REPRODUCIBILITY CHALLENGE: MISGAN / LEARNING FROM INCOMPLETE DATA WITH GAN

Yu Hong Leung, Xiang Shen
Team: HelloWorld
{yhl1u19,xs5a19}@soton.ac.uk

1 INTRODUCTION

Generative adversarial network (GAN) is a method of adversarial learning that allows two neural networks to play against each other. It is characterized by data that needs to be fully observed during training. In view of this feature, MisGAN proposes that for incomplete data, you can use the GAN-based framework to learn two generators, namely a mask generator that explicitly models the distribution of lost data and a complete data generator. The complete data generator masks its outputs with the generated masks to train adversarially against the real incomplete data. Besides, a missing data imputer can also be appended to this framework and trained using a GAN as well.

The experiments in this paper is divided into three parts. The first part is to evaluate the different attributes of MisGAN on the MNIST dataset. The second part is to use ablation research to prove the rationality of MisGAN. The third part is to compare a series of baseline methods and MisGAN under a series of missing settings for the missing data interpolation tasks of MNIST, CIFAR-10 and CelebA datasets. For this challenge, we will focus on the first part of the experiments to qualitatively evaluate the proposed framework of MisGAN and its reproducibility on MNIST data.

2 IMPLEMENTATION

Their implementation mainly referenced two research papers: DCGAN and WGAN with gradient penalty (WGAN-GP). We did our best to implement MisGAN by only following the descriptions provided in the paper, not the code associated with original paper [1][2].

The MisGAN paper specifies that all models will be trained for 300 epochs with a training procedure of Wasserstein GAN with gradient penalty (WGAN-GP). We implemented WGAN-GP following Algorithm 1 in Gulrajani et al. (2017). For models with a missing data imputer, they will be trained for 1000 epochs with WGAN-GP as well.

All ground truth input data, including MNIST, are rescaled to the range of the sigmoid activation function $[0, 1]$. The ground truth masks are binary-valued: 1 is observed and 0 is missing. Then, following Eq. 3 in Li et al. (2019), the ground truth data $\boldsymbol{x}$ and mask $\boldsymbol{m}$ goes through a masking operator $f_\tau(\boldsymbol{x}, \boldsymbol{m})$ to produce incomplete data. Both the mask and image generators take a 128-dimension latent code from standard normal distribution as input. Remaining details on the architecture and hyperparameters are scattered across 3 papers.

Regarding the distribution of lost data, we consider square observation and random dropout in the following experiments.

2.1 CONV-MISGAN ON MNIST

All generators and discriminators (or critics since we are training with WGAN-GP) in Conv-MisGAN follow the DCGAN architecture without further details from MisGAN authors. The generators contain transpose convolutional layers while the discriminators contain convolutional layers.

From MisGAN paper

[1]The code for grid results visualisation is borrowed from the original implementation
[2]https://github.com/COMP6248-Reproducability-Challenge/MisGAN

- masking operator $\tau = 0$
- $\alpha = 0.2$ when optimising the mask generator G_m by minimising $L_m + \alpha L_x$
- Outputs of the image generator go through sigmoid activation
- Outputs of the mask generator go through sigmoid activation with temperature $\lambda = 0.66$

From DCGAN paper

- Batchnorm after each layer of transpose convolution in the generators (except the generators' output layer and discriminators' input layer)
- ReLU activation in the generators (except the output layer)
- LeakyReLU activation in the discriminators (except the output layer)(slope = 0.2)
- Weights are initialised from $\mathcal{N}(0, 0.02^2)$

From WGAN-GP paper

- Penalty coefficient $\lambda = 10$
- Number of critic iterations per generator iteration $n_{critic} = 5$
- Hyperparameters for Adam optimisers in generators and discriminators: $\alpha = 0.0001, \beta_1 = 0.5, \beta_2 = 0.9$
- Replace batchnorm with layernorm in the discriminator
- Output layer of the discriminators is fully-connected to predict the score of 'realness' given an image (instead of sigmoid activation)

Then, our choice is to rescale the MNIST image to 32x32 because we want to follow closely to the DCGAN architecture without resizing the image to 64x64. Therefore, following Figure 1 in the DCGAN paper (Radford et al., 2015), we removed a transposed convolutional layer and a convolutional layer in the generators and discriminators respectively, and halved the feature maps in all hidden layers.

Hence, the shapes of generators' intermediate outputs are as follows: $(128) \rightarrow (512, 4, 4) \rightarrow (256, 8, 8) \rightarrow (128, 16, 16) \rightarrow (1, 32, 32)$. The discriminators will go in reverse from $(1, 32, 32)$ to a single linear output in the output layer.

2.2 FC-MisGAN on MNIST

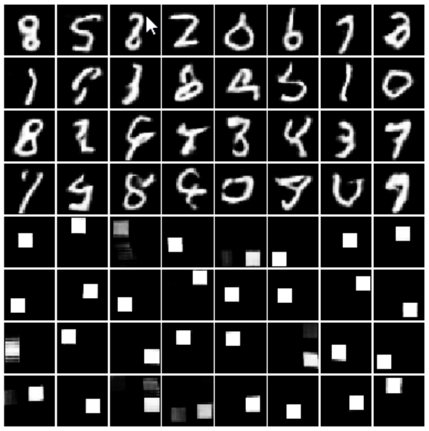
The architecture of FC-MisGAN is much simpler and is clearly stated in the paper. We implement the architecture of the generators and discriminators as described in Appendix D of the original paper, with only ReLUs and dense layers.

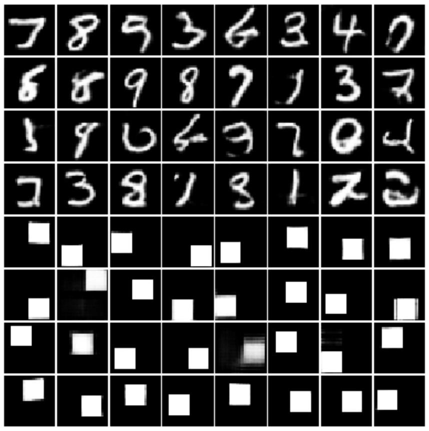
3 Findings and Discussion

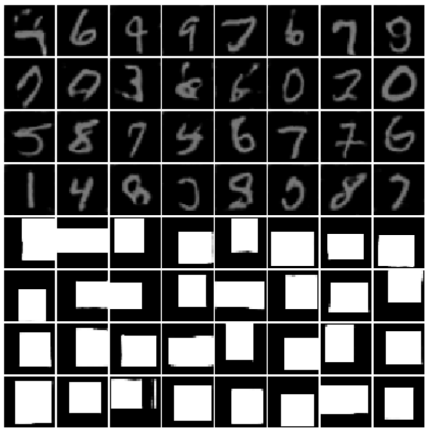
In Figure 1 and 2, the data missing pattern is a random square observation of different lengths, and missing for anywhere else. In Figure 3 the data missing pattern is independent dropout with probability = 0.5. Using the specification provided above, each figure includes 32 samples of generated complete data (above) and 32 generated masks (below) after training. Figure 1 demonstrates the results of Conv-MisGAN while Figure 2 and 3 are results from FC-MisGAN.

Compared with the figures in the original paper, we can observe that the generated square masks are less clearly defined. In Figure 1a, some generated masks have artifacts (faded grey squares) that do not form a clear white square mask while others have a blurry boundary. Fuzzy boundaries are also observed in Figure 1b. The solution with length 20 seems to have the worse performance on the mask generator as many of them are not proper squares.

The generated digits seem to be better reproduced without significant differences. However, most noticeably in Figure 1c and 1d, the solution seems to have converged poorly with the generated digits


(a) length = 9


(b) length = 13


(c) length = 20

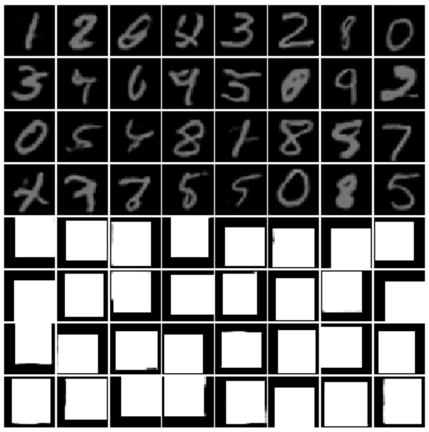
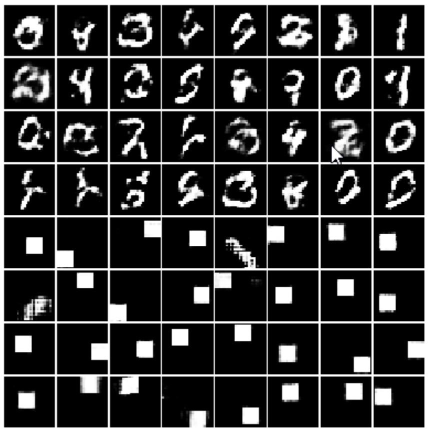

(d) length = 25

Figure 1: Conv-MisGAN, square observation


(a) length = 9

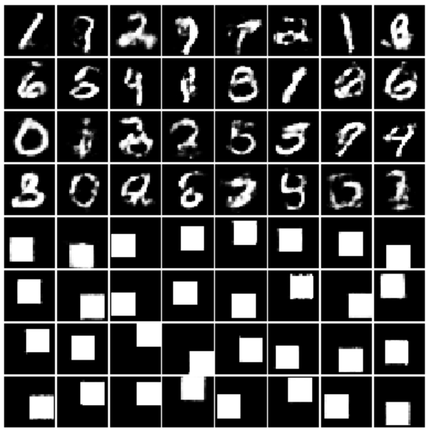
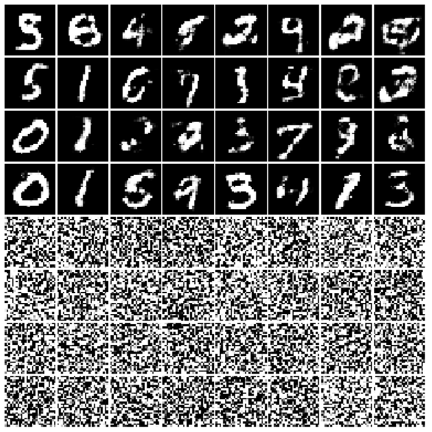

(b) length = 13

Figure 2: FC-MisGAN, square observation

faded in grey even though the shape of the digits is clear. Throughout different experimentation, we still have not figured out why this would happen when the length of the square mask increases.

For FC-MisGAN, the generated digits are much poorer in shape with a coarser outline compared to Conv-MisGAN. Especially in Figure 2a, there are two generated masks that appear to be shift-


(a) length = 9

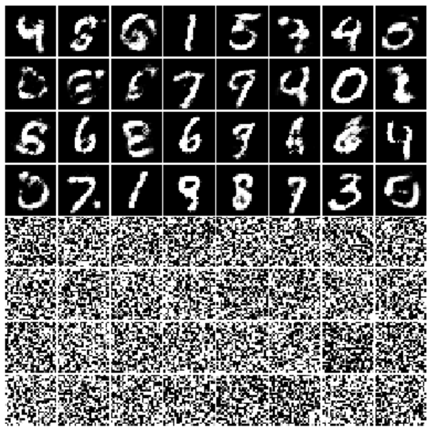

(b) length = 13

Figure 3: FC-MisGAN, independent dropout, probability = 0.5

ing across pixels and have not converged. This is in accordance with the authors' comments that Conv-MisGAN produces visually better examples than FC-MisGAN. In Figure 3 with independent dropout, the generated digits also appear to have generalised worse, e.g. some samples have blurrier and thicker stokes compared to Conv-MisGAN. However, it is not obvious that independent dropout has worse samples compared to square observations, as the authors claimed.

4 Conclusion

The part that needs the most development effort is to implement the procedure of Wasserstein GAN with gradient penalty. Our group took a lot of time to understand the algorithm in the paper and how to implement it in code. On the other hand, the architectures in both the Conv-MisGAN and FC-MisGAN are relatively clear and easy to understand. Also, the implementation of masks and missing data is much more simple just by extending a Dataset class in Pytorch.

Our group mostly focuses on reproducing the results of MisGAN rather than comparison with other models like ConvAC. We wanted to focus on a qualitative evaluation of the reproduced results without the need to implement FID as evaluation metrics, since it would take more time to understand and run all the models as well. Relatively, Conv-MisGAN requires much more computational resources as one model takes about half a day to train on Google Colab's GPU or ECS GPU Compute Service. FC-MisGAN trains much faster with a simpler architecture and fewer weights.

We have tried to implement MisGAN with an imputer but failed to produce any imputation that resembles a digit. We are not sure if there is an implementation error, but we still included the trial implementation in the GitHub repository.

To sum up, we believe that for the parts of experiments we did, the results are mostly reproducible in agreement with some of the authors' claims. In comparison, our implementation seems to have a poorer performance, given that they were quite vague on their DCGAN architecture and hyperparameters choices. Upon closer look, their actual implementation follows more closely to the code associated with WGAN-GP. Still, the experiments are reproducible following the paper and the two major references.

References

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *ArXiv*, abs/1704.00028, 2017.

Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. Learning from incomplete data with generative adversarial networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=S1lDV3RcKm`.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.