REPRODUCING AMBIENTGAN: GENERATIVE MODELS FROM LOSSY MEASUREMENTS

Xu Guo, 25538128
University of Southampton
xg1e13@soton.ac.uk

Yitian Zhang, 29695937
University of Southampton
yz1n18@soton.ac.uk

Ren Hu, 30516587
University of Southampton
rh1n18@soton.ac.uk

ABSTRACT

This paper aims to check the reproducibility of an ICLR paper AMBIENTGAN-GENERATIVE MODELS FROM LOSSY MEASUREMENTS [Bora et al. (2018)]. We selected some important experiments in the original paper and designed some new experiments. Basically, the results of our experiments support the original paper. We also found some implementation tricks and limitations of AmbientGAN during our experiments.

1 INTRODUCTION

Generative Adversarial Networks(GANs) are proved to be an efficient generative model without providing a parametric specification of a probability distribution function [Goodfellow & et al (2014)]. A GAN is composed of a generative model G and a discriminative model D. The generator G projects a randomly generated vector from any distribution to an output trying to capture the real data distribution. The discriminator D on the other hand, tries to distinguish the input generated by the generator with the real training data. The ideal situation of a well-trained GAN is that the generator G could generate outputs which subjects to the real data distribution and the discriminator could not distinguish which input is generated by G and output the probability 50% everywhere.

AmbientGAN is designed to generate better results from lossy measurements of the training samples. This is realized by introducing a specific model that could simulate the measurement of the data. Except generator G and discriminator D, AmbientGAN has a measurement function model noted by f_θ. In this case, the generated data X^g from generator will be measured in a certain way defined by the measurement function f_θ where θ stands for the parameters of the measurement function which are sampled from the distribution p_θ. The discriminator D has to distinguish the measured training samples $Y^r \in \{y_1, y_2, \ldots, y_s\}$ from the measured data generated by the generator Y^g. The objective of AmbientGAN could be expressed as below:

$$\min_G \max_D E_Y r_{\sim p_y}^r \left[q\left(D\left(Y^r\right)\right)\right] + E_{Z\sim p_z, \theta\sim p_\theta}\left[q\left(1 - D\left(f_\theta(G(Z))\right)\right)\right] \tag{1}$$

The optimization of generator and discriminator is applied in each mini-batch of the training process based on the loss function below:

$$G_{\text{Loss}} = E_{\text{minibatch}}\left[-q\left(\text{sigmoid}\left(D\left(f_\theta(G(Z))\right)\right)\right)\right] \tag{2}$$

$$D_{\text{Loss}} = E_{\text{minibatch}}\left[-q\left(\text{sigmoid}\left(D\left(Y^r\right)\right)\right)\right] + E_{\text{minibatch}}\left[-q\left(1 - \text{sigmoid}\left(D\left(f_\theta(G(Z))\right)\right)\right)\right] \tag{3}$$

Where q(.) is the quality function which could scale the output of the discriminator D. In our experiment, we use q(.) = log(.).

2 EXPERIMENTS

2.1 DESIGN AND METHODOLOGY

The idea of AmbientGAN could be implemented to a series of GANs such as DCGAN [Radford et al. (2016)] and Wasserstein GAN [Arjovsky et al. (2017)]. In this paper, we mainly use DCGAN as our baseline network and integrate the structure of AmbientGAN. Two GitHub repositories [1] [2] are

[1] https://github.com/ashishbora/ambient-gan

[2] https://github.com/shinseung428/ambientGAN_TF

used as references for our reproducing, however, both of the source codes do not fit our experiments well. In this case, we made many modifications on the original code.

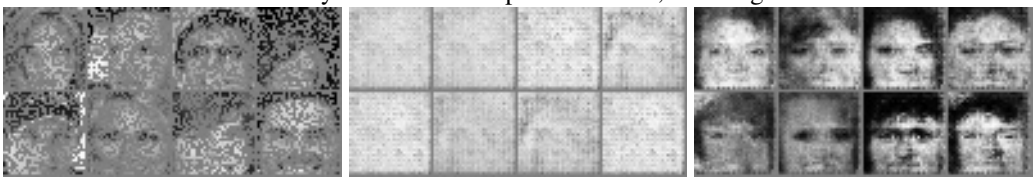
The original paper discussed a variety of measurement functions. Since our computing resource is limited, we chose Block-Pixels and Block-Patch as our main measurement for our experiences. Block-Pixels function set each pixel value of the image to 0 with the probability p. Block-Patch function randomly put the patch on an image and set all the pixels to 0 under the patch kernel.

We use MNIST and Celeb as our training dataset. The corresponding experiments will be illustrated in the following sections.

2.2 Celeb Experiments and Discussion

2.2.1 Performance of AmbientGAN with Block Pixel measurement

As there are 202,599 samples in dataset Celeb, we chose to convert the color images into gray level images and resize them to 32*32 in order to save computation resource. Furthermore, all the networks in this section are only trained in one epoch with 202,599 images.



(a) Measured training samples (Block Pixel p=0.5)

(b) Generated images by DCGAN (w/o AmbientGAN)

(c) Generated images by AmbientGAN

Figure 1: Performance of AmbientGAN & pure DCGAN

As is shown in figure 1, the pure DCGAN learned both the data distribution and the noise. However, the generated images from AmbientGAN successfully captured the underlying distribution of human face without nooise. We also test the anti-noise ability of AmbientGAN by using different probabilities (0.25,0.5,0.75) of Block Pixel measurement.

(a) Measured training samples (Block Pixel p=0.25)

(b) Measured training samples (Block Pixel p=0.5)

(c) Measured training samples (Block Pixel p=0.75)

(d) Measured training generated (Block Pixel p=0.25)

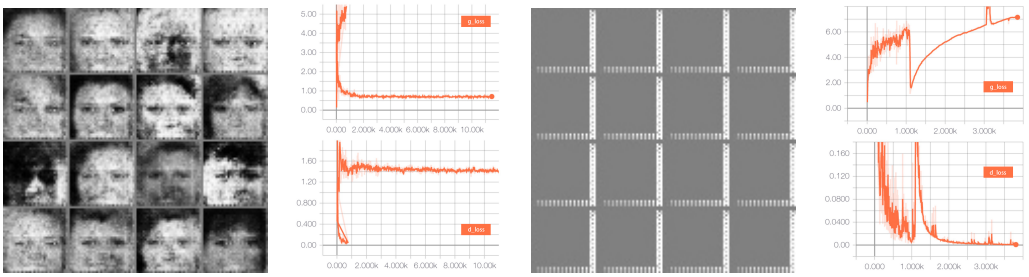
(e) Measured training generated (Block Pixel p=0.5)

(f) Measured training generated (Block Pixel p=0.75)

Figure 2: Performance of AmbientGAN on different level of noise (measurements)

Figure 2(f) shows AmbientGAN could guarantee the performance with extremely heavy noise.

For all the experiments above in this session, the generator network is composed of one linear layer and four de-convolutional layers. The discriminator network is composed of two convolutional layers and a linear layer in the end. During the experiment, we found that the structure of the generator and the discriminator could dramatically influence the performance of GANs. Only if both the generator and the discriminator have similar abilities, could the adversarial system work properly. If one of the network is too powerful, the loss for the other network will always be high and it will be hard for it to be optimised by gradient decent.



(a) Balanced AmbientGAN: generated images

(b) Balanced AmbientGAN: g_loss and d_loss

(c) Unbalanced AmbientGAN: generated images

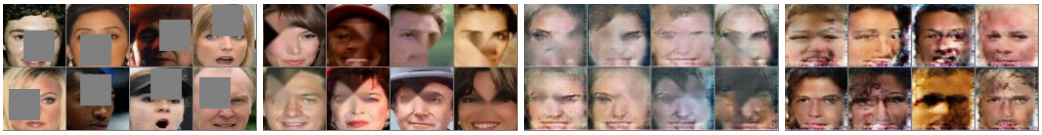
(d) Unbalanced AmbientGAN: g_loss and d_loss

Figure 3: Comparison of Balanced and Unbalanced AmbientGAN

Figure 3 show the performance of well-balanced AmbientGAN and unbalanced version. As is shown in figure 3(c) and (d), the unbalanced AmbientGAN could not learn any underlying data distributions after training. The loss of the generator keeps high, well the loss of the discriminator fells to zeros after several iterations. In contrast, the losses of well-balanced network for both generator and the discriminator keep low after a certain number of iterations.

2.2.2 Performance of AmbientGAN with Block Patch

In this section, we use Block-Patch as measurement function to train our model. In order to have relatively high-quality results, we use 64*64 color images for our training this time. Instead of directly using the measured data to train a normal DCGAN as a baseline, we try to use recovered images from measurement to train our baseline model. The recovery is done by a OpenCV tool called cv2.inpaint. Figure 4 shows the comparison between the baseline Figure 4(c)and the AmbientGAN Figure 4(d). The result shows that the baseline model could only learn the data distribution after measurement and recovery. However, the AmbientGAN could learn the real data distribution and generate high-quality images.



(a) Measured training images

(b) Recovered images with classic approach

(c) Generated images by recovered images

(d) Generated images by AmbientGAN

Figure 4: Comparison of classic approach and AmbientGAN

2.3 MNIST Experiments and Discussion

We experiment individual methods (Block-Pixel, Block-Patch and Keep-Patch) stated in the original paper, we add a new measurement function by cascading Block-Pixel and Block-Patch to give more complex circumstance. In every iteration, weights of generator is updated once and discriminator is updated 5 times according to original paper. A baseline without measurements is generated for comparison. We verified the model with both conditional and unconditional underlying DCGAN.

2.3.1 Performance of AmbientGAN with single measurement

Figure 5 shows three variation of GANs. The difference between conditional and unconditional is whether adding the class label into the input of generator and discriminator. Apparently, the result of conditional DCGAN (d) is better, while the rest of the methods (b)(c) are failed. The label might help the generator make a better difference between classes. Besides, unconditional DCGAN has a serious divergence problem during training which means it is hard to learn without label for generator. Another special feature of conditional DCGAN is that the class of the output will follow the label input with it. It could be observed that the order of output number is same as the input,

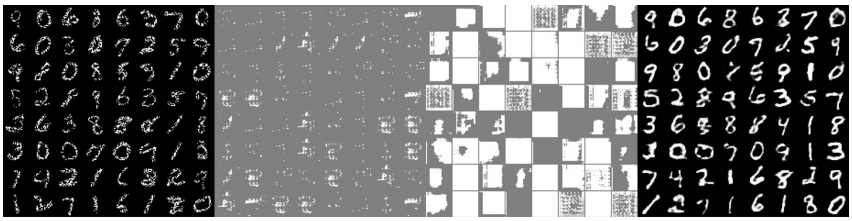


Figure 5: Results of Block-pixels with block probability=0.5: (a) input images, (b) baseline, (c) unconditional DCGAN,(d) conditional DCGAN

whereas the handwriting style is different, which means the diversity will be achieved within the same class.

2.4 Performance of AmbientGAN with cascading measurements

Figure 6: Result of cascading measurements: (a) block probability=0.5 and Keep-Patch size=14*14, (b) block probability=0.5 and Block-Patch size=14*14, (c) block probability=0.9 and Block-Patch size=14*14, (d) Different measurements on real and fake

Figure 6 shows the results of hybrid measurement with different parameters. Obviously, AmbientGAN could achieve a relatively consistent image quality under a different manner of damage according to figure 6(a), (b) and (c).

According to figure 6(d), if we apply a measurement A on real sample, but apply a different measurement B on generated sample, then the algorithm will fail. This is the limitation of AmbientGAN which assumes to know the specific measurements in advance.

3 Findings and conclusion

To sum up, we have the following findings. Firstly, the performance of AmbientGAN described is reproducible according to our experiments. Secondly, we found that, for the MNIST dataset, the original paper used label of each image combined with the image itself to train the AmbientGAN. This is a unusual training way. If we want to build a generative model to generate pictures of each number, it would be better to provide the information that which number we want to generate. Thirdly, when training our own AmbientGAN, we found that the structure of the generator and discriminator should match and form adversarial system. If one of the network is too powerful, it will be hard for the other network to learning anything. Finally, we think the weakness of AmbientGAN is that the assumption is we know what the measurement function is before training. In this case, AmbientGAN seems to be lack of ability of generalization and could be hard to fit in practise.

Source: https://github.com/COMP6248-Reproducability-Challenge/-AmbientGAN_Reproduce

References

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *CoRR*, abs/1701.07875, 2017.

Ashish Bora, Eric Price, and Alexandros G. Dimakis. Ambientgan: Generative models from lossy measurements. In *AmbientGAN: Generative models from lossy measurements*. ICLR, 2018.

Ian J. Goodfellow and et al. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pp. 2672–2680, 2014.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2016.