Styled Makeup Synthesis and Removal Using Generative Adversarial Networks

Berkin İnan
Bilkent University
Ankara, Turkey
berkin.inan@ug.bilkent.edu.tr

Ege Turan
Bilkent University
Ankara, Turkey
ege.turan@ug.bilkent.edu.tr

Emre Sülün
Bilkent University
Ankara, Turkey
emre.sulun@bilkent.edu.tr

Abstract

Applying makeup is a laborious and proficiency requiring task. Trying out new makeup styles requires specific cosmetic products, time and creative skill to divert from the reference makeup to create a new makeup. In this paper, we tried to adapt a deep learning technique that can produce new makeups similar to a given makeup style and has the ability to modify it. We propose using a synthetic labeling technique to label the makeup images in MT-Dataset, a makeup dataset created by BeautyGAN [10], and train a Generative Adversarial Network (GAN) with it to create new makeups of a desired style. After generating a makeup with a certain style, we apply the makeup to the non-makeup image using PSGAN [8]. However, training of the GAN did not succeeed and we could not get realistic makeup images from the GAN. In addition to makeup generation, we present a novel technique for removing makeup from facial images. The technique is based on Pix2Pix [7]. To improve the reproducibility, we share our source code, it code can be found in `https://github.com/InanBerkin/Styled-Makeup-Synthesis-and-Removal`

1. Introduction

Makeup editing is one of the popular research topics for computer vision researchers who are interested in facial images. There are different challenging factors in facial images such as varying pose and expressions. Similarly, makeup editing studies try to overcome these challenges.

Makeup editing include different tasks like generating makeup, removing makeup, transferring makeup etc. In this paper, we focus on generating and removing makeup problems. We try to adapt a logo generation technique (Logogen [13]) for makeup generation task.

We make the following contributions in this paper:

- A ResNet-50 Model with transfer learning to extract the makeup features
- An architecture for generating makeup images with certain styles
- Makeup removal model with Pix2Pix using input and output images of PSGAN

We also share our code and dataset in GitHub for readers who are interested to reproduce the study.

The rest of the paper is organized as follows: In Section 2, we explain the background of makeup manipulation field. In Section 3, we explain our approach in detail and give the details of architecture. In Section 4, we present the dataset, metrics and results. Finally, in Section 5, we conclude our work, and provide the details of each author's contribution.

2. Related Work

Makeup manipulation on facial images has been researched since 2009. Manipulation operations include applying, removing, transferring and generating makeup *(synthesis)*.

Guo and Sim [4] proposed a method for makeup transfer by decomposing images into three layers and transferring each layer one by one. The major advantage of their method is that it can transfer without requiring multiple images and can be achieved with a single image.

The first utilization of deep neural networks for visual attribute transfer is done by Liao *et al*. [11]. Their work is not limited makeup transfer and can be used for other visual attributes. The main advantage of their method is effectively transferring attributes where image pairs have signif-

icant variance in their appearance including lighting, color, texture, and style.

Generative adversarial networks (GAN) have an increasing popularity in research community for makeup manipulation. PairedCycleGAN proposed by Chang *et al*. [1] involve two asymmetric functions, one for makeup transfer and one makeup removal. They construct two coupled networks to implement those functions. The output of the first first function can be used as an input of the second function and the final output matches the first input.

Another method based on GAN is BeautyGAN [10]. Similar to PairedCycleGAN, BeautyGAN can be used for makeup removal in addition to makeup transfer. BeautyGAN involves dual input and dual output, and introduces a makeup loss for instance level style transfer.

Gu *et al*. [3] introduces LADN that uses multiple overlapping local discriminators for dramatic makeup transfer. The makeup transfer network of LADN separates makeup dependent variables and leads photorealistic results where facial identity is largely preserved.

A distangled representation is firstly proposed by Zhang *et al*. [15]. Their approach decomposes the original input into several independent hidden codes so that the features of each component can be better learned. The main motivation is achieving different scenarios such as pairwise, hybrid and multimodal transfer.

Chen *et al*. [2] developed BeautyGlow framework that extends Glow architecture [9]. The authors state that although the proposed framework is used for makeup transfer, it is possible to extend it to other applications that require decomposing the latent image vector into two latent vectors such as rain or fog removal.

In order to solve over-fitting problem on makeup transfer domain, Jiang *et al*. [8] developed the pose and expression robust spatial-aware GAN (PSGAN). The main contribution of PSGAN is achieving makeup transfer regardless of pose and facial expression.

In a different domain of GAN, Logo-Gen by Sage *et al*. [13], proposed synthetic labeling and generating diverse logos by latent space exploration.

Logo-Gen also constitutes the baseline of our paper, as we re-implement the ideas in that paper for makeup images, while using parts of PSGAN to apply the makeup.

3. Proposed Approach

We aim to generate makeup images with certain style and use the generated images as a reference image for a makeup transfer network, namely PSGAN. First, we label the makeup images with a re-trained ResNet-50 model by their makeup features. Then, using the labeled images, we train a DCGAN to generate makeup images belonging to a certain makeup style. Our generative model is based on Deep Convolutional Generative Adversarial Networks (DCGAN) of Radford *et al*. [12]. Our proposed architecture is illustrated in Figure 3.

We have decided to use conditional GAN for the makeup removal. In this approach, we are using our training results from the PSGAN ad combining it with the input images of the PSGAN. We have decided to use Pix2Pix [7] as our conditional GAN architecture and trained our model with output of the PSGAN.

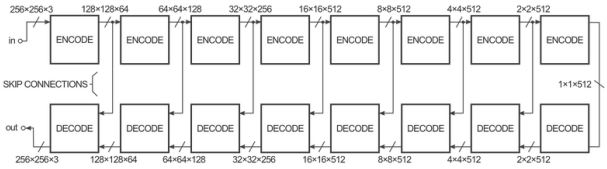

Figure 1. U-Net Architecture

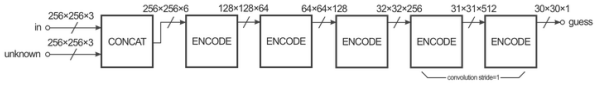

Figure 2. PatchGAN Architecture

Conditional GANs (cGANs) means that the cGAN can learn a conditional generative model with the difference of the way that the loss is learned. The loss of cGAN called as structured loss and it penalizes any structure that differs between the output and the target. Another distinction of cGAN is the inputs to the generator. GANs give only x as input to the generator. However, cGANs' provides along with a random noise, z. Thus, cGAN becomes able to provide diversity on the generated outputs, rather than producing deterministic outputs. In the model, architectural implementation of the generator and the discriminator is an important issue. The generator is implemented as "U-Net" based architecture while the discriminator implemented as "PatchGAN" architecture. For U-Net based architecture. Observe that the difference of U-Net based architecture is the skip connections. Each layer is connected to the mirrored layers which are lied on the decoder stacks. The aim of U-Net based architecture is to align the input structure with the output structure if any problem occur such as unmatched resolution. Thanks to skip connections, U-Net Based Architecture saves time by making the training more efficient and improves the performance. To examine the difference, generator architecture without skip connections is provided in figure.

3.1. Synthetic Labeling

To be able to generate makeups with a desired style, we need to cluster and label the makeup images based on their

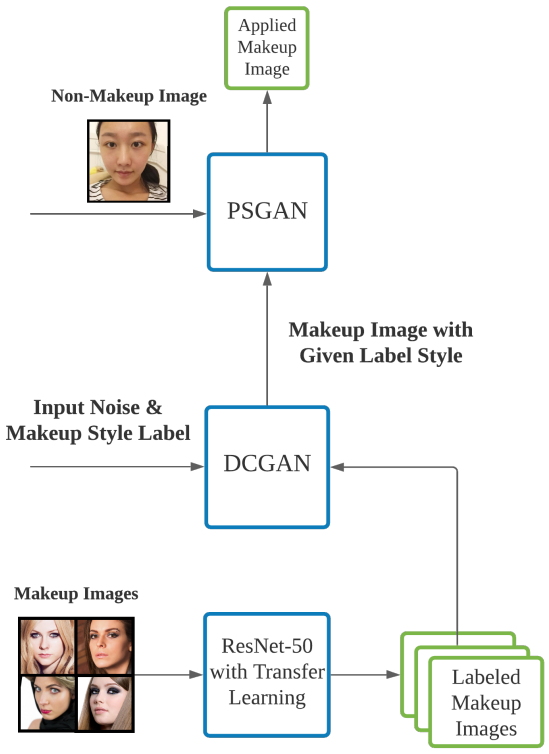

Figure 3. Proposed Architecture For Styled Makeup Synthesis

styles. Although the dataset we use, MT-Dataset, includes various makeup styles, such as “smoky-eyes”, “flashy”, “retro”, “Korean”, and “Japanese”, the images are not labeled as such. As we could not label all the images by hand, we used a synthetic labeling technique. We use a similar labeling technique used in Logo-Gen.

To label the images, we used ResNet-50 model [5] and extracted the features of the images. We perfomed feature extraction on ResNet-50 and re-trained the final layer weights to detect makeup and non-makeup images. Input to the ResNet-50 model are the makeup images and the output is the output of the final pooling layer. The output vector has a size of 2048 and we use this vector to cluster the images with similar features.

We apply principal component analysis (PCA) to reduce the dimensions of the 2048 feature vectors. After applying PCA, we cluster the data by using k-means clustering. However, obtained clusters were not highly distinct as the feature extraction is not applied to makeup but the whole image. This results in clusters with images where the subjects have similar hair color or skin tone. An example cluster can be seen in Figure 4.

Figure 4. Example of a cluster generated by the synthetic labeling. Common features of these images can be ”Korean” makeup style.

3.2. DCGAN

For our DCGAN implementation, we used the DCGAN implementation of Logo-Gen by Sage *et al*. and wrote it with Pytorch’s DCGAN tutorial [6]. We trained the DCGAN with the MT-Dataset which was synthetically labeled. The architecture of the DCGAN used can be seen in Figure 5.

In our model, we create one-hot vectors from the labels and append that to the layer input. For linear layer, we append the one-hot vector to the input as is. For convolutional layers, the one-hot vector is transformed to a one-hot feature map, where the channel at the corresponding label is all ones, while other channels contains all zeros.

For the input to the generator, we generate a latent vector of size 512 from the standard normal distribution. We feed this noise to the generator and the generator returns an 64x64 makeup image output. For the discriminator, we pass the real images with their generated labels in the first update of the discriminator network. After the generator generates the fake makeup images, we pass the fake images to the discriminator with their labels and update the discriminator a second time.

3.3. Loss Function

For the loss function of DCGAN, we used a binary cross entropy(BCE) loss. BCE calculates how different the distribution between the output of network (generator discriminator) and the label vector. It uses mean reduction, which divides the sum of all l_n by the number of elements in the output. Equation can be seen in equation below, where x_n is the output and y_n is the label.

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top$$
$$l_n = -\left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)\right]$$

3.4. Makeup Removal

Our main training approach for at the end is maximize the discriminator while minimizing generator. The main

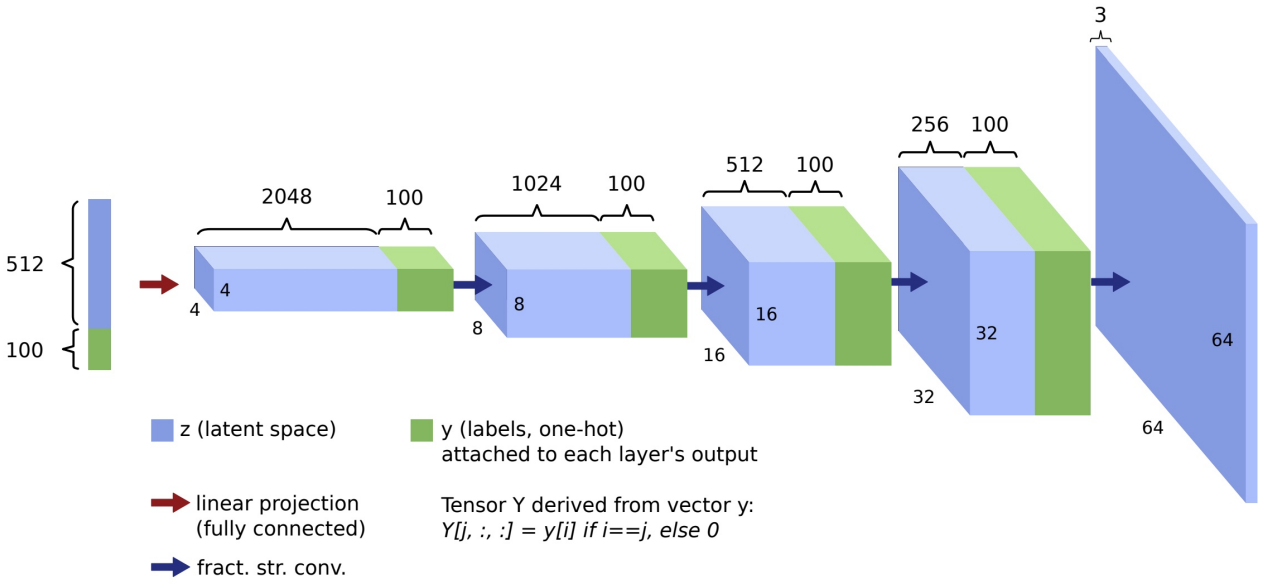

Figure 5. Architecture of DCGAN [13]. In the case of 100 clusters, a vector of length 100 is appended to the input of size 512 and a one-hot feature maps are appended to the convolutional layers.

reason for maximizing the discriminator is to accurately evaluate the image as real or fake. As discriminator gets better and better, the generator also learns from the discriminator and tries to minimize itself. While doing so, the algorithm applies stochastic gradient descent in each step of discriminator and updates the generator with each step of the discriminator (Eq.1). The general objective for the model is given in equation below.

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z))]$$

Notice that in Eq.3.4, G takes two arguments which is x, the observed image, and z which is a Gaussian noise vector to provide diversity in the generation process. As mentioned before, the model tries to minimize the generator while maximizing the discriminator which leads us to the following and final equation, which is below.

$$G^* = \arg\min_G \arg\max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \quad (1)$$

4. Results / Experiments

In this section, we present the dataset we used, the details of experiments that we did throughout the semester and the results of the experiments.

4.1. Dataset

We used *Makeup Transfer* dataset to train and test our approach. The dataset consists of 3834 facial images, 1115 of them have no makeup while the remaining 2719 images have makeup applied. All the images have the same resolution which is 361 x 361. *Makeup Transfer* dataset is also used by PSGAN [8], BeautyGlow [2] and BeautyGAN [10].

After that we used *Makeup Transfer* dataset for obtaining makeup images. Then combining outputs of PSGAN and original Makeup Dataset we created new dataset for training Pix2Pix.

4.2. Synthetic Labeling Experiments

For the synthetic labeling, we had to encode the makeup in the makeup images to a vector where similar makeups have similar vectors. First, we used a pre-trained ResNet-50 model to extract the features of the images. To measure to accuracy of the model, we ran our model on a test dataset to see if it can distinguish makeup and non-makeup images.

Then, we re-trained the final layer weights of ResNet-50 with makeup and non-makeup images and performed transfer learning [14]. To train, validate and test our newly trained network, we created sets of images consisting of makeup and non-makeup images. We resized the images to 224x224 and trained the model for 50 epochs, optimized by stochastic gradient descent, with learning rate of 0.001 and batch size of 128. The pre-trained model had an accuracy of 30% while the re-trained model had an accuracy of 91% detecting makeup and non-makeup images.

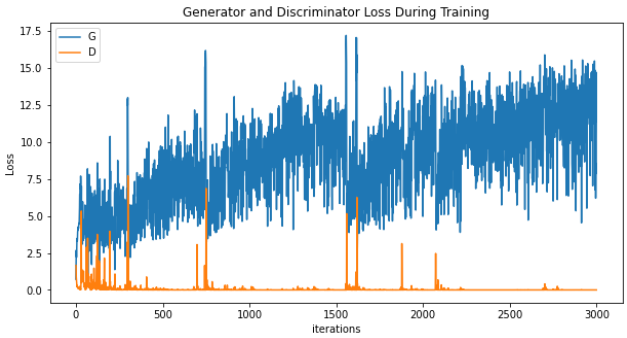

Figure 6. *L*oss Graph of Generator and Discriminator of DCGAN

Comparing the clusters of two models, the results were not significantly different. The newly trained model could distinguish between makeup and non-makeup images with high accuracy but were not enough to provide significantly distinct makeup clusters.

4.3. Results

Although we have changing the layer parameters and the architecture, we could not train our DCGAN model as we liked. We expected the results to be makeup images with a style similar to the images in the cluster with the same label. However, the generated images were not meaningful and therefore we could not use the output images as a reference image for the makeup transfer. We trained our model with 64x64 images, 100 epochs, batch size 128 and a learning rate of 0.0002. The loss graph for the training with 3000 iterations with batch size 128 can be seen in Figure 6

One of the reasons why our GAN training failed can be that the Logo-Gen's DCGAN architecture may not be suitable for generating makeup images. Logo images were abstract illustrations while makeup images we needed to produce have to have human faces. Another possible reason is that 64x64 sized images are not good for training a GAN for human faces with details. Implementing a new DCGAN that can accept 256x256 images and output at the same may produce better results. Another reason can be that appending one-hot vectors and feature maps to layer outputs may corrupt the image data and can be the reason why images contain seemingly random colors.

An example image set with generated makeup is given in Figure 7.

For the training makeup removal in Pix2Pix. We have acquired following data chance.

4.3.1 Explanation of the data

If we examine the loss functions, we can observe that sometime loss spikes up and the reason for that is if the Discriminator sees an image that includes an object which Discriminator has not seen before, it does not recognizes the object and counts as error. For example in the loss graph of the model when its trained with Summer-to-Winter dataset, $GL1$ loss seems to spike up when model encounters an image that contains bridges or any other uncommon object that the model is not familiar with.

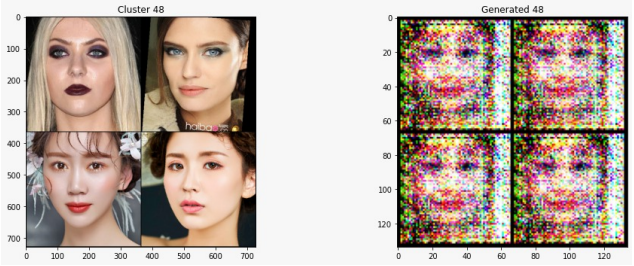

Figure 7. *Left:* Images clustered by Resnet, *Right:* Images generated by DCGAN

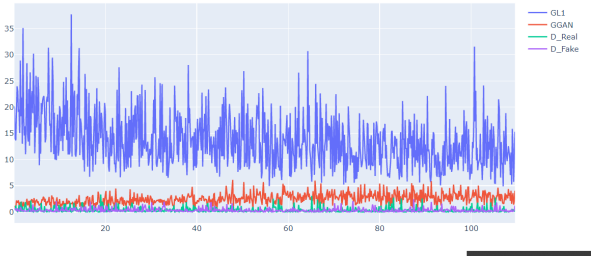

Figure 8. Loss Data of Makeup Removal

Examine one of the loss plots, e.g. Figure 8, there are four labels for each loss. Two of them are for the discriminator, D_Real and D_Fake and where the other two are for the generator, $GGAN$ and $GL1$. Label $GGAN$ represents the loss of model's generated image between the ground truth. $GL1$ represents the L1 distance of the generated image from the random noise. Since it takes a random noise (z) for input, examine Eq. 3.4, it is expected to be clamorous. In the discriminator side, D_Real represents the loss for real images while D_Fake represents the loss for fake images, as the names suggest. Since PatchGAN runs across the image convolutinally, it takes the averages for the final decision. For instance, if D_Real is greater than D_Fake, discriminator decides the image is real, or vice versa.

4.3.2 Explanation of training

In Figure 8, we have showed how does the makeup on the original image changed for epochs. We have trained our Pix2Pix model for 100 epochs and in this figure we showed every 25 epoch effect on the makeup removal. As seen from the figure, epoch 100 gives the best result and makeup re-

Figure 9. Changes of Makeup Removal [13]. From left to right, Original Image, Epoch 25, Epoch 50, Epoch 75, Epoch 100

Figure 10. Comparison of makeup removal with PairedCycleGAN

moval applied succesfully and original image features are preserved.

5. Conclusion

In this paper, we developed a new technique for applying custom styled makeup to non-makeup images. We used ResNet-50 Model to achieve that. We tried to train a DCGAN model for generating makeup images with the generated labels but failed to generate meaningful images.

Another contribution of this paper is proposing a novel method for removing makeup from facial images generated by PSGAN. The removal operation is done with Pix2Pix.

5.1. Contributions

Berkin Inan: Implemented the synthetic labeling for makeup images and DCGAN architecture of Logo-Gen in PyTorch and modified it for makeup generation.

Ege Turan: Trained PSGAN for the sample dataset and used output of that dataset on the Pix2Pix for 100 epochs. Using PyTorch acquired checkpoint based removal results.

Emre Sülün: Experimented with different parameters to improve the output of synthetic labeling and DCGAN, wrote Related Work section.

References

[1] H. Chang, J. Lu, F. Yu, and A. Finkelstein. Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 40–48, 2018.

[2] H.-J. Chen, K.-M. Hui, S.-Y. Wang, L.-W. Tsao, H.-H. Shuai, and W.-H. Cheng. Beautyglow: On-demand makeup transfer framework with reversible generative network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10042–10050, 2019.

[3] Q. Gu, G. Wang, M. T. Chiu, Y.-W. Tai, and C.-K. Tang. Ladn: Local adversarial disentangling network for facial makeup and de-makeup. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10481–10490, 2019.

[4] D. Guo and T. Sim. Digital face makeup by example. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–79. IEEE, 2009.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] N. Inkawhich. Dcgan tutorial - pytorch tutorials 1.7.1 documentation, 2018.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[8] W. Jiang, S. Liu, C. Gao, J. Cao, R. He, J. Feng, and S. Yan. Psgan: Pose and expression robust spatial-aware gan for customizable makeup transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5194–5202, 2020.

[9] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[10] T. Li, R. Qian, C. Dong, S. Liu, Q. Yan, W. Zhu, and L. Lin. Beautygan: Instance-level facial makeup transfer with deep generative adversarial network. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 645–653, New York, NY, USA, 2018. Association for Computing Machinery.

[11] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017.

[12] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[13] A. Sage, R. Timofte, E. Agustsson, and L. V. Gool. Logo synthesis and manipulation with clustered generative adversarial networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[14] Cs231 convolutional neural networks for visual recognition.

[15] H. Zhang, W. Chen, H. He, and Y. Jin. Disentangled makeup transfer with generative adversarial network. *arXiv preprint arXiv:1907.01144*, 2019.