 30x30 image with each pixel has value zero or one, which represents it is generated or real. The discriminator observes given image pair(input/target and input/generatot's-output) and make expectations about how they look like real. The weight are modified according to the error of the iput/output pair and the input/target pair.

4 Experiment Result

4.1 Input

At first the size of the all photo data is 512x256. The photo data is composed of two photos, each photo has 256x256 size. And the two are stitched horizontally. Also we focus on the edge model, analyzing edge photo, left one is edge object photo and right one is real object photo. Also we found that each photo data size is smaller than 70KB. To get an input image, it have to go through a conversion process. First, we have to adjust the size of the obtained chicken images to 256px and go through the edge detection process that detects only its edges. Finally, by combining two images (left one is edge detected, right one is resized), we got a chicken input image.

4.2 Libraries info

1. Beautifulsoup4(Python) – Crawling the web site and get images
2. Pil(Python) – Changing the size of images.
3. Scipy(Python) – Drawing the edges of the images.

4.3 Collecting images

At the first time to get a chicken images, we tried to crawl Google images, however, by Google's tight defense of crawling, we were able to get very small pixel size (less than 256px) of chicken images. These were useless. So we crawled some image repository sites (Pixabay, Stocksnap, Gettyimages), and we got about 1600 fried chicken images. Among them, some images have removed that are not obvious fried chicken, such as a person appears or mixed with other food. Then, we finally collected 400 fried chicken images. After training data with image-to-image pager, test the model with the trained data. At first, we input the specific information which data would be trained with command. And in the code, there are the default setting values of gan weight, layer weight, batch size, and etc. With setting values and data to be trained, we run the image to image deep learning following the paper. And then, we type the testing command with the data to be tested and trained data. Then, load the configuration and evaluate the test images.

4.4 Result

We tested 88 images, which is 20% of whole data, and got 88 results. Some results of our experiment with chicken data is below. With trained data, test image is processed and provide the image similar to the original one.

Figure 2: Results of the test images with trained image (left one is edge image, middle one is processed image with trained data, right one is original image)

(Running environment CPU : i7 GPU : NVIDIA Gefore GTX 970
OS : Windows 10 Python version : ver 3.5 Tensorflow version : tensorflow-gpu 1.10)

4.5 Demo

We made an demo version of the test. By uploading uncolored image and the process button allows corresponding output image to be generated and appeared on the screen.

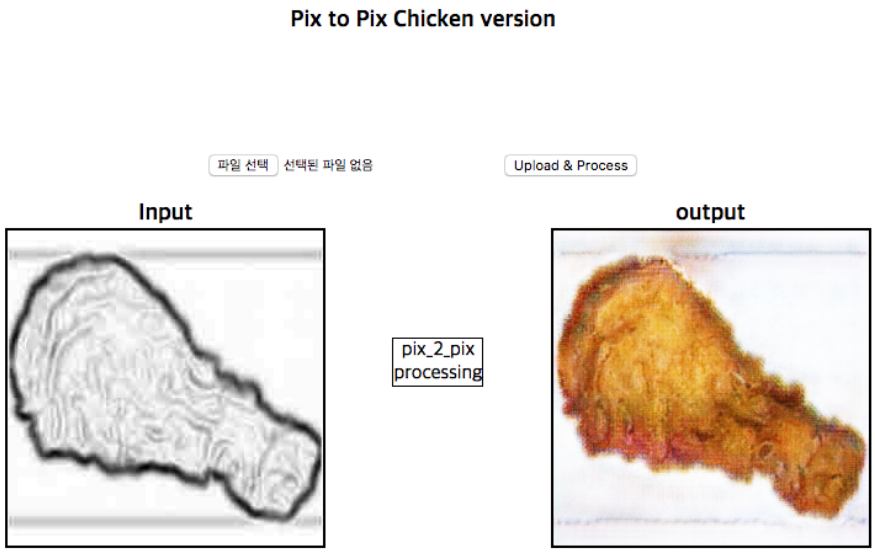

Figure 3: Demo

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.

[2] Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks ICLR, 2016.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In CVPR, 2017.

[4] I.Olaf Ronneberger, Philipp Fisher, Thomas Brox.U-Net: Conveolutional Networks for Biomedical Image Segmentation. MICCAI 2015

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image

[6] Christopher Hesse, Image-to-Image demo, https://affinelayer.com

[7] Christopher Hesse, tensorflow-pix2pix, https://github.com/affinelayer/pix2pixtensorflow

[8] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, Alexei A.Efros Conext Encoders: Feature Leaning by Inpainting 2016.

[9] Diederik P.kingma, Jimmy Lei Ba ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. In ICLR, 2015. Diederik P.kingma, Jimmy Lei Ba ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. In ICLR, 2015.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskerver, Rusalan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting journal of Machine Learning Research 2014.

[11] Sergey Ioffe, Christian Szegedy Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043.