Generation of Real Looking Images Using GANs

BOHNSTEDT Timo
Friedrich Alexander University
timo.bohnstedt@fau.de

***Abstract*—Automatic image generation is a complex task with many applications in several domains such as security (e.g., generating portraits from the description), styling and entertainment. In this project, advanced versions of Generative Adversarial Networks (GANs) are used to generate real images. A conditional GAN (cGAN) was implemented, followed by an evaluation with visual examination, k-nearest-Neighbours (kNN) and Fréchet Inception Distance (FID). The evaluation indicates that a cGAN can generate realistic images of handwritten digits. Whereas, the discussions shows that more work must be done to create realistic images from datasets which contains larger and more complex images.**

***Index Terms*—Machine Learning in the Industry 4.0, EDA, Data Analytics, FAU, Python, GANs, CGANs**

1. Introduction

Generative Adversarial Networks (GANs) have been getting a lot of attention recently. They are used for different tasks, such as image to image translation where images can be transferred from one domain to another, text to photo-realistic image synthesis and to increase the resolution of low-resolution images. GANs were introduced as an unsupervised learning model, trained on data without labels. The model learns to take noise as input and output data that resembles examples from the training set.

There exist several challenges to model distributions in a stable and discriminative way. The two main issues are a slow training process and the so-called mode collapse.

Most real-world data distributions consist of different modes. Technically speaking, they are multimodal. For example, in the MNIST dataset, there are ten modes from digit zero to digit nine. Assume that there are two different GANs which sample from the MNIST dataset. The first model produces all ten modes while the second model creates a single mode only, for example, the digit six. This problem is called mode collapse when only a few modes of data are generated. That means training will fail because the model is not able to introduce enough inter-class variance. Slow training appears if the gradient to train the generator vanishes. That means as well that the training fails because there will be no result in feasible time.

There are versions of GANs that can use additional information as input, such as conditional GANs (CGANs). They are mentioned in the paper Image-to-Image Translation with Conditional Adversarial Networks and in the paper Improved Techniques for Training GANs [4] [8].

While hearing from the opportunities offered by CGANs and reading through the related work, a question of interested is:

> *Is it possible to generate real-looking images, and is it possible to introduce enough inter-class variance for training?*

This paper focuses on answering both questions. The paper gets submitted in partial fulfilment of the requirements for the seminar machine learning in industry 4.0 at the Friedrich Alexander University Erlangen Nürnberg.

2. Methods

In principal GANs are a framework for estimating generative models via an adversarial process, in which two models get trained simultaneously. A generative model $\boldsymbol{G}$ that captures the data distribution and a discriminative model $\boldsymbol{D}$ that estimates the probability that a sample came from the training data rather than from $\boldsymbol{G}$ [1]. Simplified the generator tries to produce data that came from some probability distribution. For example, $\boldsymbol{D}$ is a child who tries to impress his parents with a realistic image. The parents are the equivalent for $\boldsymbol{D}$ in this metaphor, and they are supervising the child to ensure it

makes progress. Let's assume the child is drawing a cat as in the top left of Figure 2. First, the parents can easily distinguish whether it is a cat or not, because the image is looking just like a scatch. But after a few attempts and suggestions from the parents, the child will be able to draw an image which is hard to separate from an actual cat image.

Mathematically this problem can be described in solving a two-player min-max game. The problem is divided into three parts:

1) Maximize the chance to recognize real images as real
2) Maximize the chance to recognize fake images as fake
3) optimize G such that it fools D the most.

The first two steps can be summarized with the following equation where $D(x)$ outputs a value indicating the chance that x is a real image [2]:

$$\begin{aligned} \max_D V(D) = \mathbb{E}_{x\sim p_{data}(x)}[log\, D(x)] \\ + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))] \end{aligned} \tag{1}$$

Note that the first addend is to recognize real images better and the second addend is to recognize fake images better. So, the objective function of G can be described as follows:

$$\min_G V(G) = \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))] \tag{2}$$

By using gradient decent as an optimization technique and using the chain rule to get the derivates of the objective functions one step of particular training involves the following steps [2]:

- sample a mini batch of m noise vectors $\{z^{(1)}, \ldots, z^{(m)}\}$
- sample a mini batch of m m training samples $\{x^{(1)}, \ldots, x^{(m)}\}$
- Update D by doing one gradient descent step on its loss function:

$$\begin{aligned} J_D = -\frac{1}{m}\sum_{i=1}^{m}\Big[log\, D(x^{(i)}) \\ + log(1 - D(G(z^{(i)})))\Big] \end{aligned} \tag{3}$$

- Update G by doing one gradient descent step on its loss function:

$$J_G = -\frac{1}{m}\sum_{i=1}^{m}\Big[log\, D(G(z^{(i)}))\Big] \tag{4}$$

The cGAN can be constructed by merely feeding the labels y into both the generator and discriminator. Transferred to the metaphor which was given above, the labels are more specific instructions such that the child can improve his drawing skills [3].

Mathematically, the only thing that changes is that the generator G and the discriminator D now condition on additional information y. So the steps of training are now as follows [4]:

- sample a mini batch of m noise vectors $\{z^{(1)}, \ldots, z^{(m)}\}$
- sample a mini batch of m m training samples $\{x^{(1)}, \ldots, x^{(m)}\}$
- Update D by doing one gradient descent step on its loss function:

$$\begin{aligned} J_D = -\frac{1}{m}\sum_{i=1}^{m}\Big[log\, D(x^{(i)}) \\ + log(1 - D(G(z^{(i)})))\Big] \end{aligned} \tag{5}$$

- Update G by doing one gradient descent step on its loss function:

$$J_G = -\frac{1}{m}\sum_{i=1}^{m}\Big[log\, D(G(z^{(i)}))\Big] \tag{6}$$

The prototype shall prove whether the model can generate MNIST digits conditioned on class labels or not. MNIST requires significantly fewer computational resources as other datasets, while still providing useful feedback on the cGAN architecture. The MNIST dataset contains 60000 images of handwritten digits, as can be seen in Figure 1.

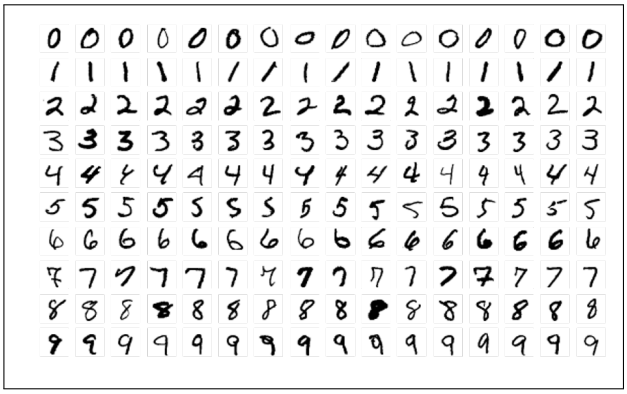

Fig. 1. A few examples of the so-called MNIST dataset which contains 60000 images of handwritten digits.

Figure 3 reveals how the conditional deep convolutional neural network architecture works in practice for generating handwritten digits.

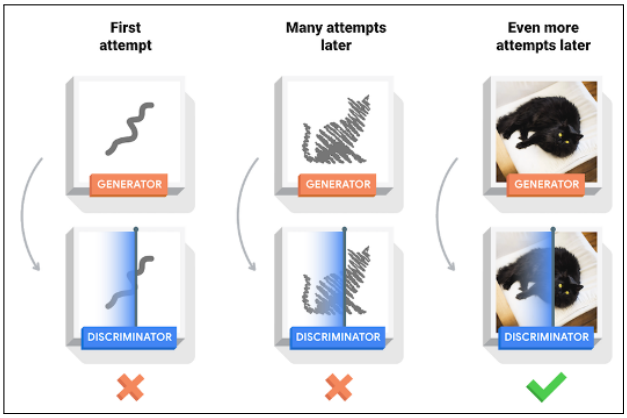

Fig. 2. Two models are trained simultaneously by an adversarial process. A generator ("the artist") learns to create images that look real, while a discriminator ("the art critic") learns to tell real images apart from fakes [5].

First, the noise and the labels passed to G. The different layers of the network perform dense and reshape operations

before the vectors are concatenated. Finally, G produces images at the same size as the original images. Afterwards, a real image and the generated image get passed to D, which uses dense and reshape operations as well. Finally, he decides whether or not this is a real or a generated image, as can be observed on the bottom right in Figure 3.

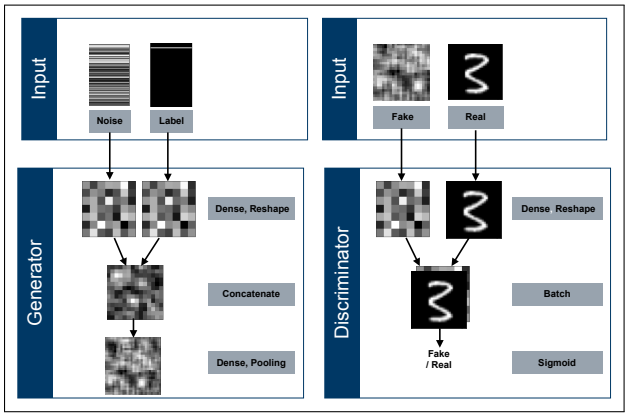

Fig. 3. Two distinct inputs get feed into the generator to create images. Different dense, reshape operations and finally the concatenation of the images transforms the images such that the model can be trained by using gradient decent as optimization technique.

As mentioned in the Introduction GAN-training faces some problems when it comes to generate realistic images. Therefore, a conditional GAN (CDCGAN)offers a more stable training process. Figure 4 illustrates the flow of information within the conditional deep convolutional neural network. After performing on a particular step of gradient decent, a maximum error is used to adjust the weights within the layers of G. Therefore a minimum error is used to adjust the weights within the layers of D.

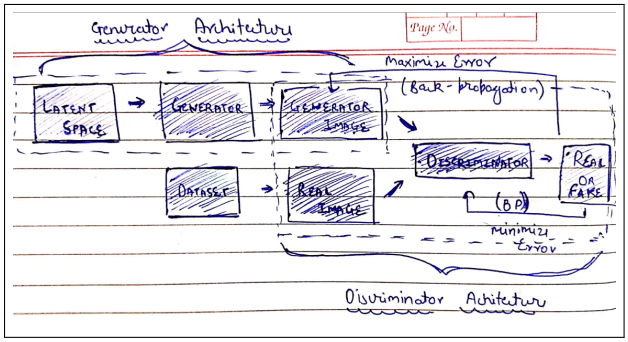

Fig. 4. Within the architecture of a cGAN, a maximum error gets back to the generator, and a minimum error gets back to the discriminator. So, it is possible to fit the weights of the model at the same time [6].

G is implement such that $-log\, D(G(z))$ instead of $log(1 - D(G(z)))$ gets minimized as it is suggested in the original GAN paper. It ensures stronger gradients for G early in training while keeping the same fixed point dynamics of G and D, as shown in the value function (Equation 2).

The performance of G compared to D is usually poor in the early stages of the training process. That means D can easily distinguish fake examples from real images. Therefore the learning of D is implemented such that it learns slower, which is done by running more than one training step for G for every step of D and lowering the learning rate of D.

GANs and CGANs stipulate a high-level framework with a lot of freedom in designing its various components. The prototype which will be used in order to find answers to the research question uses a so-called conditional deep convolutional neural network (CDCGAN) and uses the hyperparameters suggested by Phillip Isola and his colleagues in their paper Image-to-Image Translation with Conditional Adversarial Networks [4]. More precisely the batch size is 100, the learning rate 0.0002, the amount of training epochs is 100. Furthermore, it uses the Adam optimizer. To set a ground truth within the evaluation a more simple GAN will be implemented as well.

3. Results

Great care must be taken when it comes to presenting and evaluating the results of any GAN. The objective functions presented in the methods section can not tell how good the quality of the image is. Furthermore, there is not an exact definition of what precisely an image with good quality is. Instead, the objective functions for G and D are relative measurements. Their results indicate the performance with respect to each other. Research suggests using a visual examination of generated images as a starting point instead. Furthermore, quantitative measures, such as the inception score and the Frechet inception distance, can be combined with a qualitative evaluation approach to provide a robust evaluation [7] [8].

So, when it comes to evaluating the results to find an answer to the given research question, visual examination, nearest neighbour (kNN) and the Fréchet Inception Distance (FID) was used.

The FID score uses the last pooling layer before the output classification of images form a classifier model to capture computer vision-specific features of an input feature. The activations are calculated for a collection of real and generated images. The activations from each real and generated image are summarized, and then the distance between these two distributions get calculated by using a distance metric like the Gaussian Distance [9].

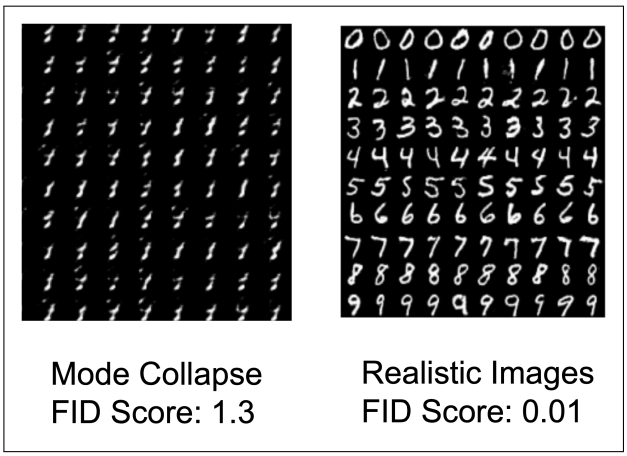

Fig. 5. Images generated by the DCGAN (left) and the cDCGAN (right) and their corresponding FID-Scores show that the images generated by DCGAN are more realistic.

Visual examination shows that just the cDCGAN can generate realistic images. As presented in Figure 5, the FID Scores indicates that the first impression is quite right. The score for the cDCGAN is too high as that it could be a realistic image. Furthermore, it seems that a mode collapse has appeared. The kNN method supports this impression as well because it shows that just one digit in the original dataset is close to the generated images (Figure 6).

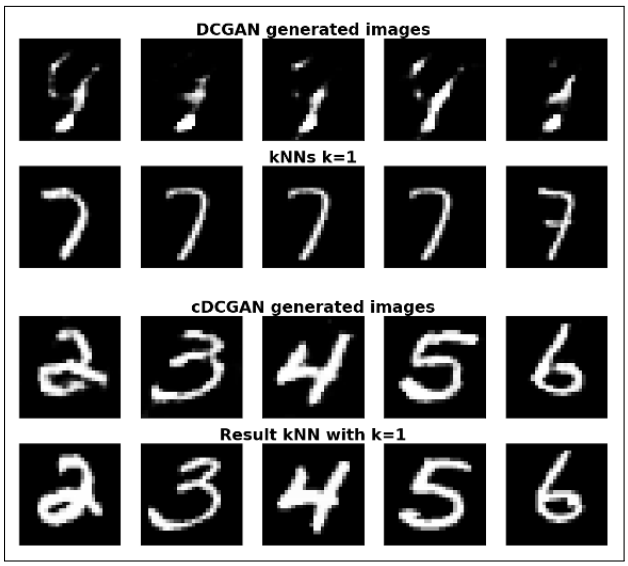

Fig. 6. The evaluation with the kNN shows that just one digit in the original dataset is close to the generated images.

If it comes to the evaluation of the training epochs itself Figure 7 shows that the loss fluctuates more within the CDCGAN training. At the same time the loss curves from the CDCGAN decreases more within the final epochs. The GAN trained for 3.32 seconds per epoch on average, whereas the CDCGAN trained 53.02 seconds per epoch on average.

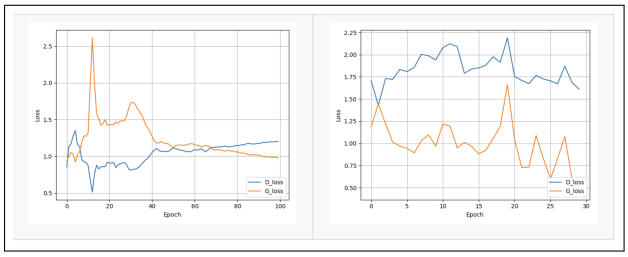

Fig. 7. The loss of $\boldsymbol{G}$ and $\boldsymbol{D}$ compared for a GAN (left) and a CDCGAN (right) architecture.

4. Discussion

As shown above, CGANs have performed remarkably well on the task of generating realistic images. However, as the complexity rices the results are getting surprisingly worse. For example, if the dataset contains photographs from objects on a conveyor belt, the generated images are not realistic at all. It is not even necessary to go towards visual examination because the results are so obvious, as can be seen in Figure 8. Note that to get the resulting image from Figure 8, two more dense layers and additional training time was necessary.

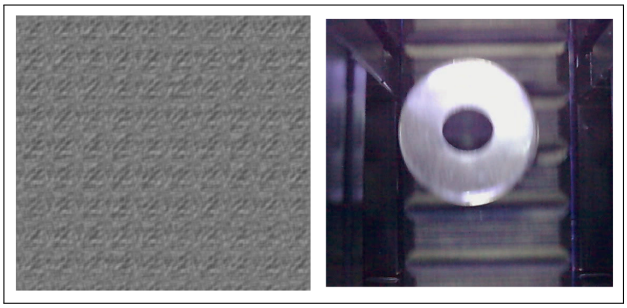

Fig. 8. The image on the left shows what happens if the domain gets changed from a simple dataset like handwritten digits to more complex dataset (right). It is evident that this result is not realistic (left); even a few adjustments were made to ensure the model works at all.

One opportunity to get better results is to train the model. Furthermore, within an iterating process, it could be helpful to tune a few hyperparameters. Some of the hyperparameters to play around with could be the following [8]:

- The discriminator's optimizer
- The learning rate
- Dropout probability
- Batch size

Tuning hyperparameters and changing the domain would lead to another experiment. Due to the lack of time within the seminar, this was just a theoretical discussion.

More time is necessary to improve the corresponding prototype before can evaluated like the prototype, which generates realistic handwritten digits.

5. Conclusion

Image generation conditioned on labels is a complex task that requires a model to learn not only the general data distribution for images, but also label embeddings for each individual class. It can be concluded from the results that well-designed CDCGANs can perform admirably well on this very complex task. Provided high-quality input data, these CGANs can be trained so that the combination of a conditional and the noise vector ($\boldsymbol{z}$) actually can generate images of handwritten digits. It is planned to expand this work in two directions. First, improving the training process by tuning the hyperparameters and secondly supporting any combination of training images and training labels.

Acknowledgment

I want to say thank you to Franz Köferl from the Machine Learning and Data Analytics Lab, I appreciate his thoughts towards the usual supervising and I am grateful for every piece of advice. I totally recommend this seminar to anyone who is willing to improve his skills in machine learning.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," Tech. Rep., Jun. 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[3] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784
[4] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016. [Online]. Available: http://arxiv.org/abs/1611.07004
[5] 2020. [Online]. Available: https://www.tensorflow.org/tutorials/generative/dcgan?hl=en
[6] J. Pawan, "My first encounter with gans," 2020. [Online]. Available: https://towardsdatascience.com/my-first-encounter-with-gans-6c0114f60cd7
[7] A. Borji, "Pros and cons of GAN evaluation measures," *CoRR*, vol. abs/1802.03446, 2018. [Online]. Available: http://arxiv.org/abs/1802.03446
[8] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016. [Online]. Available: http://arxiv.org/abs/1606.03498
[9] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR*, vol. abs/1706.08500, 2017. [Online]. Available: http://arxiv.org/abs/1706.08500