Article

dbzGAN: Conditional Deep Convolutional Generative Adversarial Networks for Animation Character Generation

Doosol Han[1]

[1] Graduate school of information, Yonsei University; fjordy@yonsei.ac.kr

Abstract: Generative models have shown its' capability of producing high quality images. In particular, Deep convolutional generative adversarial networks (DCGAN) showed a better performance compare to the original GAN model as it exploits convolutional architecture which is well known to be a good fit for images. In this paper, we present conditional DCGAN for generating images of desired target class. The result showed that our model is capable of producing distinct images according to their classes.

Keywords: Generative Adversarial Networks, DCGAN, Conditional GAN

---

1. Introduction

The task of image generation has drawn attention from researchers ever since Ian Goodfellow introduced Generative Adversarial Nets(Goodfellow et al., 2014). The idea of training two networks via an adversarial process allowed generator to successfully learn the distribution of input images. Followed by his work, DCGAN was introduced in 2016 which improved both the quality of generated images and stability of training process.

Despite the success of DCGAN, we have noticed some problems when we applied our own dataset. In real world, unlike images such as Cifar-10, LSUN that were used to train DCGAN, the images we use as inputs are not always guaranteed to have a fixed size, little variation across images, and similar background. Moreover, the trained generator of DCGAN is only able to produce random images within the distribution of original images.

In this paper, we propose dbzGAN, which uses images of characters from Japanese animation Dragon Ball to generate different forms of characters and explicitly control the form class to be generated by feeding conditional information to both generator and discriminator. In addition, we show that our model successfully handles the noisy image dataset.

2. Related Work

Generative Adversarial Networks. Generative adversarial networks have shown promising results in various computer vision tasks. The original GAN model has two modules: discriminator and generator. The discriminator learns to distinguish fake images from real images, while generator learns to generate fake images that are indistinguishable by discriminator.

Deep Convolutional GANs. DCGANs used CNN architecture on both discriminator and generator networks. The discriminator network uses regular 2d convolutional layers to down sample the input image and the generator network uses transposed 2d convolutional layers to up sample the input latent vector. They used strided convolution instead of max-pool and the generator uses transposed convolution layer for final output with tanh activation.

3. Dragonball forms Dataset

Figure 1. Transformation stages

Dragonball is a famous Japanese animation. The characters of Dragonball has the ability to change their bodies in order to tap into greater stores of power, which is called transformations. The stages of transformation is known as Super Saiyan, and it comes with 5 stages as shown in Figure 1: Super Saiyan 1(SS1), Super Super Saiyan 2(SS2), Super Saiyan 3(SS3), Super Saiyan 4(SS4), Super Saiyan Blue(SSB). The SS1 is the first stage and the only characteristics is that it has yellow hair. The SS2 form is where the hair becomes spikier and covered with electric effects. The SS3 form has very long hair and it has no eyebrows. The SS4 form has black hair, red furs growing on body, and with red tail. The final form, SSB, is similar to the first stage form, but has a blue hair color.

We used chrome-driver and google image scraper python module to scrape the images. Since the scraped results had lots of unwanted images such as fanarts, screenshots of a related games, and cosplay pictures, we manually selected images that are in reasonable quality. The resulting dataset consisted of 2250 images, with 450 images per transformation. The size of the images were all differed in scale therefore we had to resize the images into the shape of 64x64.

4. Methods

4.1 Optimization objective

Our goal is to train both generator and discriminator using adversarial loss. To conditionally constrain the variation of generated samples, we apply cGAN(Mirza and Osindero, 2014)'s loss function. The only difference from the original GANs optimizing function is that we now provide conditional information (Transformation label) to both discriminator and generator. The resulting formulation is

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

, where constrained variations are modeled with y as it directly correlates with features of the data that are explicitly correlated with y and the data. Other variations are encoded in z, the latent vector.

4.2 Implementation

Generator takes latent vector (dim=100), discriminator takes both real and fake image (64x64x3), and both take conditional label y (dim=1,) . The conditional label is encoded using embedding layer and resulting embedded vector gets concatenated as additional channel to the image. Generator network is composed of five transposed convolutional layers with the stride size of two for downsampling. We use LeakyReLU activation for first four layers, tanh activation for last layer as well as batch normalization. The discriminator network is composed of four convolutional layers

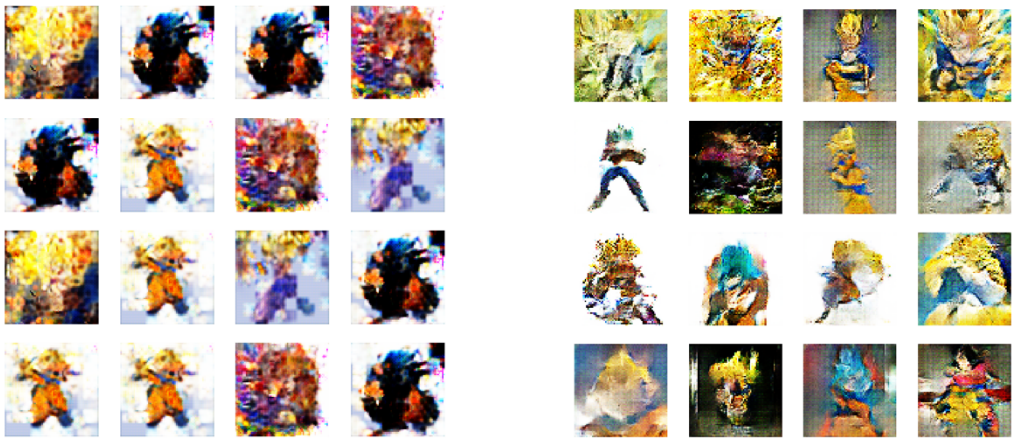


Figure 2. Generated images of Baseline Model(Left) and our Model(Right) at 4000 epoch

with the stride size of two for upsampling, and one dense layer for final output. We use LeakyReLU activation for all layers and linear activation for final layer. The Dropout layers and Batch normalization layer were used except for the last layer. We also added white noise to the input image of discriminator network to ensure the stable training process. The detailed architecture is illustrated in the appendix.

4.3 Training

All models are trained using Adam optimizer with alpha = 0.0002, $\beta 1 = 0.5$ and $\beta 2 = 0.999$. The batch size is set to 64 and trained for 4000 epochs. Training takes about 4 hours on a Google Colab environment, with NVIDIA Tesla K80 GPU.

4.4 Baseline Model

As our baseline model, we adopt original DCGAN with authors' recommended configuration settings: ReLU activation for generator, no dropout layers for all networks, no batch normalization on generator's input layer. As for conditional attributes, we applied the same logic to the baseline model for the sake of comparing conditional outputs.

Since the goal of our study is to build a model that can subjectively distinguish generated samples' conditional difference, we qualitatively compare the results with generated images of each model.

5. Results

5.1 Generated Images from random latent vector

We randomly created 16 samples of latent vectors with randomly assigned conditional labels before training. These vectors act as test dataset by testing trained generator to generate conditioned images for every 100 epochs. As shown in Figure 2, baseline model generated distorted images with mode collapses as there are only 5 variations among 16 images. Meanwhile, our model produced more clear image, while still distorted, with many variations among the images.

5.2 Conditionally Generated Images

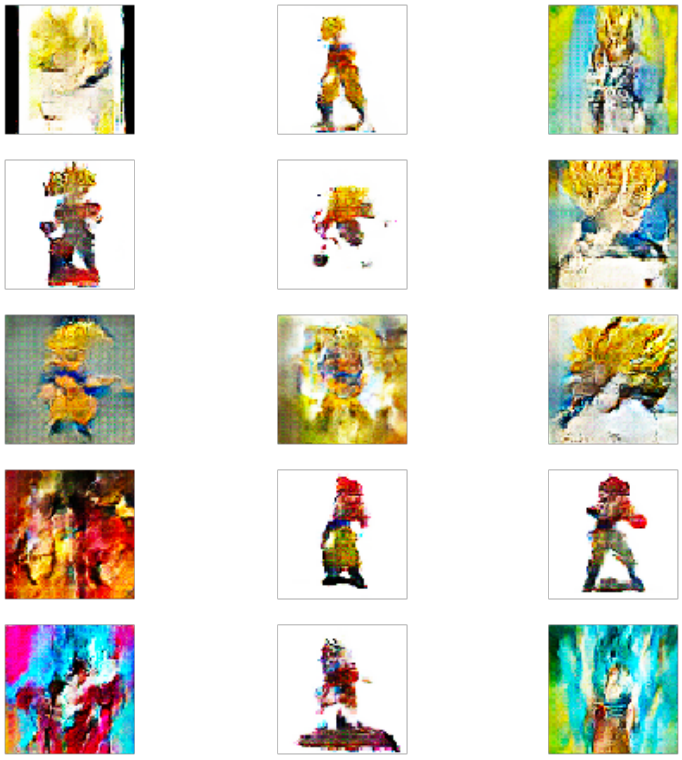


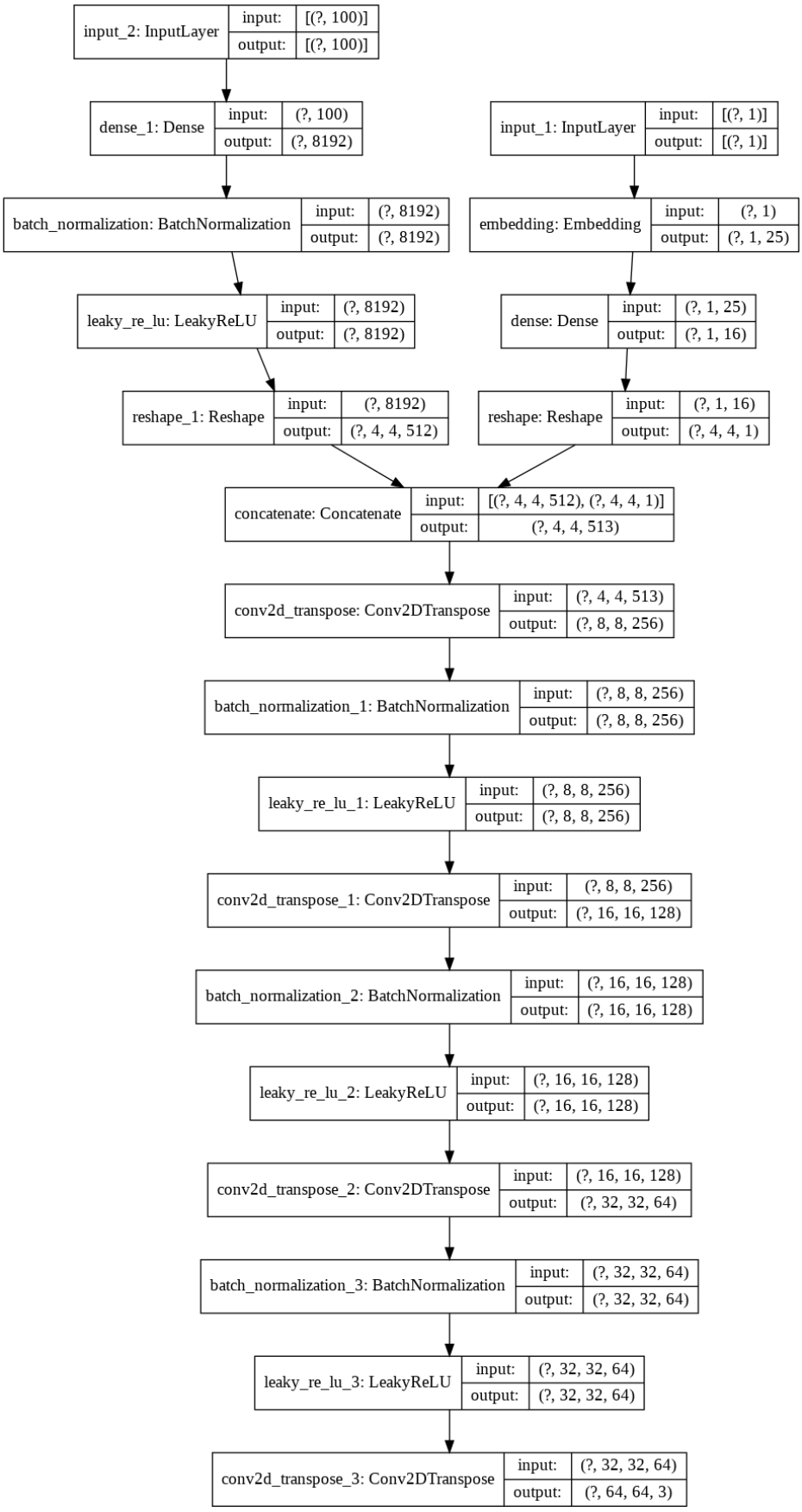
Figure 3. Sample generated images of each conditioned form

We provided trained generator with conditions(transformations) to generate 3 samples of each transformations. Figure 3 shows generated samples(3 samples of each class per row, from ss1 to ssb). As we can see from the figure, the quality of generated images were not as promising as we initially thought. However, we could see that our generator was successfully trained to generate each class image with their characteristics. Third row images are conditioned to generate SS3 form, which has characteristics of long yellow hair without eyebrows. Although the images are blurry therefore not able to see if there is eyebrows or not, the model successfully captured the long hair characteristics of the SS3 form. The generator also captured the characteristics of the SS4 and SSB forms shown at the fourth and fifth row: red(black) hair, red fur, yellow pants for SS4 ; blue hair with blue aura for SSB. However, the model was not able to distinguish the characteristic differences between SS1 form and SS2 form. Lastly, we can see that generated images are reasonably accurate in terms of showing the overall posture and body shape of characters.

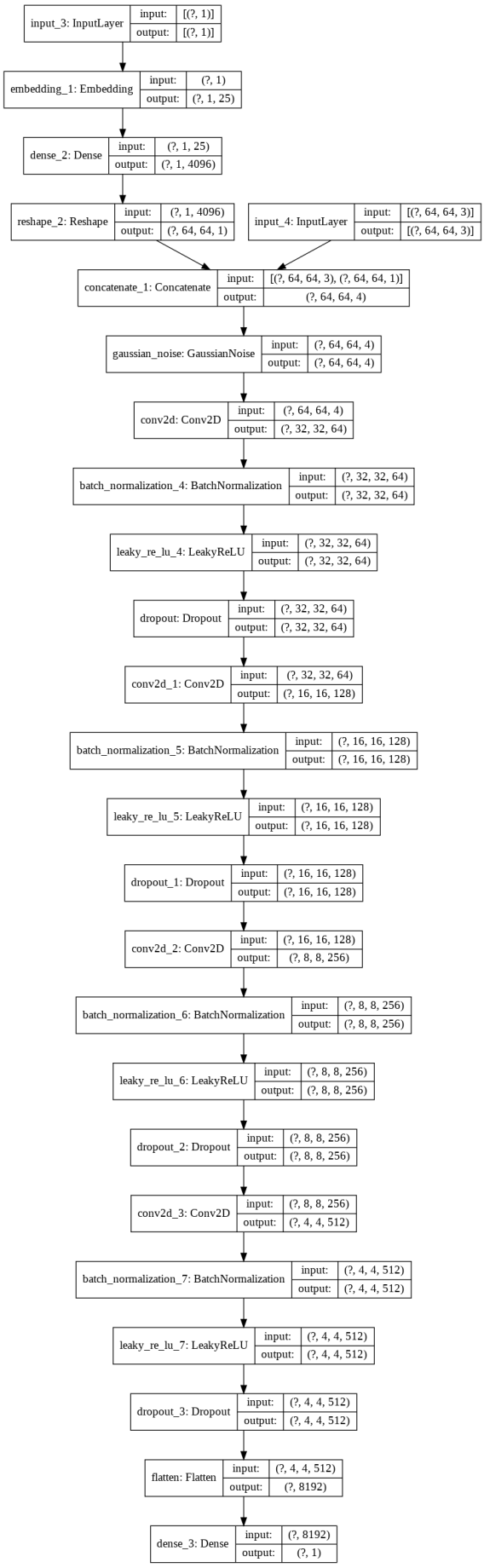
6. Conclusion

In this paper, we present conditional DCGAN for generating images of desired target class. The result showed that our model is capable of producing distinct images according to their classes. Fact that we used web-scraped image files with different postures, different size, and variety of background noises, and still be able to produce reasonable quality images shows that our model can generalize to different domain images with custom, noisy dataset. With more advanced preprocessing techniques such as cropping images with image detection algorithm to reduce variability in image distribution, we expect our model to produce better quality images.

Appendix (Network Architecture)



Appendix 1. Generator Architecture



Appendix 2. Discriminator Architecture

References

1. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *3*, 2672–2680.
2. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.* **2016**, 1–16.
3. Perarnau, G.; van de Weijer, J.; Raducanu, B.; Álvarez, J.M. Invertible Conditional GANs for image editing. **2016**, 1–9.