Image Editing with Invertible Conditional GANs

Boyao Sun
Department of Civil and Environmental Engineering
Stanford University
boysun@stanford.edu

Wensheng Li
Department of Civil and Environmental Engineering
Stanford University
we077@stanford.edu

Xinyu Xu
Department of Chemical Engineering
Stanford University
xinyu17@stanford.edu

Abstract

Generative Adversarial Networks (GANs) has led to promising results in image generation in the last few years. Conditional GANs incorporates an external conditional representation that determines some characteristics of images created by generator. Our project focuses on one extension of cGANs, Invertible Conditional GANs (IcGAN). IcGAN has motivated image editing by first modifying the conditional representation and then reconstructing images. In this paper, we implement IcGAN on MNIST[1] and celebA[2] datasets. Although the experimental results on CelebA do not reach our expectation, the results of MNIST have reached our expectation and illustrated the capacity of IcGAN in image editing.

1 Introduction

Image attributes editing can be a tough task. For example, changing person's hair color and facial expression need to take the content of face into account in order to output a natural and satisfactory image. This task might be finished by human but with image editing software offered and a certain amount of time taken. In this project, we are interested in employing generative models in deep learning area to solve this problem. Generative Adversarial Networks (GANs) has led to promising results in image generation in the last few years[3]. Its extension, conditional GAN (cGAN), incorporates informational constraints in creating images so that images will be generated based on conditional information y[4]. Additionally, Perarnau *et al.* propose invertible conditional GAN (IcGAN). The extension is that IcGAN is equipped with two encoders to inversely map from input images into latent information z and conditional information y, which, as a result, can be manipulated to generate output image with specific attributes[5]. The generator part is explained in Figure 1.

Even though Perarnau's *et al.* IcGAN has yielded some satisfactory results in image editing, we find that some generated images are still unnatural. As a result, we would like to re-implement IcGANs and perform hyperparameters tuning and architecture modification to better the performance of IcGANs.

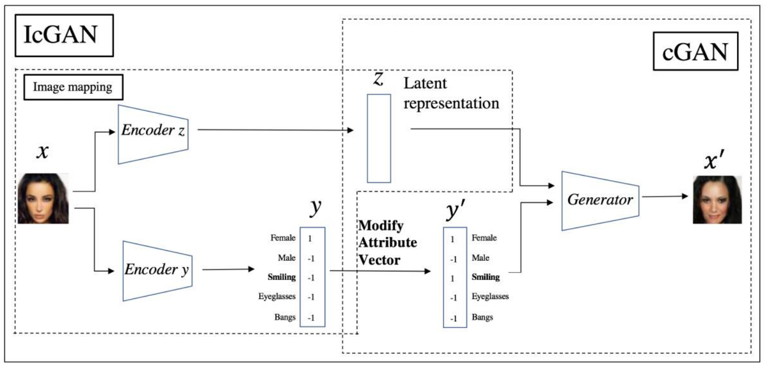


Figure 1: Mechanism of Generator

2 Problem statement

The work of this project is based on the Tensorflow implementation of IcGAN [1]. However, we find that the existing code differs from the original paper[6] in architecture and only works on MNIST dataset, which is gray-scale. Our baseline will be this code. Therefore, we will firstly modify the existing code to have architecture consistent with the paper[6] and then examine the performance of IcGAN on MNIST dataset. Second, we modify the architecture of the existing code and train the model on a colored face dataset. Perarnau *et al.* also published their codes, but the codes are implemented on Torch and we aim to outperform their implementation.

3 Related Work

GANs, proposed by Goodfellow*et al.*, is an unsupervised learning approach that creates a minimax game between generator and discriminator, which has managed to learn and approximate data distributions, such as images[3]. Its performance has been further leveraged by a deeper architecture (DCGAN)[4]. Even though DCGAN has strength of reconstructing realistic images, it lacks control of generating data distributions based upon constrained information. cGAN, by Radford*et al.*, adds additional information to manipulate the direction of data generation process [5]. The additional information can be informal text descriptions. However, if we would like to modify some attributes of an existing image, such as hair color of person, text descriptions as constrained information are insufficient. Instead, we have to obtain image latent attribute representation as our conditional information in cGAN. IcGAN, as a state-of-art approach by Perarnau *et al.*, proposes such a encoder, which maps from an image to latent representation z and attribute representation y, so that we are in control of some attributes of generated images [6]. As an image can be generated based upon y changed, image editing can be achieved. In addition, a combination of GAN and VAE by Larsen *et al.* [7] shares some ideas with IcGAN [6]. For example, Larsen *et al.* also integrates an encoder mapping from data into latent space z with GANs. The main difference between the two encoders is loss function. Larsen's *et al.* model adds feature-wise reconstruction loss to VAE encoder, which is an output from a hidden layer of discriminator. This improvement might better the performance of IcGAN and can be part of our future work.

4 Dataset and Features

The datasets consist of MNIST dataset[1] and CelebFaces Attributes (CelebA) dataset[2].

MNIST has 70000 images, which have been divided into 60000/5000/5000 as train/dev./test data. Each sample is a gray-scale $28 \times 28 \times 1$ image labeled with the class of digits from 0 to 9. The feature of each image is a label of an one-hot vector which represents the corresponding digit. Also, we normalize each channel of the input in MNIST into [0,1]. The output is a $28 \times 28 \times 1$ image.

[1]https://github.com/zhangqianhui/ICGan-tensorflow

CelebA has 202599 images, which have been divided into 97%/1.5%/1.5% as train/dev./test data. Each sample is a colored $178 \times 218 \times 3$ face image labeled with 40 binary attributes. Also, we crop a $108 \times 108 \times$ image at center of each CelebA sample, in order to eliminate the effects of background in image generation. Then, we resize each cropped image into 64×64. The feature for a CelebA image is a label with a ground-truth attribute vector of length of 40 provided by the work[2]. Each element of the vector is labeled with 1 if the attribute exists and is labeled with -1 if not. Also, each channel of the input image is normalized into [-1,1]. The output is a $64 \times 64 \times 3$ image with RGB channels. We also create a small dataset (8192 images) for facilitating our experiments.

The baseline code of this project can only deal with 28×28 gray-scale image, so we have modified the model to make it work on multiple-channel images.

5 Methods

5.1 Architecture

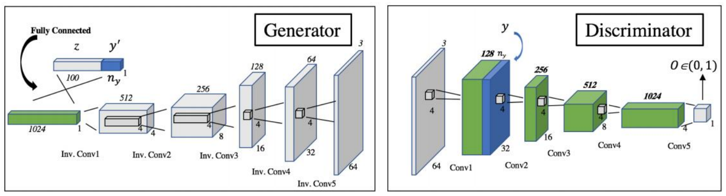


Figure 2: Architecture of Generator and Discriminator (our modification is marked in green)

Our architecture is built based upon Perarnau's model *et al.*[6] and the baseline model, as shown in 2. We firstly change the baseline model to accept a 3-channel input and output 3-channel image for the purpose of face image editing. In generator, we concatenate the latent space with attribute vector (ground-truth label in training and modified label in testing). After one fully connected transformation and five deconvolutional transformations, we obtain a fake image. In discriminator, an image is transformed with five convlutional layers to yield $1 \times 1 \times 1$ output, representing if discriminator thinks the input image as real or fake. Note that attribute vector has been inserted into one hidden layer. The details are shown in 2.

Additionally, we follow some techniques used in the works[4, 6]. First, we use batch normalization in both generator and discriminator, except the first layer of discriminator and the last layer of both. For generator, we use ReLU activations in the deconvolution layers except the last layer. Tanh activation is employed in the last layer of the generator. On the other hand, for discriminator, Leaky ReLU is used in the convolution layers of except the last layer. At last, Sigmoid is used in the last layer.

For the models of encoder z and y, we follow exactly what Perarnau *et al.* used in the work[6].

5.2 Loss Functions

This model is to generate new images by minimize the generator and maximize the discriminator respectively. θ_g and θ_d represent the parameters of G and D, equation1 shows the function which should be optimized. For the generator, we use the non-saturating cost learned in class. And the cross entropy function has been used for the discriminator.

$$\begin{aligned} min_g max_d v(\theta_g, \theta_d) = \mathbb{E}_{x,y \sim p_{data}}[log D(x, y)] \\ + \mathbb{E}_{z \sim p_z, y' \sim p_y}[log(1 - D(G(z, y'), y'))] \end{aligned} \tag{1}$$

In the IcGAN paper[6], different loss functions have been used to train the encoder E_z and E_y independently. The equations are shown below. For E_z, z is corresponding to the normal distribution, and we want to minimize the difference between latent representation and normal distribution to regularize the model. For E_y, we want to minimize the difference between the ground truth labels and the encoder results.

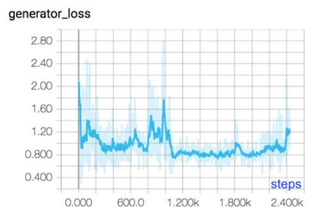
$$L_{ez} = \mathbb{E}_{z \sim p_z, y' \sim p_y} \| z - E_z(G(z, y')) \|_2^2 \tag{2}$$

$$L_{ey} = \mathbb{E}_{x,y \sim p_{data}} \| y - E_y(x) \|_2^2 \tag{3}$$

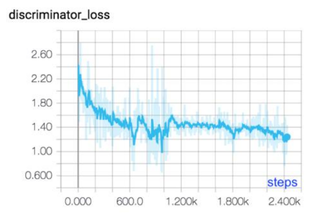
5.3 Approach

In order to build a stable IcGAN model, we follow the guidelines in Radford's paper[4] and in Perarnau's paper[6]. The model works well on MNIST, but the quality of the output images are unsatisfactory. We conjecture that it is because we are unaware of some details that are not elaborated in the papers[4,6]. Therefore, we modify the model based on the output images and losses. We will evaluate our attempts by explaining the effects of each modification.

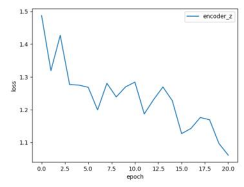
- **Input Image Normalization:** Our first attempt does not apply input image normalization. The discriminator can easily tell the difference between real images and fake images (D loss converge to 0), and the generator loss never converges. Because we use Tanh in the last layer of the generator, the fake images are in [-1,1], but the real images are in [0,255]. Therefore, we normalize the input images to [-1,1].
- **Learning Rate Decay:** During the training process, we find the D loss converges quickly and becomes plateau-shape, but the output image quality stops improving after D loss converges. We attempt step learning rate decay. The D loss decreases a little and quickly converges again after learning rate decay is applied. So, the effect is not significant.
- **Balance loss via modifying architecture** We modify the architecture of both generator and discriminator to balance the loss. We increase the number of filters in the convolution layers of the discriminator, and we add a fully-connected layer in the generator before all the deconvolution layers.
- **Dropout Regularization** After we modify the architecture, the model works well at first, but the discriminator will start overfitting in the last few epochs. The D loss suddenly drops to 0, and the generator starts to generate random pixels. Therefore, we add dropout in the discriminator to alleviate the overfitting problem.



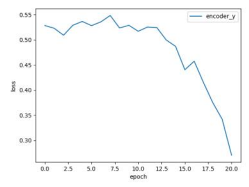
(a) Generator Loss



(b) Discriminator Loss

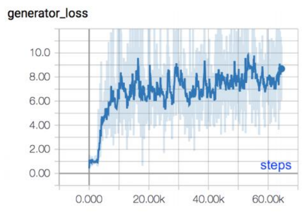


(c) training loss of encoder z

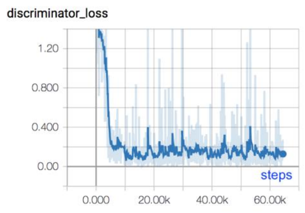


(d) training loss of encoder y

Figure 3: Loss Curve of training on the small dataset



(a) Generator Loss



(b) Discriminator Loss

Figure 4: Loss Curve of training on the entire dataset

6 Experiments & Results & Discussion

6.1 Training Process

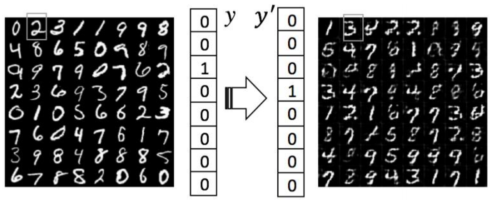
In our experiment setting, both generator and discriminator use 64 as batch size and 0.0002 as learning rate, which are consistent with the work[6]. We focus our discussion on CelebA, because the implementation on MNIST has converged well and has satisfactory results. The results are shown in Figure 5a. We first train our model on a small dataset with size of 8192. As observed in Figure 3, Discriminator loss decreases during training while Generator loss does not have significant reduction. These loss curves meet our expectation because in minimax game, discriminator becomes powerful and hence generator loss cannot decrease. After the losses converge on the small dataset, we run the model on the entire train set which has 196608 images.

In addition, we train encoder z and encoder y. The training losses are converging as epoch increases, shown in Figure 3 part c and d. This observation indicates that our encoders can perform well. The loss curves also perform well on the entire dataset and will not be repeated here.

6.2 Results and Evaluation Metrics

Figure 5a shows IcGAN results of MNIST dataset. The image on the left is the real MNIST digits and is added with one to reconstruct the new image on the right. It is shown that most of test examples have been generated with the same style. To evaluate our model, we randomly select a test batch and count how many digits over total batch size are visually clear and correctly edited by human perception. The accuracy is 74.60%.

Figure 5b shows an example of edited CelebA image. We remove the glass of the person on the left to reconstruct the new image on the right. We first observe loss curves shown in Figure 4. The generator loss has a sharp increase at around 5k steps (3 epoch) while the discriminator decreases simultaneously. This is probably because we have strengthened discriminator too much. However, visual results on dev. set are much better than those on the small datasets. For example, human faces are not as blurry as before and some attributes can be edited. The better visual results indicate that training on a large dataset help improve the performance of the IcGAN model.



(a) Results on MNIST datasets

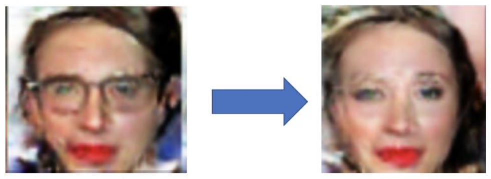

(b) The removal of glass on CelebA

Figure 5: Results of MNIST and CelebA

7 Conclusion & Future Work

In this project, we have implemented IcGAN on MNIST dataset, and the results indicate that 74.6% digit images have been edited as expected. These results illustrate the capacity of IcGAN in image editing. We also modify and then implement IcGAN on CelebA dataset. We firstly have our modified model loss converged on a small dataset of CelebA referring to the loss curves. Then, we train our model on the whole CelebA dataset. Even though loss curves fail to converge on the whole dataset (discriminator loss comes to zero at about 3 epoch), the visual results of image editing are satisfactory. We conclude that our model does not outperform the model by Perarnau's *et al.* due to the loss curves. However, as running one epoch takes 17 hours, we do not manage to revise our model and run the whole dataset due to constrained time.

In the future, we would like to modify our model and run on the whole dataset again. For example, we could use a less powerful discriminator to prevent its loss from reducing quickly. Also, as mentioned before, we could utilize the combination of VAE and GAN to leverage the performance of IcGAN, as more high-level features extracted in discriminator will be used to train generator.

8 Contributions

- Boyao Sun: IcGAN implementation and modification on CelebA dataset, data preprocessing.
- Wensheng Li: IcGAN implementation on MNIST dataset, modification on CelebA dataset, data preprocessing.
- Xinyu Xu: IcGAN modification on MNIST dataset, modification on CelebA dataset, data preprocessing.

References

[1] LeCun, Y., Cortes, C., & Burges, C. J. (2010). MNIST handwritten digit database. *AT&T Labs [Online]*. Available: http://yann. lecun. com/exdb/mnist, 2.

[2] Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. *In Proceedings of the IEEE International Conference on Computer Vision* (pp. 3730-3738).

[3] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., & Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

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

[5] Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[6] Perarnau, G., van de Weijer, J., Raducanu, B., & Álvarez, J. M. (2016). Invertible conditional gans for image editing. *preprint arXiv:1611.06355*.

[7] Larsen, A. B. L., Sønderby, S. K., Larochelle, H., & Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.