Generation and Completion of Human Face Images

CS 230 Final Project, Winter 2018

Yijun Jiang **Yi Liu** **Zhengli Wang**

Github repository: https://github.com/Yi-61/Image_Completion_CS230

1 Introduction

Image completion is an active area of research with many important applications such as image editing, demosaicing and super-resolution. It is challenging because an image-completion algorithm not only needs to "infer" the missing parts from the neighboring pixel information, but also needs to "learn" the content from the existing parts of the image. In the past, many classical approaches had been proposed but were not satisfying. This motivates us to tackle image completion using a deep learning framework.

Image completion is essentially a generative problem, for which an appealing model is the generative adversarial network (GAN). A GAN consists of two networks competing against each other in a zero-sum game framework. This system is known to be able to generate images that look natural to humans [1]. Moreover, algorithms based on GANs have been proven successful in inferring missing pixels in a given image [2].

In this project, we established a deep-convolutional GAN (DCGAN) structure and trained it to generate natural human face images. With the trained network, an image completer was designed to fill out missing pixels in a target image from the test dataset. Despite the lack of details, the completer was able to generate completions that are correct both contextually and perceptually.

2 Dataset

RGB images of human faces were obtained from the Large-scale CelebFaces Attributes (CelebA) Dataset [3]. This dataset contains 10,177 identities, 202,599 face images, 5 landmark locations (x and y locations for left and right eyes, a nose and a mouth for each image), and 40 binary attributes annotations.

We first cropped each image from the origin size of 178 x 218 pixels to a new size of 80 x 80 pixels. The images were cropped according to the landmark locations so that the faces are centered and fill the whole pictures. Examples are shown in Fig.1.

Fig.1 Examples of original CelebA images, images cropped to 80 x 80 pixels, and images with masks at the mouth

Then we selected faces with "good quality". Our standards for "good quality" include: 1) the image is neither too dark nor too bright; 2) not a face seen in a profile; 3) not wearing glasses. 84,975 images were selected, some of which are shown in Fig.2.

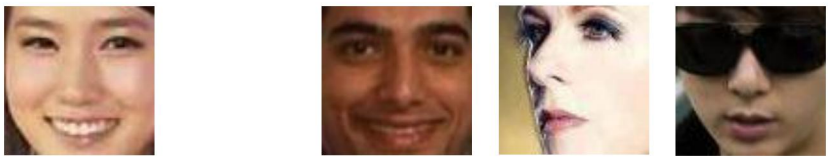

Fig.2 Example training images of (left) “good quality”, and (right) “bad quality” due to unsatisfactory brightness, side profile as well as the existence of glasses

Before feeding the data into our CNN, we rescaled the value of each pixel from integers in [0,255] to float numbers in [-1,1]. Generated images were rescaled back to [0,255] for visualization.

For the image completion task, we masked a 30 x 30 pixel area at the mouth.

3 Models and Methods

3.1 Image Generation

We constructed a DCGAN with structures illustrated in Fig.3. The generator gets a random noise vector of length-100, uniformly distributed in (-1,1), as its input. It is then propagated through two dense layers as well as four up-sampling and convolutional layers, before reaching the output shape of 80 x 80 x 3. tanh activation is used throughout the generator. The discriminator contains four max-pooling and convolutional layers as well as two dense layers, propagating an input image of 80 x 80 x 3 to a single-bit unit, interpreted as the probability that the input is drawn from the training set distribution. The discriminator uses tanh activation for all but the final layer, which uses sigmoid.

Various modifications to this architecture have been tested as suggested by the deep learning community, and are elaborated in the “Discussion” section below. Nevertheless, the network reported previously generated the most satisfying images.

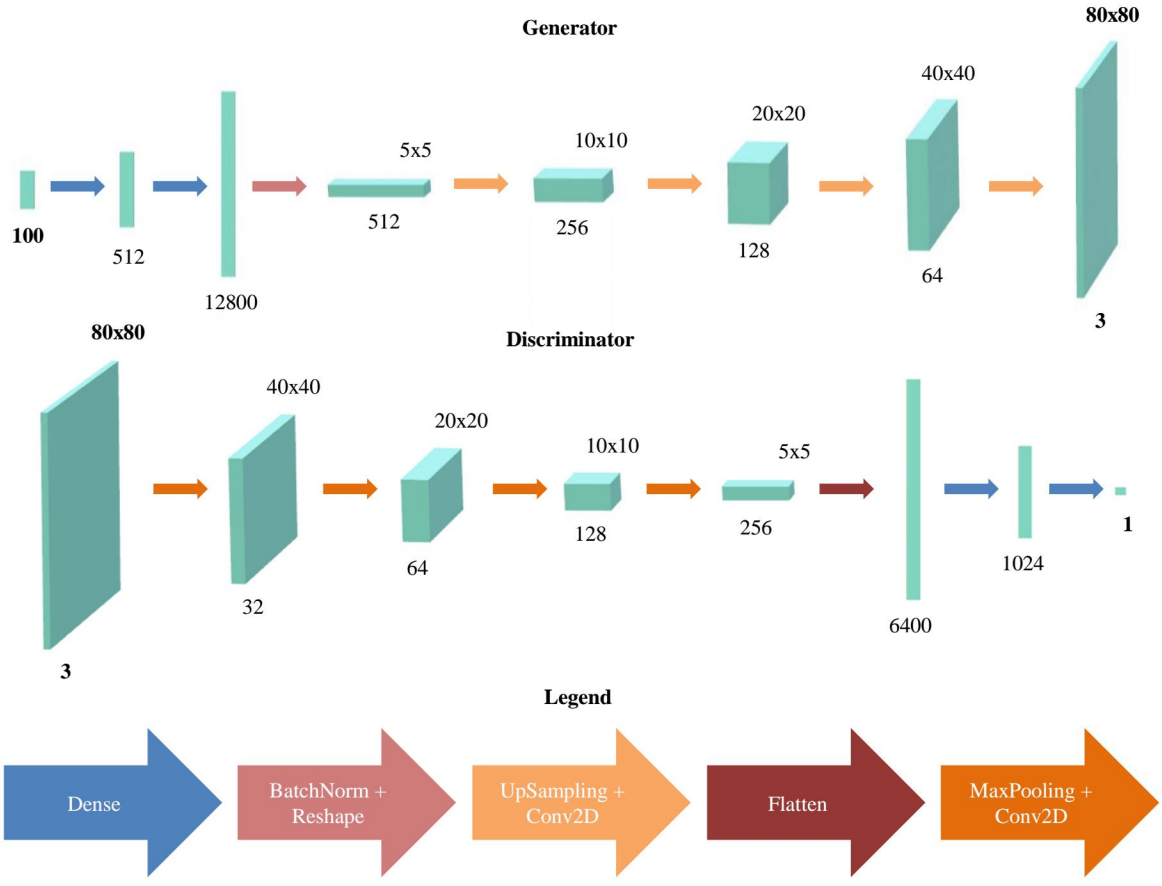

Fig.3 Structure of our GAN, with the legend in the bottom row showing the color-coding of the arrows

3.2 Image Completion

Once the GAN has been trained, we fixed its weights and used it to construct an image completer. Given a target image with missing pixels, the objective of the completer is to generate an image that resembles the target image on the known pixels, and in the meantime looks natural to the discriminator. This is achieved by back-propagating a properly defined loss to the input noise vector of the generator. By tuning the input vector (we have used stochastic gradient descent), the generator is induced to output the best possible replica of the target image, with the missing pixels being inpainted. The inpainted pixels can be regarded as the best guess of the lost information, and are used to complete the target image. This procedure is illustrated in Fig.4.

The loss function consists of two parts shown in Eq.1. The perceptual loss describes how natural the generated image is, and is obtained from the discriminator. The contextual loss measures how the generated image resembles the target, and is calculated as the L1 norm of the pixel-by-pixel RGB difference between the generated image and the target on the known pixels. A hyperparameter lambda weighs the two losses before summing them together. In our implementation, the loss is 99% contextual and only 1% perceptual.

$$L_{contextual}(z) = ||M * G(z) - M * y||_1$$
$$L_{perceptual}(z) = \log(1 - D(G(z)))$$
$$L(z) = L_{contextual}(z) + \lambda L_{perceptual}(z)$$
$$\hat{z} = \arg\min_z L(z)$$

Eq.1 The contextual, perceptual, and total loss for image completion, where M is the mask of missing pixels

In the actual completion process, the previously reported architecture is too large for the noise vector to be updated efficiently. Significant improvement in runtime is achieved by using a smaller GAN with the number of filters reduced by half. All subsequent image completion results come from this smaller completer.

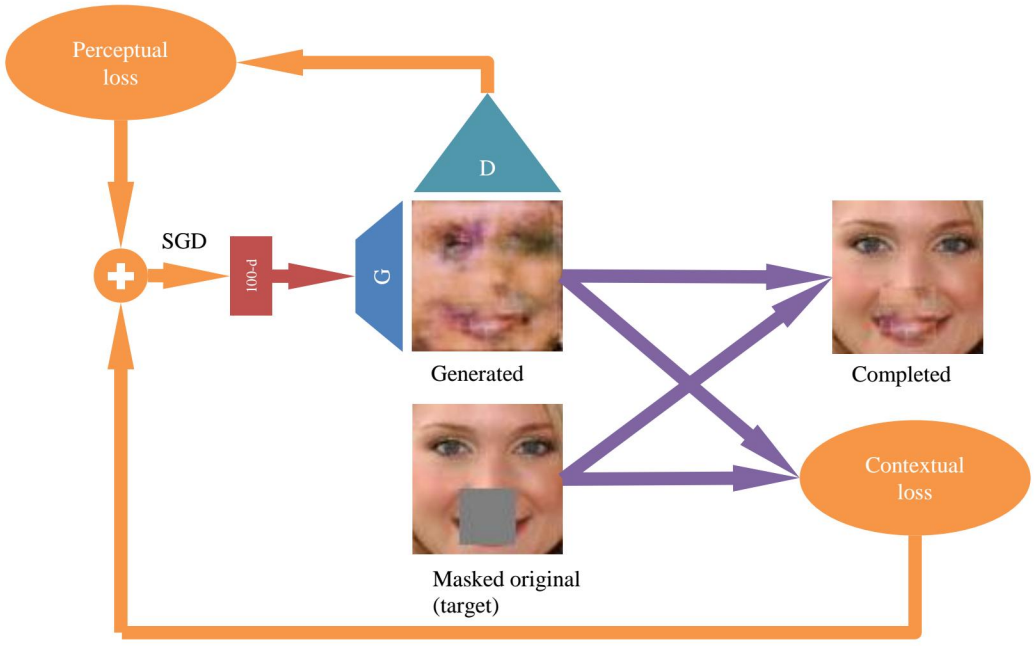

Fig.4 Image completion procedure

4 Experimental results

4.1 Image Generation

During the training of DCGAN, it is very important to keep the generator and the discriminator equally strong. Once this balance is broken, the loss function diverges and the generated images degrade. Besides changing the architecture

of DCGAN, an efficient way to keep this balance is to adjust the training ratio between the generator and the discriminator. For the architecture mentioned above, the discriminator usually learns faster than the generator, making it necessary to train the generator more. However, if the generator gets too much training, the discriminator will fail to function well. This makes the training of the generator ineffective and consequently leads to the failure in generating natural images. After many attempts, we found that the best way of training our DCGAN is to train the generator twice for every training on the discriminator. Examples of generated images after 5 training epochs are shown in Fig.5. The generated images have a moderate quality in details, but the skin color is less even than real faces. Further trainings worsened the quality of generated images.

Fig.5 Examples of generated human faces

Then we visualized some of the outputs of selected layers in the generator and the discriminator in Fig.6. For the generator, the human face begins to take shape in the second convolutional layer, and the details are developed in the third and fourth convolutional layers. For the discriminator, the first convolutional layer extracts the outlines and edges of the face and its details. The rest three convolutional layers have broader views so the details are harder to find in the filtered images.

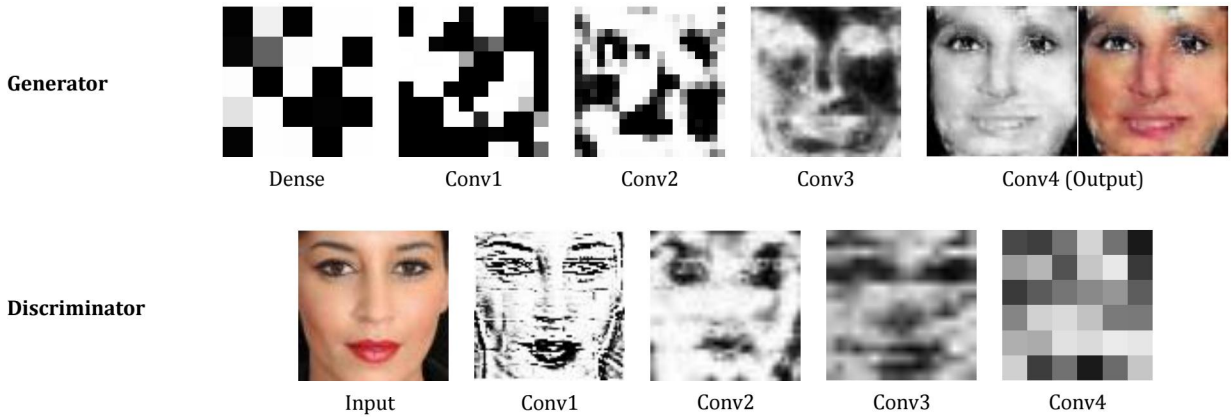

Fig.6 Outputs of selected hidden layers in the generator (top) and the discriminator (bottom)

4.2 Image Completion

With a random input noise, the first image generated is usually completely different from the target image with a large loss. To accelerate the completion process, we randomly created 1000 input noises at the beginning. Then we only selected the one with the smallest loss for further updates through SGD. This saves about 25% of runtime. The generated image becomes closer to the target image with more iterations. This trend can be seen in Fig.7.

Fig.7 Generator output after back propagating on the 100-dimensional noise vector for 0, 10, 30, 50, and 100 thousand iterations (from left to right), with the target image (mouth masked) on the far right for reference

The corresponding part of the generated image was used to fill in the missing part in the target image. Although the

DCGAN is unable to fill in very detailed features of the face (completed pixels are usually blurry), the completion is perceptually correct and moderately consistent with the target. More examples of image completion are shown in Fig.8.

Fig.8 Examples of completed images, with 30 x 30 pixels at the mouth coming from the generator

5 Discussion

GANs are very sensitive to the architecture and hypermeters. We explored several architectures and many tricks suggested by the DCGAN paper [4] and the deep learning community. For example, it is suggested in the DCGAN paper that max-pooling layers should be substituted by strided convolutions, and that fully connected layers on top of convolutional features should be eliminated. However, they all hurt the performance in our case. In addition, Batch Normalization is generally recommended to stabilize the training. But we found that only one Batch Normalization layer on top of the first convolutional layer in the generator is good for the DCGAN, while more Batch Normalization layers in the generator or the discriminator do not improve or even worsen the quality of the generated images. In addition, we tried ReLU activation and the leaky ReLU activation for the hidden layers, but neither of them works better than tanh. As for hypermeters, we found that tuning the momentum beta to 0.5 stabilizes the training, in agreement with the DCGAN paper.

GANs are also notoriously hard to train because of its intrinsic instability. As is mentioned before, the unbalanced strength between the generator and the discriminator leads to divergence. Even with an appropriate training ratio, we achieved our best model after 5 epochs of training, but it worsened with further trainings. It is probably necessary to tune the training ratio during the middle of the training, which makes the training process harder and more time-consuming. Wasserstein GAN and residue networks might improve the stability and performance.

The image completion process takes far more time than we expected. Therefore, we used a smaller DCGAN to complete images, as mentioned before. We might be able to complete the image with more details if we use out best trained DCGAN. In addition, better optimizers like Adam may facilitate the completion process.

6 Conclusion

In this project, we trained a DCGAN to generate natural human faces. Then we used the trained GAN to generate new images that resembled the known parts of target images, in order to complete the missing pixels. Faces with moderate details were generated from our best GAN. A smaller GAN was able to reasonably complete images of human faces in a moderate amount of time. The instability of the GAN may be improved by using Wasserstein GAN and residue networks.

7 Contributions

Yijun Jiang: GAN architecture setup, image completer implementation

Yi Liu: image pre-processing, GAN architecture iteration, GAN training

Zhengli Wang: literature review, GAN training

References

[1] I. Goodfellow *et. al*, "*Generative Adversarial Networks*", arXiv:1406.2661 (2014)

[2] Y. Li *et. al*, "*Generative Face Completion*", arXiv:1704.05838 (2017)

[3] Z. Liu *et. al*, CelebA dataset "*Deep Learning Face Attributes in the Wild*", arXiv:1411.7766 (2015)

[4] A. Radford, L. Metz, and S. Chintala, "*Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*", arXiv:1511.06434 (2016)

[5] B. Amos, GitHub blog "*Image Completion with Deep Learning in TensorFlow*", http://bamos.github.io/2016/08/09/deep-completion/

[6] DCGAN implementation in Keras, https://github.com/jacobgil/keras-dcgan