Mask CycleGAN : Unpaired Multi-modal Domain Translation with Interpretable Latent Variable

Minfa Wang
Stanford University
minfa@stanford.edu

1 Introduction

CycleGAN [6] is a popular approach for unpaired image-to-image translation between two domains. It has been proven to be effective in a wide variety of domain translation tasks, including horse-zebra, apple-orange, summer-winter, etc. While it keeps inspiring generative modelling community to build up more and more applications and research ideas, CycleGAN has its limitations too. One notable limitation is that the translation is deterministic and hence lack of variation. People have discovered that achieving multi-modality through CycleGAN is challenging [2, 7], largely due to the reason that the source image is at high dimensionality and usually causes the generator network to ignore other noise sampled from other distribution. A natural idea to enable multi-modal image generation is by introducing additional latent variables, which are often modeled as Gaussian distributions. However, samples from Gaussian distributions are generally lack of interpretability.

Mask CycleGAN aims to address both issues above by using pixel mask as latent variables. Figure 1 shows a high-level overview of our architecture and comparison with other popular architectures. In later sections, we will elaborate the architecture in details. We will show its formulation is a full generalization of CycleGAN, and hence is at least equally expressive. Moreover, the pixel mask offers great control of the image generation outcome.

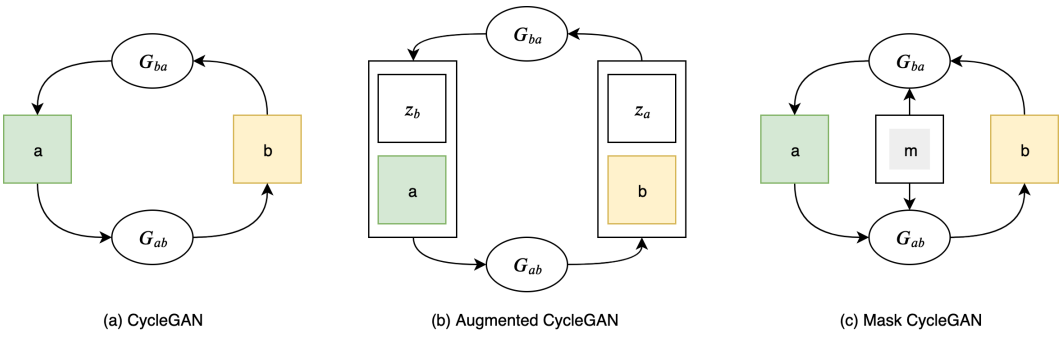


Figure 1: Comparison of variants of CycleGAN.

2 Related Work

CycleGAN [6] is a popular approach for unpaired image-to-image translation between two domains. One of the most innovative ideas from this work is *cycle consistency*, which encourages the mapping between two domains to be invertible, and indirectly alleviate the problem of mode collapse. CycleGAN is capable of generating visually appealing images.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Augmented CycleGAN [2] brings multi-modality into CycleGAN by augmenting it with two latent variables Z_a and Z_b, and corresponding encoders E_a, E_b and discriminators D_{Z_a}, D_{Z_b}. The latent variables are optimized through the minimax game between the encoder and the discriminator.

BicycleGAN [7] is an architecture for paired multi-modal image-to-image translation. The name Bicycle refers to the two cycles from 1) Conditional Variational Autoencoder GAN (*cVAE-GAN*) and 2) Conditional Latent Regressor GAN (*cLR-GAN*).

Image inpainting is a technique to reconstruct a image patch from a partially covered or blurred image. The input of image inpainting can be thought as a clear source image applied on a mask, which is similar to our setup. Some of the works [5] in this field inspires the generator design of the Mask CycleGAN to impose soft constraint on pixel invariance on certain image region during transformation.

Attention is technique to discover and make use of region of interests from the input by assigning different weights to different parts of the input. When the input is image, the weight is typically computed at pixel level, and the attention weight map can be thought of as a soft mask. People [4] have attempted to use unsupervised attention mask to improve generative modelling.

3 Problem statement

The task for CycleGAN-alike architectures is unpaired image-translation between two domains. At inference time, an image from source domain will be given as input, and the output will be an image from target domain, retaining the basic features of the input image. The mathematical formulation is introduced in the Notation section below.

3.1 Notation

We introduce the following terminologies to help elaborate the technical approach. Same as CycleGAN, we have a as a sample image from the source domain, b as a sample from the destination domain, and G_{AB} as generator mapping an image from source to destination. With mask m, we could derive the following properties: $\tilde{b} = G_{AB}(a, m)$ as the fake image in destination domain, $a' = G_{BA}(\tilde{b}, m)$ as the recovered image in source domain. Other terms like $\tilde{a}$, b' and G_{BA} are defined symmetrically.

Regarding the mask, we call *masked region* to be the region where the information of the pixel values of the image is kept, and the *contextual region* to be the region outside the *masked region*.

4 Approach

Our architecture is based on CycleGAN. Please refer to [6] for more details of its original architecture. The sections below elaborate the key modifications we made to incorporate masks into the whole system.

4.1 Masking

4.1.1 Discussion: binary vs. continuous mask

A binary mask is a mask that has value 1 on its masked area, and 0 elsewhere. It is a kind of **hard** mask, eliminating all the information from the contextual region of the image.

A continuous mask is a mask that has float number between 0 and 1 for all its dimensions, representing the weights of different pixels of the image. It is a **soft** mask where the boundary between masked and contextual regions could be blurry, and the information for contextual region will be partially retained after the mask is applied on the image.

One of the reasons that it is challenging to introduce multi-modality in CycleGAN is that when you feed a concatenation of image features and latent variables as the input to the generator, the

generator quickly learns that the dimensions of the latent variables provides little additional values in optimizing the overall objective, and hence zeros out those dimensions, forfeiting multi-modality.

Thanks to the interpretability of the mask, we can re-design the interaction between the mask and the input image to force the generator to respect the effect of the mask. The revamped generator design is detailed in the following Generator section. To avoid unexpected information leaking to the generator, we choose binary masks in this work. Below is a list of masking schemes that we has considered.

4.1.2 Centered-square masking scheme

The masked region always is centered in the image, has square shape, and has a size of $(0.5|0.8|1.0)$ * image size. Figure 2(a) provides an visualization of the masks generated from this scheme.

The centered-square masking scheme is simple to understand and fast to evaluate. On the other hand, it is very limited on the amount variation it is capable of producing.

4.1.3 Multi-rectangles masking scheme

Multi-rectangles masking scheme, elaborated in Algorithm 1, is a generalization of the centered-square masking scheme. It allows more variations in size, position and (compound) shape of the mask, encouraging the generator to generalize better. The limitation of this masking scheme is that it still produces rectangular edges. Figure 2(b) provides an visualization of some samples generated from this masking scheme.

Algorithm 1: Multi-rectangles masking scheme

```
// Minimal number of rectangles to draw.
Set MinMaxNumRects = 5, MinNumRects = uniform(1, MinMaxNumRects) ;
// Minimal accumulative area relative to the whole image area.
Set MinSumRelArea = 0.15 ;
// Size of image; lower and upper bound of individual rectangle size.
Set Size = 128, MinRectSize = Size / 10, MaxRectSize = Size ;
// Initialize variables to keep track of status in the loop.
Set numRects = 0 ;
Set sumRelArea = 0.0 ;
Set mask = zeros(3, Size, Size) ;
while numRects < MinNumRects or sumRelArea < MinSumRelArea do
    // Randomly choose the top left corner.
    i0 = uniform(0, Size - MinRectSize) ;
    j0 = uniform(0, Size - MinRectSize) ;
    // Randomly choose the bottom right corner.
    i1 = uniform(i0 + MinRectSize, min(i0 + MaxRectSize, Size)) ;
    j1 = uniform(j1 + MinRectSize, min(j0 + MaxRectSize, Size)) ;
    mask[:, i0:i1, j0:j1] = 1 ;
    numRects += 1 ;
    sumRelArea += ((i1 - i0) * (j1 - j0)) / (Size * Size) ;
end
```

Some properties of this masking scheme:

- When MinRectSize = 1 and MaxRectSize = 2, this masking scheme is equivalent to sample individual pixels independently.
- When MinSumRelArea = 1, this masking scheme will always generate the full mask, making the whole algorithm become equivalent to CycleGAN.
- The time complexity of this algorithm is O(max(MinNumRects, MinRelArea / (MinRectSize / Size)^2)).

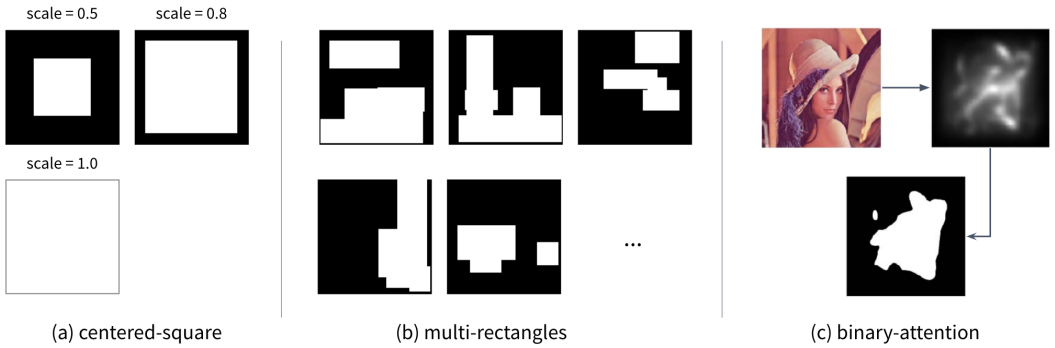


Figure 2: Visualization and comparison of different Masking schemes. (c) is adopted from an image in [1].

4.1.4 Binary-attention masking scheme

If we have access to some pre-trained image attention network, which is able to produce a pixel level attention map, then we could potentially convert it to a binary mask by binarizing the attention weights on all pixels based on some threshold.

Figure 2(c) provides an illustration of the binary-attention masking scheme.

4.2 Generator

The generator design is one of the critical parts of the architecture. Figure 3(a) shows the plain-vanilla generator used in original CycleGAN that maps a source image to a destination image. It is easy to see that the mapping is deterministic with a fixed source image. In literature, when people try to introduce the additional latent variable z, a common approach [7, 2] is concatenates the latent variable vector to some intermediate vector representation of the source image. The drawbacks of this approach are 1) the influence of latent variable is hard to interpret and control, and 2) empirically the generator often tends to ignore variations in the latent variable.

Since our latent variable, m, is interpretable, we could design its interaction with the source image in a way that enforces the masking behavior. The interaction is defined through the **mask encoder** E, with its architecture detailed in Figure 3(b). The full generator design is shown in Figure 3(c), where complicated domain translation logic, captured in G, is required to only depend on masked region of the source image.

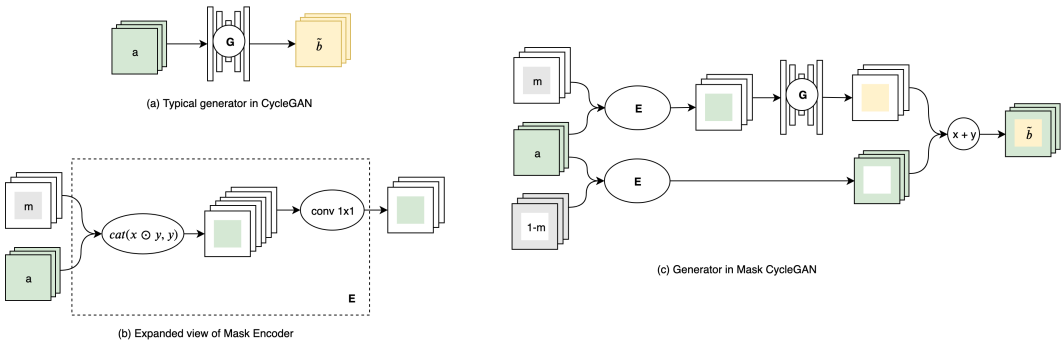


Figure 3: Comparison of generators. (b) is the expanded view of the mask encoder E used in (c).

4.3 Losses

Similar to CycleGAN, we optimize the model by minimizing three pairs of losses. We have made modifications on each pair to accommodate the architecture change introduced by the mask.

GAN loss For each triplet $(a, \tilde{a}, m)$, we will have two discriminators D_{AM} and D_{AF}. The first generator D_{AF} is functionally equivalent to the discriminator D_A in original CycleGAN, which is trying to distinguish a true image from a generated one for the source distribution, and implicitly encourages the generator to produce overall coherent image inside and outside the masked area. The second discriminator D_{AM} is responsible for the same task for the masked image pair. We have the final GAN loss as the normalized weighted sum of the two discriminators' losses. Formally:

$$\mathcal{L}_{GAN}^{AF} = -\mathbb{E}_{a\sim A}[\log D_{AF}(a)] - \mathbb{E}_{\tilde{a}\sim\tilde{A}}[\log(1 - D_{AF}(\tilde{a}))]$$
$$\mathcal{L}_{GAN}^{AM} = -\mathbb{E}_{a\sim A}[\log D_{AM}(a \odot m)] - \mathbb{E}_{\tilde{a}\sim\tilde{A}}[\log(1 - D_{AM}(\tilde{a} \odot m))]$$
$$\mathcal{L}_{GAN}^{A} = \lambda_{GAN}^{M}\mathcal{L}_{GAN}^{AM} + (1 - \lambda_{GAN}^{M})\mathcal{L}_{GAN}^{AF}$$

There are two remarks. First, we need to introduce the second discriminator D_{AM} because despite the fact that generator tries to produce coherent images, there would generally exists some discontinuity around the mask boundary, making the task relatively easy for D_A, and in turn causes gradient vanishing problems to the generator. Even if the generator manages to learn, without D_{AM}, the generator would tend to ignore the mask and suffer from mode collapse. Second, when $\lambda_{GAN}^{M} L_{GAN}^{AM} = 0$, this objective falls back to the GAN objective in original CycleGAN.

Cycle loss facilitates cycle-consistency for translation between two domains. The high-level idea is that when a image goes through a forward mapping and then a backward mapping, it should recover to itself, which requires the forward mapping to preserve the information of the source image. This is an effective method to prevent mode collapse. Our loss objective is the normalized weighted sum of the cycle losses for the masked and context areas.

$$\mathcal{L}_{CYC}^{A} = \lambda_{CYC}^{M}||(a - a') \odot m||_1 + (1 - \lambda_{CYC}^{M})||(a - a') \odot (1 - m)||_1$$

Different weights are applied for masked and context areas, and the weights could be adjusted to control how strict we hope the generator to keep the pixels in the context area intact. When $\lambda_{CYC}^{M} = 0.5$, this loss falls back to the cycle loss in original CycleGAN (with a scale factor).

Identity loss states that when you try to map an image from the destination domain to the destination domain, it should map to itself. Formally:

$$\mathcal{L}_{IDT}^{A} = ||a - G_{BA}(a, m)||_1$$

Note this formulation is identical to the identity loss in original CycleGAN. We don't need special treatment for the masked area because we intentionally want this specific mapping to be invariant of the mask.

4.4 Full objective

The full objective is

$$\mathcal{L} = (\mathcal{L}_{GAN}^{A} + \mathcal{L}_{GAN}^{B}) + \lambda_{CYC}(\mathcal{L}_{CYC}^{A} + \mathcal{L}_{CYC}^{B}) + \lambda_{IDT}(\mathcal{L}_{IDT}^{A} + \mathcal{L}_{IDT}^{B})$$

4.5 Algorithm

Algorithm 2: Masked CycleGAN

initialize weights and optimizers ;
initialize G_{AB}, G_{BA}, D_{AF}, D_{BF}, D_{AM}, D_{BM} ;
while *True* **do**
  Read next batch a, b ;
  Sample m from M ;
  $\tilde{b}$ = $G_{AB}(a, m)$, $\tilde{a}$ = $G_{BA}(b, m)$;
  a' = $G_{BA}(\tilde{b}, m)$, b' = $G_{AB}(\tilde{a}, m)$;
  `// -- optimize generator below --`
  l_cyc_a = λ^M_{CYC} * cyc_loss(a * m, a' * m) + (1 - λ^M_{CYC}) * cyc_loss(a * (1 - m), a' * (1 - m)) ;
  l_cyc_b = λ^M_{CYC} * cyc_loss(b * m, b' * m) + (1 - λ^M_{CYC}) * cyc_loss(b * (1 - m), b' * (1 - m)) ;
  l_idt = idt_loss(a, G_{BA}(a, m)) + idt_loss(b, G_{AB}(b, m)) ;
  l_gan_a = (1 - λ^M_{GAN}) * gan_loss($D_{AF}(\tilde{a})$) + λ^M_{GAN} * gan_loss($D_{AM}(\tilde{a}$ * m))) ;
  l_gan_b = (1 - λ^M_{GAN}) * gan_loss($D_{BF}(\tilde{b})$) + λ^M_{GAN} * gan_loss($D_{BM}(\tilde{b}$ * m))) ;
  l_g = (l_gan_a + l_gan_b) + λ_{CYC} * (l_cyc_a + l_cyc_b) + λ_{IDT} * l_idt ;
  l_g.backward() ;
  optimizer_g.step(l_g) ;
  `// -- optimize discriminator below --`
  l_da = (1 - λ^M_{GAN}) * d_loss(D_{AF}, a, $\tilde{a}$) + λ^M_{GAN} * d_loss(D_{AM}, a * m, $\tilde{a}$ * m) ;
  l_db = (1 - λ^M_{GAN}) * d_loss(D_{BF}, b, $\tilde{b}$) + λ^M_{GAN} * d_loss(D_{BM}, b * m, $\tilde{b}$ * m) ;
  l_d = l_da + l_db ;
  l_d.backward() ;
  optimizer_d.step(l_d) ;
end

5 Experiments

5.1 Setup

We evaluated the model on the following datasets: *MNIST-SVHN*, *Horse-Zebra*, *Monet-Photo*, *Vangogh-Photo*. The image resolution for all datasets is 128x128, except for MNIST-SVHN, which is 32x32. For all the experiments, we used the same hyper-parameter settings as follow: $\lambda^M_{GAN} = 0.7$, $\lambda^M_{CYC} = 0.3$, $\lambda_{CYC} = 10$ and $\lambda_{IDT} = 5$.

We experimented with the centered-square and multi-rectangles masking schemes, and evaluated the algorithm both qualitatively and quantitatively. For quantitative analysis, we used the FID score of test set and the set generated with full mask as **baseline**.

5.2 Quantitative Results

Frechet Inception Distance (FID) [3] is a method to measure the performance of a generative model by computing the following distance on the inception feature representation of the generated data distribution and true data distribution fitted by multivariate Gaussian. We use FID as the main quantitative metric for our model.

5.2.1 Train with centered-square masking scheme

We computed FID scores for various pairs of datasets derived from the *Horse-Zebra* dataset. The results are represented in two matrices shown in Figure 4, where (a) shows FID scores for various horse datasets, and (b) shows FID scores for various zebra datasets.

There are several conclusions we could draw from the matrices:

- $FID_*(scale = *, train) < FID_*(scale = *, test)$. The generator was trained on the training dataset, and hence would be able to emulate the training data distribution better.

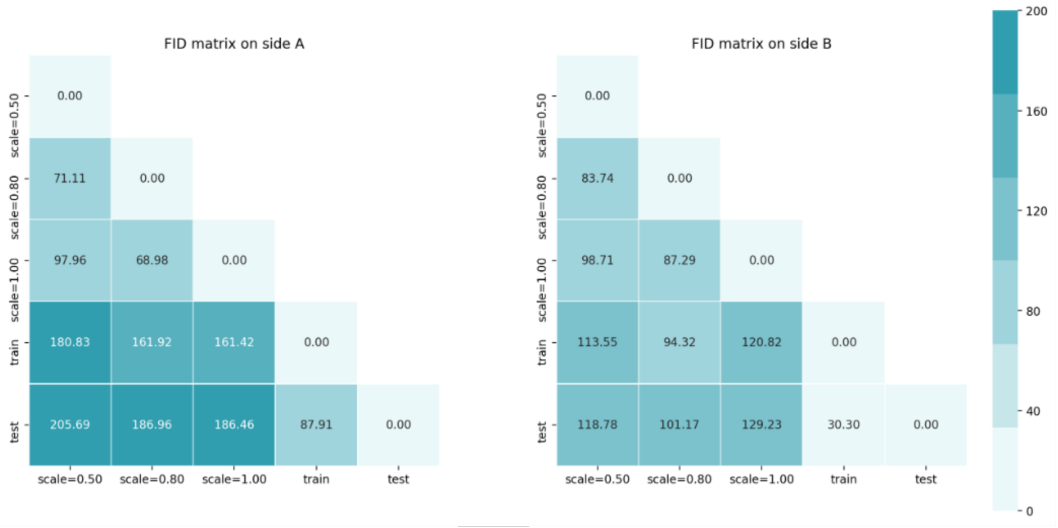


Figure 4: FID matrix of Mask CycleGAN trained with **centered-square** masking scheme on Horse-Zebra dataset. (a) shows FID scores among real and generated A:horse datasets. (b) shows FID scores among real and generated B:Zebra datasets.

- $FID_A(train, test) > FID_B(train, test)$. According to FID, the horse dataset has more variations in style. This characteristic matches with our human-eye judgement.
- $FID_A(scale = *, test) > FID_B(scale = *, test)$. The generator performs worse in emulating the horse distribution. This could be attributed to the intrinsic difficulty of horse dataset.
- $\boldsymbol{FID_B(scale = 0.8, test) < FID_B(scale = 1.0, test)}$. This is probably the most interesting finding. Usually, the larger the mask scale, the more information is exposed to the generator, and hence generator has more expressive power to fit a distribution. However, for the horse dataset, we see that the model performs better with the comparably smaller mask. One explanation is that typically the object of interest is presented around the center of the image, and a smaller mask actually filters out some noise and produces some mild regularization.

5.2.2 Train with multi-rectangles masking scheme

We also conducted FID evaluation on the model with multi-rectangles masking scheme. The result is shown in Figure 5.

From the matrices, we could draw the same conclusions as the previous experiment. In addition, What is worth noting is that training with this multi-rectangles masking scheme, we achieve FID score of 84.83 on horse test set using centered-square scale=0.8 mask at inference time, which surpasses performance of the model trained with centered-square masks. This is a sign that more variations in masks in training could help the model generalize better in inference.

5.3 Qualitative Results

We examine the qualitative results of the model through a output grid shown in Figure 6. The model used in the grid was trained only on centered-square masks, but was evaluated on both centered-square and multi-rectangles masks. It is shown from the output that Mask CycleGAN is robust in translation across many image domains, and is able to generalize to work with mask it has never seen during training.

5.3.1 Generalization to round mask

One interesting question is that if only trained with rectangular masks, will the model be able to generalize to round masks? We did the analysis by feeding the generator with round masks of

Figure 5: FID matrix for model with **multi-rectangles** masking scheme, evaluated on horse-zebra dataset.

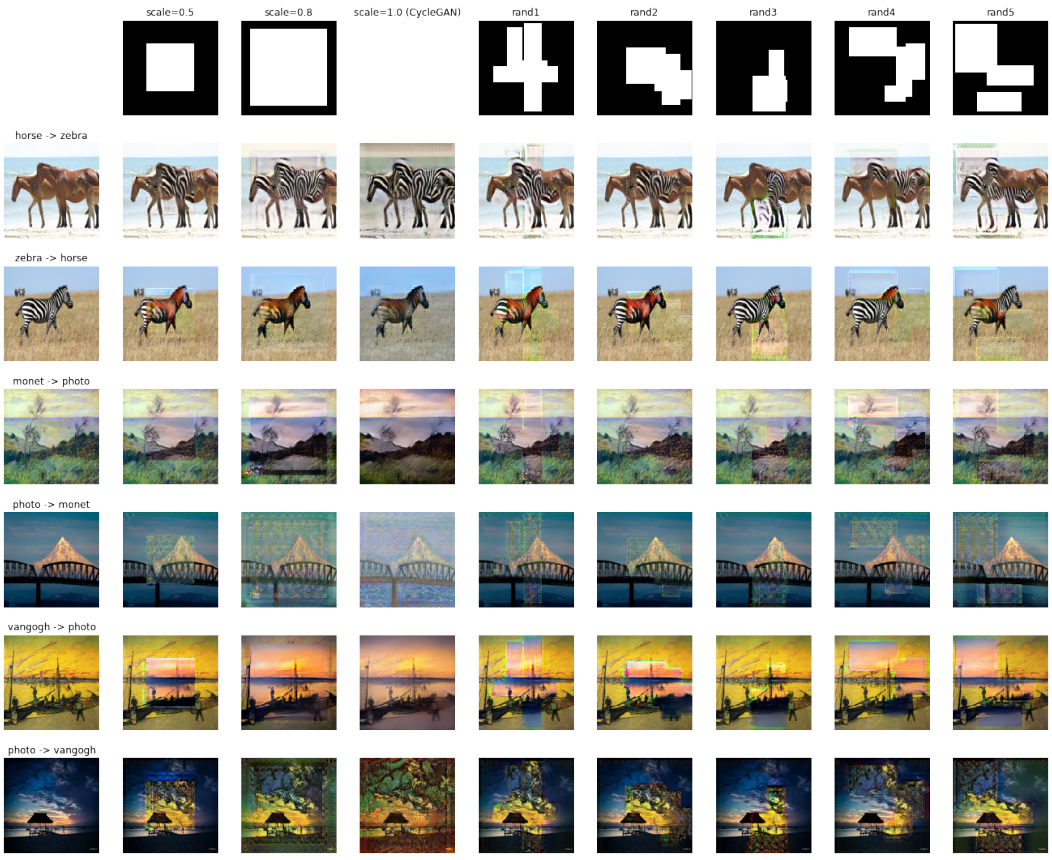


Figure 6: Outputs grid of Mask CycleGAN evaluated on several datasets with different masks. First row shows different masks used on different columns. First column shows the source images and the related tasks. Other cells show the generator output with the corresponding source image and mask as inputs.

different scales and the outputs are shown in Figure 7. Interestingly, model trained with centered-square masking scheme appears to generalize better to round masks.

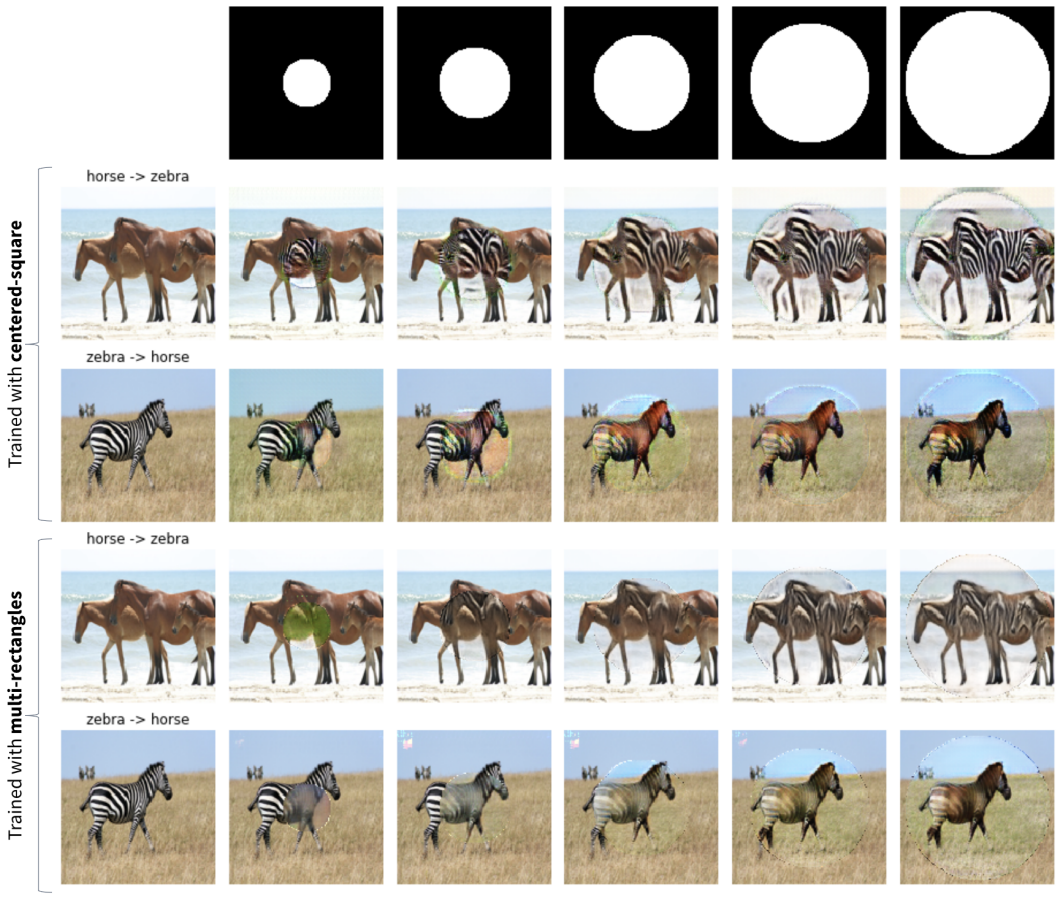


Figure 7: Outputs grid of Mask CycleGAN trained with different masking schemes.

6 Conclusion

In this work, we proposed a novel generative modelling algorithm called **Mask CycleGAN**. We introduced the motivation of the idea, the mathematical formulation, and quantitative and qualitative evaluations of the algorithm. We illustrate that Mask CycleGAN is capable of bringing variations in CycleGAN-alike generators in a controllable manner. We believe this architecture will open doors for a series of interesting applications.

In the future, we plan to further improve the robustness of the algorithm along two directions. One direction lies in the design of generator, where we may experiment with more sophisticated architecture applied on the context region of the image. The other direction is to explore different masking schemes like binary-attention to help the model generalize better to more variations of masks during inference.

7 Acknowledgement

Special thanks to Kristy Choi for her feedback and advice offered in the development of the project. The code for this project is available at `https://www.github.com/minfawang/mask-cgan`.

References

[1] C. Agarwal, A. Bose, S. Maiti, N. Islam, and S. Sarkar. Enhanced data hiding method using dwt based on saliency model. pages 1–6, 09 2013. ISBN 978-1-4673-6188-0. doi: 10.1109/ISPCC.2013.6663414.

[2] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. C. Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. *CoRR*, abs/1802.10151, 2018. URL `http://arxiv.org/abs/1802.10151`.

[3] 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*, abs/1706.08500, 2017. URL `http://arxiv.org/abs/1706.08500`.

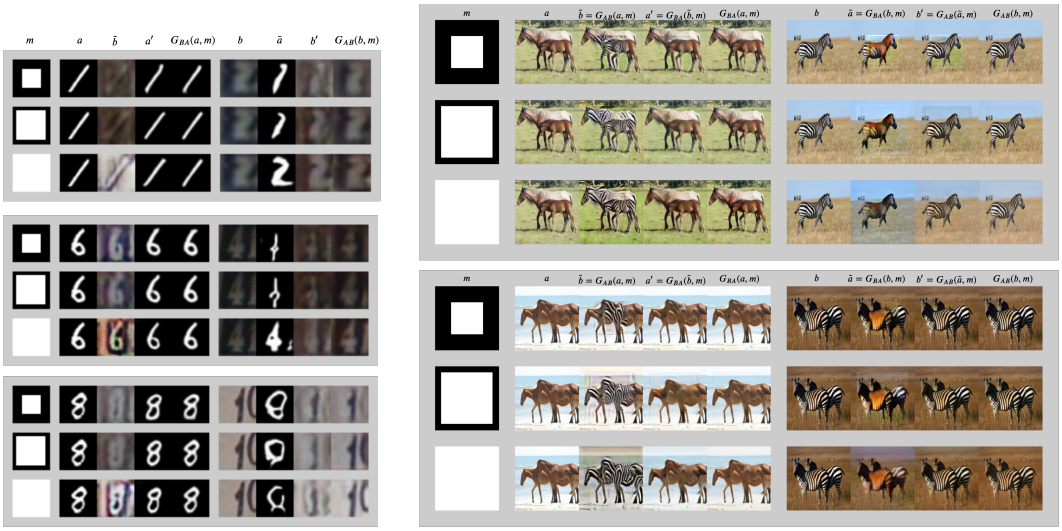
[4] Y. A. Mejjati, C. Richardt, J. Tompkin, D. Cosker, and K. I. Kim. Unsupervised attention-guided image to image translation. *CoRR*, abs/1806.02311, 2018. URL `http://arxiv.org/abs/1806.02311`.

[5] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016. URL `http://arxiv.org/abs/1604.07379`.

[6] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. URL `http://arxiv.org/abs/1703.10593`.

[7] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. *CoRR*, abs/1711.11586, 2017. URL `http://arxiv.org/abs/1711.11586`.

8 Appendix



(a) MNIST-SVHN

(b) horse-zebra

Figure 8: Outputs of Mask CycleGAN. (a) shows the results from MNIST-SVHN translation. (b) shows horse-zebra translation. Please zoom in to inspect the details.