CS230

Makeup Removal System with Deep Learning

Mingchen Li
Department of Electrical Engineering
Stanford University
limc@stanford.edu

Yiyang Li
Department of Electrical Engineering
Stanford University
yiyang7@stanford.edu

Yifan He
Department of Electrical Engineering
Stanford University
heyifan@stanford.edu

Abstract

Makeup is widely accepted by the public for people to improve facial attractiveness. However, it remains a challenge for systems to remove makeup from people's faces and match the makeup and makeup-removed images of a person. This paper proposed a WGAN-GP approach to the makeup removal task. We gathered nearly two thousand of makeup dataset and trained the system accordingly. The experimental results prove that the system is able to remove makeup from a with-makeup image of a person.

1 Introduction

Facial makeup has been ubiquitous in our daily life and social networks. It is quite common for people to wear makeup to hide facial flaws and have more attractive appearances. However, the use of makeup poses a significant challenge to face verification, since facial cosmetics can sometimes be deceivable and hide one's original appearance. In this project, we provide a solution to this challenge: a makeup removal deep learning system which can erase people's makeup. This novel system carries many practical applications, such as face authentication on security systems, face identifier on social network apps, etc. We built a WGAN-GP to perform the task. The inputs to the generator are images with makeup, and the inputs to the discriminator are images without makeup and the generated images from the generator. The outputs of the system are the removed makeup images from the generator.

2 Related work

The problem of image-to-image translation has been studied by many research groups. Recent approaches proposed by [10] for image-image translation is to use a dataset of input-output examples to learn a parametric translation function using CNNs. Zhu's approach to this problem is to build on the "pix2pix" framework of Isola et al. [11], which uses a conditional generative adversarial network to learn a mapping from input to output images. Unlike CNNs, Zhu's method learn the mapping without paired training examples.

Li [5] proposed a learning algorithm from generation approach for makeup-invariant face verification by introducing a bi-level adversarial network (BLAN). This network reduce the sensing gap between makeup and non-makeup images. It achieves state-of-the-art verification accuracy across makeup

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

status and can produce non-makeup images. We referred to this paper for loss functions in this project.

GAN is one of the most successful deep generative models and is applied for image-to-image translation, but suffers from training instability. The recently proposed Wassertein GAN (WGAN) improves the stability of GANs training, but sometimes may still fail to converge. Gulrajani [4] proposed alternative to clipping weights: penalize the norm of gradient with respect to its input and this method performs better than standard WGAN and enable stable training almost hyperparameter tuning . We referred to this paper for building the WGAN model.

3 Dataset and Features

The dataset we collected consists of five separate datasets (FAM, MIFS, VMU, MIW, YMU) of total 2600 images of 1300 different people. The dataset is collected from [1], [7], [8] and [9]. Each person has two images: one with makeup and the other one without. Some people may have more than one images with makeup.

Since the datasets may contain some useless photos, we manually went through all the data, and threw away the bad ones. One example of the bad data is a woman uses makeup to make herself look like a man. Even human cannot tell this person is originally a woman, so we discarded this example. We flipped our images left to right for data augmentation. When training the model, we resized all images to the size of 96 * 96 * 3 using Matlab.

4 Methods

To approach this problem, we first built some baseline models (convolutional neural networks (CNNs/ConvNets) and inception models) and then built a generative adversarial networks (GANs) model to remove the makeup from the image.

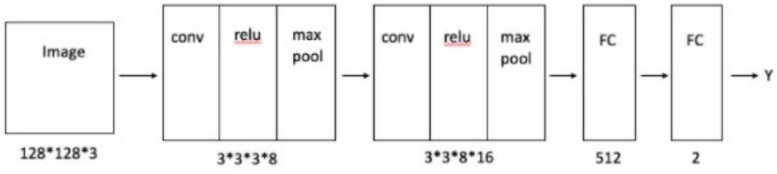


Figure 1: CNN Model for Classifying Images with/without Makeup

We first built a CNN model to classify whether a person is with or without makeup. This CNN model was intended to be a testing model for the discriminator of the future GAN.

We tried different complexity of architectures, including one to two convolutional layers before pooling layers, two to four groups of conv-pooling layers, one to two fully connected layers, the final architecture was settled, which is shown in Figure 1. The best accuracy we could reach from this model was 80% on training and 79% on testing.

Since the model output a 2*1 vector of 0 and 1, the loss function we used was softmax cross entropy instead of sigmoid cross entropy, which is showing in equation (1). This choice of loss function might cause an over parameterization in the model. However, since this was only a baseline model, and this problem was not critical, we chose to run the testing and moved on to GAN.

$$L^{(i)} = \frac{1}{m} \sum_{i=0}^{m} \sum_{j=0}^{1} y_j^{(i)} \log \hat{y}_j^{(i)}, \tag{1}$$

where m is number of examples, i is index of example, and j represents if the image is with(1) or without(0) makeup.

Then we built an inception model to retrieve images' encoding (shown in Figure 2). This model was intended to provide a part of the loss function for the generator of GAN. However, we didn't actually

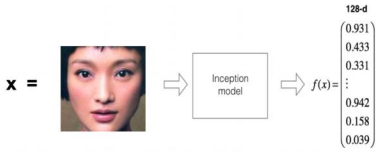


Figure 2: Inception Model for Encoding Retrieval

implement it in our final model, because the function required numpy inputs and gave numpy outputs, but the GAN was implemented using tensorflow which required tensor inputs. We found a solution to convert numpy vectors to tensor (tf.py_func), but we didn't implement it successfully due to time constraint.

The intended loss function provided by the encodings was to take L2 norm between the feature encodings of a real without makeup face and a generated face. The formula is shown in equation (2):

$$L = \|E(G) - E(R)\|_2, \tag{2}$$

where E(G) is the encoding of generated images, and E(R) is the encoding of real images.

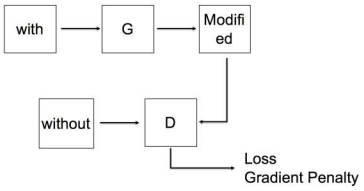


Figure 3: GAN Model for Removing the Makeup from Images

After training baseline models and preparation work, we built the GAN model to remove the makeup from the image, as shown in Figure 3. GANs are deep neural network architectures comprised of two networks, generator and discriminator. Our final model is a WGAN-GP.

The input to the generator is an image with makeup, and it will output an modified image which gets the makeup removed and will be used as an input for the discriminator. The discriminator also has the original image without makeup as its other input.

Both images may not enter the discriminator at the same time. The ideal output for the discriminator is 1 when input is a true image without makeup, and 0 when input is a modified image from the generator. The ideal state for the generator is that the discriminator will output 1 when its input is a modified image, which means that the discriminator cannot distinguish between modified images and true images without makeup.

The architecture of the generator consists of a U-NET which provides direct connection between a convolutional layer and a de-convolutional layer at symmetric positions. Our loss function of the generator contains two parts. The first part is the opposite number of the output of discriminator on the generated image. The second part is a L1 pixel to pixel difference between the generated image and the real without makeup image. This second part of the loss is adopted from [5]. This should enforce the generator to generate images as closed to the real image as possible. The total loss function for the generator is showing in equation (3):

$$L_G = -\frac{1}{m}\sum_{i=0}^{m} D(G(I^{w^{(i)}})) + \frac{1}{m}\sum_{i=0}^{m} \|G(I^{w^{(i)}}) - I^{wo^{(i)}}\|_1, \tag{3}$$

where m is number of examples, i is the index of each example, I^w is with-makeup, I^{wo} is without-makeup, $G(I^w)$ is generated image from generator, and $D(G(I^w))$ is output of discriminator on the generated image.

The architecture of the discriminator is a simple four-stack of convolutional layers. The loss function of the discriminator also consists of two parts. The first part is the combination of its output on the

generated image and the opposite number of its output on the real image. The second part is a gradient penalty for the discriminator, which is the "GP" in the WGAN-GP name. The gradient penalty term takes the square of the difference between the L2 norm of the gradient of the discriminator and 1, and the whole term multiplies by a coefficient. This should constrain the discriminator from learning too rapidly and defeating the generator too quickly. The total loss function for the discriminator is showing in equation (4):

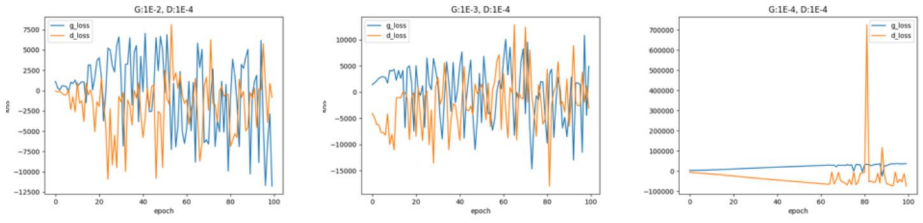
$$L_D = \frac{1}{m}\sum_{i=0}^{m}[D(G(I^{w^{(i)}})) - D(I^{wo^{(i)}})] + \lambda(\| \bigtriangledown D(\hat{I})\|_2 - 1)^2, \tag{4}$$

where λ is a coefficient for gradient penalty. We used λ = 10 as proposed in [4]. $\hat{I}$~$P_{\hat{I}}$ are random samples. $P_{\hat{I}}$ is sampling uniformly along straight line between pairs of examples sampled from the data distribution I^{wo} and generator distribution $G(I^{w})$ [4].

5 Experiments

Before we adopted the WGAN-GP as our final model, we've tried different models. First of all, we built a GAN from scratch by ourselves, but the results weren't satisfying for it output images with color pixels that seemed random to human eyes. We then adopted BicycleGAN model from [3] with some modifications, but the results were still unsatisfying. Finally, with the suggestion from our mentor Russell, we built the model using WGAN-GP from [4]. Although our final implementation is built upon the WGAN-GP code with modifications, the Github link we submitted contains all the trials we have done to show our efforts.

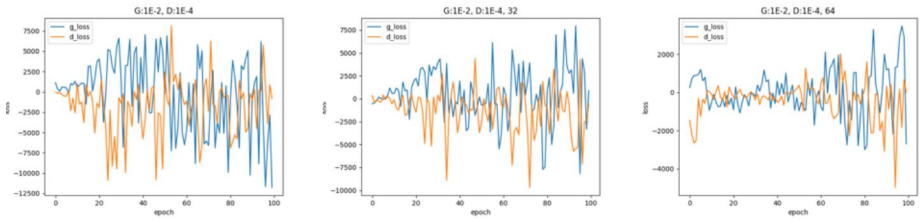
We performed several experiments to choose the hyperparameters, including mini batch size and learning rates for both generator and discriminator. We were aware that training GAN is very hard that we have to perform lots of hyperparameter tuning. Given the limited time of the quarter, we've tuned the model the best we could. Since this project didn't have an accuracy metric, we used loss function as our primary metric. Figure 4 below are the generator and discriminator loss on different learning rates with mini batch size of 16 during 100 epochs. Figure 5 shows the loss with different mini batch sizes with same learning rates. Note that the system runs 400 epochs during actual training, but since 400 epochs takes too long to run, we only performed 100-epoch experiments. We chose to use mini batch size of 16, generator learning rate of 1E-2 and discriminator learning rate of 1E-4. According to the figures, both loss functions fluctuated heavily, and the discriminator has a lower average loss than that of generator. However, we can also notice that the performance of generator improved overtime that managed to have a lower loss eventually in Figure 4(a) and 6(a). Since this was just a 100-epoch experiment, the generator was actually getting better as the system iterated 400 times.



(a) G learning 1E-2 (b) G learning 1E-3 (c) G learning 1E-4

Figure 4: Learning rate experiments

Besides the plots of loss functions, the other metric to help us choose the hyperparameters was the actual results. Since this was a makeup-removal system, we went through the results of different combinations of hyperparameters, and looked at the output images by eyes and determined which combination was the best.

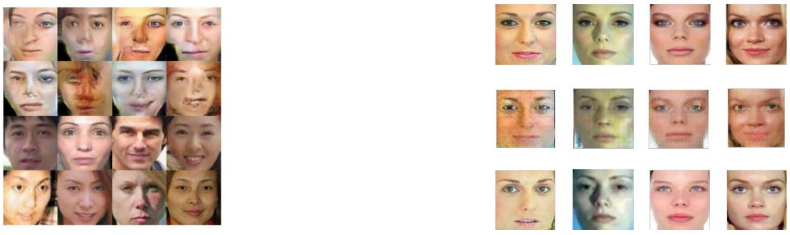


(a) Mini batch size 16 (b) Mini batch size 32 (c) Mini batch size 64

Figure 5: Mini batch size experiments

6 Results

Before actually inputting images to the GAN, we first tried to use random noises as inputs to test the capability of the GAN. One of the results of noisy input is showing in Figure 6(a), with the input to the generator as random noise and the input to the discriminator as without-makeup images.



(a) Results of GAN with random noise input (b) Results of GAN with image input

Figure 6: Results of the system

The results of the system are showing in Figure 6(b). The first row contains the with-makeup images that were input into the generator. The second row shows the makeup-removed images generated from the generator. The final row contains the ground truth images of the same people without makeup. The generated images are different with the ground truth image in poses and facial expressions, because the pair of photos of a same person in the dataset were taken in different poses and facial expressions.

7 Conclusion

We find the current results satisfying, given the limited time of the project, the difficulties of training a GAN and the fact that makeup removal is a relatively new area with only a few papers for us to refer to. However, the model is not perfect. There are still issues with this model that need to be solved.

Our Github link:
http://github.com/yiyang7/cs230Proj_makeup_removal_system

8 Future Work

First of all, we need to continue tuning the hyperparameters. Besides, we can try to alter the architectures of the generator and discriminator. Thirdly, we can try to refine the loss function. For example, since the human faces are symmetric at most of the time [5], we can add a symmetry loss to the generator. The loss takes difference between two horizontally symmetric pixels. We believe makeup removal is a practical and widely applicable area, and more work should be done in this area.

9 Contributions

Mingchen Li:
Implemented finetune_makeup.py deriving from Olivier's finetune code; implemented self-build CNN binary classifier makeup_bin.py; manually went through the dataset to get rid of useless data; implemented self-built GAN GAN_1.py; implemented testing WGAN wgan_gp.py and the final program train_wgan_GP.py based on the original WGAN-GP code; applied modification on all the related files including model.py, layers.py, layer_old.py, etc; cooperated in writing the report of Experiments, Results, Conclusion and Future work.

Yiyang Li:
Pre/post-processed the images of makeup dataset (cropImg.m); Helped implement makeup.py; Debugged and tuned the neuron network; Built an inception model to retrieve images' encodings and perform face recognition; Helped train GAN model; Cooperated in writing the report of Related Work and Methods.

Yifan He:
preprocessed the images of makeup dataset; implemented cropImg.m; Implemented finetune_makeup.py deriving from the TA Olivier's finetune code; built an inception model to retrieve images' encodings and perform face recognition; tuned the parameters of pre-trained model; help training WGAN wgan_gp.py; cooperated in writing the report of Introduction, Dataset and Features.

References

[1] Dantcheva, A., Chen, C., & Ross, A. (2012) Can Facial Cosmetics Affect the Matching Accuracy of Face Recognition Systems? *5th IEEE International Conference on Biometrics: Theory, Applications and Systems*

[2] Wang, S., & Fu, Y. (2016). Face Behind Makeup. *Association for the Advancement of Artificial Intelligence*, pp. 58-64.

[3] Jun-Yan, Z., Richard, Z., Deepak, P., Trevor, D., Alexei, E., Oliver, W. & Eli, S. (2017) Toward Multimodal Image-to-Image Translation. *Computer Vision and Pattern Recognition*

[4] Ishaan, G., Faruk, A., Martin, A., Vincent, D. & Aaron C. (2017) Improved Training of Wasserstein GANs. *Learning*

[5] Yi, L., Lingxiao, S., Xiang, W., Ran, H. & Tieniu, T. (2017) Anti-Makeup: Learning A Bi-Level Adversarial Network for Makeup-Invariant Face Verification. *Computer Vision and Pattern Recognition*

[6] Jun-Yan, Z., Taesung, P., Phillip, I. & Alexei, E. (2018). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *Computer Vision and Pattern Recognition*

[7] Chen, C., Dantcheva, A., & Ross, A. (2013) Automatic Facial Makeup Detection with Application in Face Recognition. *6th IAPR International Conference on Biometrics*

[8] Chen, C., Dantcheva, A., & Ross, A. (2016) An Ensemble of Patch-based Subspaces for Makeup-Robust Face Recognition. *Information Fusion Journal*, Vol.32, pp. 80-92

[9] Chen, C., Dantcheva, A., Swearingen, T., & A. Ross, (2017) Spoofing Faces Using Makeup: An Investigative Study. *3rd IEEE International Conference on Identity, Security and Behavior Analysis*

[10] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks forsemantic segmentation. In *CVPR*, pages 3431-3440, 2015. 2, 3, 6

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2, 3, 5, 7, 20