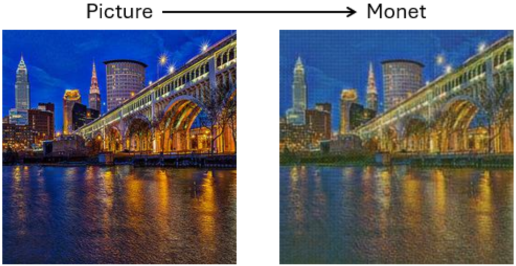
Picture Monetization

Photo-to-Monet Painting Translation using a CycleGAN



Joakim Colpier Jeroen Verweij

Delft University of Technology, The Netherlands

Outline

Introduction – GANs

GAN: **G**enerative **A**dversarial **N**etwork

GAN

Converts random noise into a distribution (e.g. image)

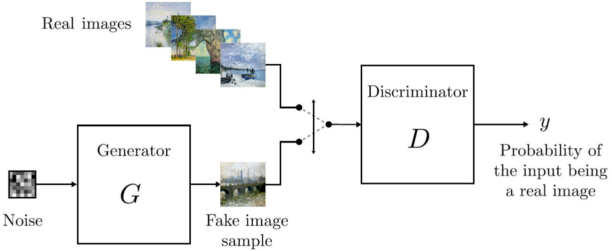


Figure: Examples of distribution conversions using GAN [1].

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros (2017a). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*

Introduction – GANs

GAN: **G**enerative **A**dversarial **N**etwork

Train a **generator** and **discriminator**

Generator&Discriminator

- Generator: generate images
- Discriminator: decide whether an image is fake or not

Core idea: Use the generator to train the discriminator, and use the discriminator to train the generator

Comparison: Evolution of wolves and deers.

Figure: Wolf: discriminator; Deer: generator (image taken from [2]).

[2] IllustAC (n.d.). Image taken from https://en.ac-illust.com/clip-art/24355246/wolf-hunt

Introduction – CycleGANs

CycleGAN

- Converts data **between two distributions**
- Keeps the most important features

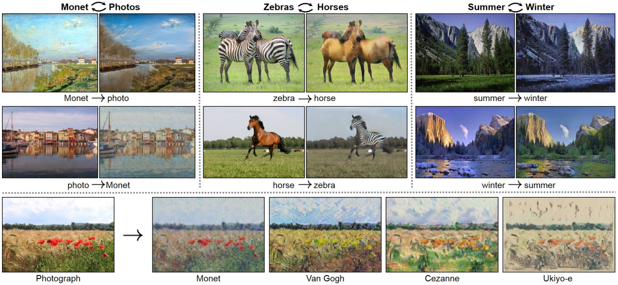


Figure: Examples of distribution conversions using CycleGAN [3].

[3] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros (2017a). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*

Introduction – Goal

Goal

- Implement CycleGAN between 2 datasets
  - Pictures
  - Monet Paintings
- Implement an evaluation metric
- Compare results with theoretical Bayesian implementation

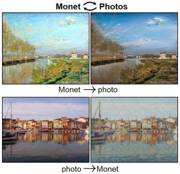


Figure: Example of conversion between Monet and picture distributions, cropped from [4].

[4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros (2017a). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*

Architecture

For each distribution we have 2 generators and 1 discriminator.

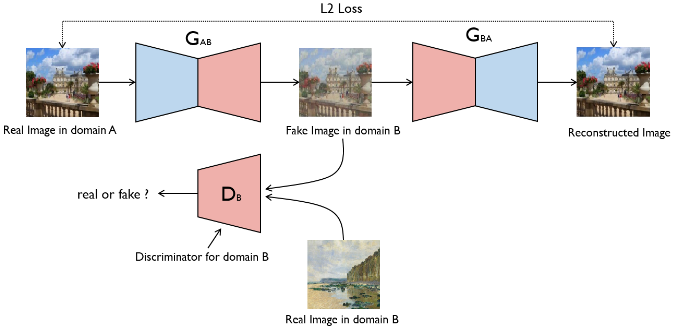


Figure: General structure of CycleGAN in the case of photo to Monet translation adapted from [5] with [6]

[5] *CycleGAN — kaggle.com* (n.d.). https://www.kaggle.com/code/himasha0421/cyclegan/notebook. [Accessed 25-03-2025]

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros (2017b). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*

Architecture – Discriminator

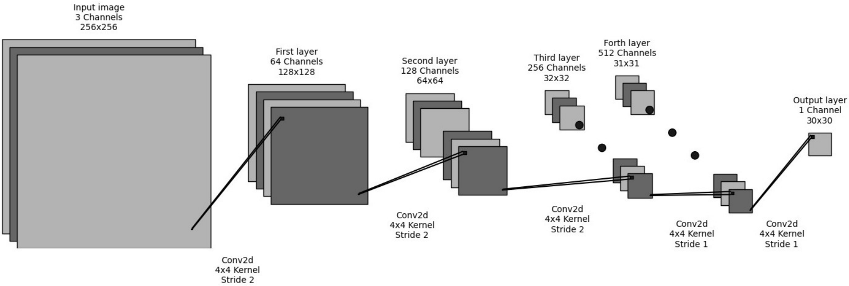


Figure: Architecture of the discriminator, created using PyDrawNet[7]

[7] *GitHub - nhansendev/PyDrawNet: A python utility for plotting neural network (and other) diagrams — github.com* (n.d.). `https://github.com/nhansendev/PyDrawNet`. [Accessed 25-03-2025]

Architecture – Generator

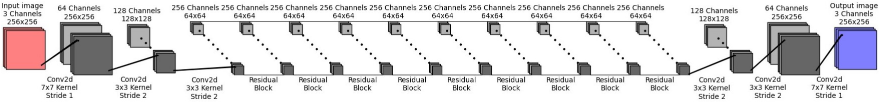


Figure: Architecture of the generator, created using PyDrawNet[8]

[8] *GitHub - nhansendev/PyDrawNet: A python utility for plotting neural network (and other) diagrams — github.com* (n.d.). https://github.com/nhansendev/PyDrawNet. [Accessed 25-03-2025]

Architecture – Training

The training losses are our tools to make the generators and discriminators compete.

Discriminator loss

- Discriminator on real image, take loss wrt 1 (true image)
- Discriminator on fake image (from generator), apply the discriminator and take the loss wrt 0 (false image)

Generator loss

- **Adversarial loss:** "Trick" the discriminator: take the loss of the discriminator on a generated image wrt 1
- **Cycle loss:** Make image change distribution twice, and compare with original
- **Identity loss:** Change image into its own distribution, and compare with original

Architecture – Loss

Mathematically, this amounts to the following losses (m=Monet, p=pictures). λs are hyperparameters.

$$L_D = \frac{(1 - D_m(x_m))^2 + D_m(G_m(x_p))^2}{2} + \frac{(1 - D_p(x_p))^2 + D_p(G_p(x_m))^2}{2}$$

$$L_G = \underbrace{(1 - D_p(G_p(x_m)))^2 + (1 - D_m(G_m(x_p)))^2}_{\text{adversarial loss}}$$
$$+ \underbrace{\lambda_C(\|G_p(G_m(x_p)) - x_p\|_{L_1} + \|G_m(G_p(x_m)), x_m\|_{L_1})}_{\text{Cycle loss}}$$
$$+ \underbrace{\lambda_I(\|G_m(x_m) - x_m\|_{L_1} + \|G_p(x_p) - x_p\|_{L_1})}_{\text{Identity loss}}$$

Architecture – Bayesian Approach

The normal GAN can be reformulated as follows [9] (CycleGAN implemented in a similar way)

- z: generator input, following source distribution
- x: data, following target distribution
- α_g, α_g: prior parameters for generator and discriminator
- θ_g, θ_d: parameters of the generator and discriminator.

$$p(\theta_g | z, \theta_d) \propto \left(\prod_{i=1}^{n_g} D(G(z^{(i)}; \theta_g); \theta_d) \right) p(\theta_g | \alpha_g)$$

$$p(\theta_d | z, x, \theta_g) \propto \prod_{i=1}^{n_d} D(x^{(i)}; \theta_d) \times \prod_{i=1}^{n_g} (1 - D(G(z^{(i)}; \theta_g); \theta_d)) \times p(\theta_d | \alpha_d)$$

[9] Y. Saatchi and A. G. Wilson (2017). "Bayesian GAN". In: *Advances in neural information processing systems*, pp. 3622–3631

Architecture – Bayesian Approach

Classical approach

- Uniform priors $p(\theta|\alpha)$
- Deterministically sample from posterior using maximum likelihood

Bayesian approach

- Use arbitrary priors (although in practice some might lead to faster computations)
- Randomly sample from posterior

Conclusion: Bayesian approach is a generalization of the classical one

Architecture – Motivation of Bayesian Approach

- Generalization of classical – more possibilities
- Fixes mode collapse
  - Poor training
  - Due to generator/discriminator unbalance (one much better than the other)
  - E.g. generator learns a few examples that trick the discriminator – little to no learning
- Broader outcome distribution, more realistic

Evaluation – Assessing image quality

- Qualitative vs quantitative inspection

	Fréchet distance	MMD distance
Inception embeddings	✗ Weak image embeddings ✗ Normality assumption ✗ Sample inefficient ✗ Biased estimator	✗ Weak image embeddings ✓ Distribution-free ✓ Sample efficient ✓ Unbiased estimator
CLIP embeddings	✓ Rich image embeddings ✗ Normality assumption ✗ Sample inefficient ✗ Biased estimator	✓ Rich image embeddings ✓ Distribution-free ✓ Sample efficient ✓ Unbiased estimator

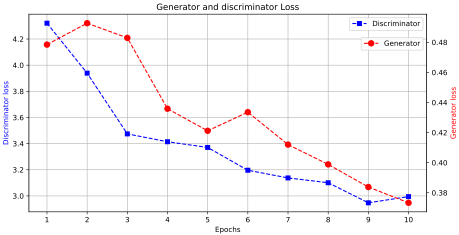
Figure: Comparison of FID and CMMD as a performance metric[10]

[10] S. Jayasumana, S. Ramalingam, A. Veit, D. Glasner, A. Chakrabarti, and S. Kumar (2024). *Rethinking FID: Towards a Better Evaluation Metric for Image Generation*. DOI: 10.48550/ARXIV.2401.09603

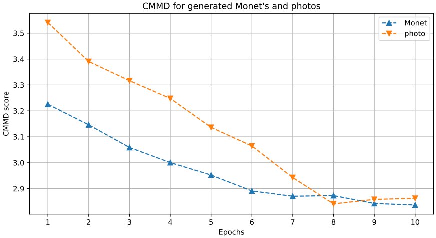
Evaluation – Data usage

- 300 Monets and 7038 pictures
- 75 % training 25% validation
- 10 epochs
- After each epoch
    - Calculate average discriminator and generator loss
    - Forward validation data through both generators
    - Calculate CMMD for validation data

Results – Metrics



(a) Losses of the generator and the discriminator



(b) CMMD score

Figure: Quantitative performance of the network

Results

(a) Original

(b) Epoch 10

Figure: Evolution of Fake Photos

Results

(a) Original

(b) Epoch 10

Figure: Evolution of Fake Monets

Discussion

+ We do see a difference
+ Saturation
+ Sharpness
+ Broad strokes – smoother in the pictures
– Still not on par with the literature
– Scores low, but not very low
– Scores stagnate

Discussion

We have identified several points that might help explain our findings.

- Too few datapoints, especially Monet paintings (only 300)
  - In CMMD, see similar results
  - Images seem to be equally good
  - Models from both distributions work together however, so more data might still lead to better results
- Too few ran epochs
  - We see that the graphs are slowing down, CMMD even seeming to converge
- Mode collapse
  - Discriminator cannot keep up with the generator
  - Discriminator doesn't learn, and thus generator doesn't learn
  - Fixed by using a Bayesian approach

Conclusion

- We found that our implementation of CycleGAN could learn from data
- It did not learn very well however, especially compared to literature
- Bayesian CycleGANs seem to resolve our problems

For future research

- Do Bayesian CycleGAN perform as well as predict from the theory?
  - What are its limitations?
  - E.g. slow training speed
  - E.g. blurry images as for VAE
- Can the training speed for CycleGANs be sped up?

Conclusion – Ethical Aspects

Should we continue researching CycleGANs?

+ Useful, for example self-driving cars
  - Convert video into images that can be read by the program
  - No need for expensive radar equipment
  - Reduces costly manual labeling
- Mimic voice, create fake videos
- Hard to separate true from false
- Can be used for tricking, frauds, disinformation...

References I

CycleGAN — kaggle.com (n.d.). https://www.kaggle.com/code/himasha0421/cyclegan/notebook. [Accessed 25-03-2025].

GitHub - nhansendev/PyDrawNet: A python utility for plotting neural network (and other) diagrams — github.com (n.d.). https://github.com/nhansendev/PyDrawNet. [Accessed 25-03-2025].

IllustAC (n.d.). Image taken from https://en.ac-illust.com/clip-art/24355246/wolf-hunt.

Jayasumana, S., S. Ramalingam, A. Veit, D. Glasner, A. Chakrabarti, and S. Kumar (2024). *Rethinking FID: Towards a Better Evaluation Metric for Image Generation*. DOI: 10.48550/ARXIV.2401.09603.

Saatchi, Y. and A. G. Wilson (2017). "Bayesian GAN". In: *Advances in neural information processing systems*, pp. 3622–3631.

References II

Zhu, J.-Y., T. Park, P. Isola, and A. A. Efros (2017a). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on.*

— (2017b). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on.*