Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks-
A re-implementation for ECE50024 (2023)

Arnab Saha [*1]

Abstract

Zhu et al[1], researched a novel technique that avoids the need for resource-heavy methods of gathering paired datasets and instead brought a generative network making use of double generator neural networks and discriminator neural networks to create a cycle of generation of images, which are kept in check with cycle-consistency constraints to prevent unrealistic images. The paper demonstrates the success of this novel technique over style transfer, image colorization, and domain adaptation for different datasets, all of which remain unpaired.

1. Introduction

The popularity of Generative Adversarial Networks(GANs) rose exponentially since their introduction by Ian Goodfellow et al.[2] in 2014, especially in the field of image manipulation, which included exciting stuff like image-to-image translation, image restoration, and even new image generation. With its vast opportunities, came along the flaw of relying on the availability of paired training image datasets. The availability of such paired datasets could come across to be a resource-heavy or computation-heavy affair or even sometimes not possible at all for several real-life scenarios.

The paper by Zhu et al[1], introduced an innovation built on the framework of a Generative Adversarial Network, to solve the bottleneck of requiring paired datasets. The concept of Cycle-GAN functions to provide translations between different unpaired domains, without the need for supervised learning as usually exercised by general GAN models. Cycle-GAN instead employs supervision at the level of complete domains, with the objective that domain 'A' translates into the characteristics of domain 'B' in such a manner that they are barely distinguishable. This is where the supervision for Cycle-GAN comes in, to make sure the generated images remain within the bounds of the two available domains and does not synthesize unrealistic images. The supervision combats the recurring issues of mode collapse, where all input images map to the same output image and the optimization fails to make progress.

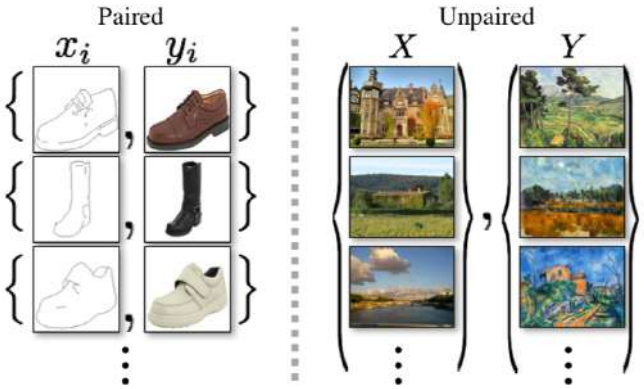

Figure 1. Example of Paired vs Unpaired Datasets[1], as observed in the paired dataset every image from set x_i is directly related to its corresponding image in y_i, which is far from the car for the unpaired datasets, who have no correlation between set X and Y

The method used to employ such set-based supervision by the paper was a newly introduced 'cycle-consistency loss' along with the usual adversarial loss to embody the cyclic sense of the proposed neural network. The cycle-consistency loss essentially incorporates the assumption that if images from set 'A', are translated into having characteristics, be it stylization or changing of objects of set 'B', if these constructed images are translated back into set 'A', it should be pair-to-pair identical for a perfect Cycle-GAN model. Therefore while running the Cycle-GAN model, the cycle-consistency loss and adversarial loss are brought into the domains to achieve Unpaired Image-to-Image translation.

2. Related Work

Pytorch and Deep Learning

For a successful implementation of Cycle-GAN, knowledge of deep learning, as well as how to implement said deep learning techniques are some rudimentary knowledge that is absolutely necessary. Some plausible packages that would

allow the use of deep learning in a Google Colab environment include Keras and PyTorch. For this paper, PyTorch was made use of, with use of their well-fleshed documentation and tutorials[8]. For the mathematical knowledge of layers, activation functions, convolution, etc., lecture slides by Prof. Qui Guo were the key foundation to achieve any form of re-implementation.

Image-to-Image Translation
Cycle-GAN is an advancement in the phenomenon of Image-to-Image translation, hence being able to develop a fundamental understanding of the principles behind functions to achieve image-to-image translations, be it through paired datasets or unpaired. Since the work by Zhu et al[1], was based on the pix-to-pix method developed by Isola et al. [3], understanding their inferences and fallacies was important in the reimplementation of Cycle-GAN. Isola et al.[3], worked with paired datasets, which provided a good foundation.

Generative Adversarial Networks
Since the fundamental part of Cycle-GAN is, in fact, the GAN, a strong understanding of the principles behind Generative Adversarial networks was garnered through Ian Goodfellow et al.[2], the original paper introducing GANs. They provided a framework demonstrating the fundamentals of building a GAN and an overview of GANs, and identified various methods for training GANs, with suitable examples. Alec Radford et al.[4], provides analysis in more depth, with deeper neural network architecture for building Generator and Discriminator models, and reaches the conclusion of decent results from unlabeled data, providing space for growth in the unpaired image-translation space.

Adversarial/Consistency Loss
While Zhu et al.[1] does provide a brief context towards the principles for Adversarial loss and Consistency Loss, with identity loss being a unique concept to the paper, the explanation is still lacking depth, which is where Radford et al.[4] and Ian Goodfellow et al.[2] come in clutch with more examples and depth for the mathematical equations, allowing a tinge of flexibility in formulating the mathematical formulae used for the re-implementation. Dual-GAN[5] provides some breadth to the concept of using the different losses in different scenarios to yield functioning similar to Cycle-GAN but yet quite different.

Neural Network Architecture
The neural network structures for Generator were based on Johnson et al.[6], while the discriminator is a variation of a Patch-GAN[3]. So the relevant papers were briefed upon to first understand how the generators and discriminators were meant to function on an architectural level, but upon more understanding, also allowed a bit of freedom and ideas for plausible changes while experimenting with the re-implementation.

Evaluation
Zhu et al.[1], made use of two methods of evaluation, AMT(Amazon Mechanical Turk) and FCN score. To understand how segmentation-based FCN scores are calculated and set up, Long et al.[7] was crucial, as they write in quite specific detail about different FCN methods and their respective scores, etc. AMT was not possible to be accessed for the reimplementation.

3. Method

The methodology for the re-implementation can be divided into subsections as demonstrated by the block diagram given below:

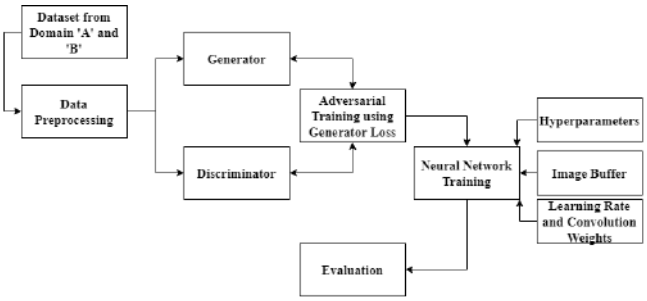

Figure 2. Block Diagram for Reimplementation

3.1. Data Pre-processing

Figure 3. Training Dataset pipeline through Google Drive

In the re-implementation of the paper, the dataset[9] by Taesung from the University of California, Berkeley was utilized, specifically the datasets for Summer-to-Winter, Apple-to-Orange, Horse-to-Zebra, and Vangogh-to-Photo. The datasets were uploaded into a google drive and pipelined to suit the code created in a Google Colab notebook environment. Due to the nature of the structure of Cycle-GAN, it was not necessary to discretely split the data to create another validation class, as the provided dataset already

splits the data into test and train for each domain for the specific translation experiment, i.e, for example, the dataset for Summer-to-Winter contains folders with images, named Train A, Test A, Train B, and Test B where 'A' is Summer images and 'B' are Winter Images.

Since no paired image datasets were used, a singular dataset loader class was created, with two modes: 'test' and 'train', to pipeline the uploaded datasets. The images in the dataset were then pre-processed, after loading the same to ensure all the images are in RGB format, and then transposed these images to fit the structure used by the tensors in PyTorch Module, since the structure for the PILLOW module and PyTorch module are inverted. Finally, the loaded images are normalized using the convolution weights specified by Zhu et al.[1]

3.2. Model Architecture

The brief neural network model architecture for a Cycle-GAN consists of two distinct network models, one for the generator model and the other for the discriminator model. Since we make use of two generators and two discriminators respectively, we define a generator model and discriminator model separately.

Generator Model: The architecture for the Generator Neural network model is based on the neural network structure in Johnson et al.[6], which includes three convolution layers with Re-LU and Instance Norm, two convolution layers with stride 'half', and a convolution layer to map the features to RGB, and also includes a total of 18 residual blocks, which is double the number of residual blocks as used by Zhu et al.[1]

The structure for these residual blocks consists of two 3x3 convolutional layers with the same number of filters on each of these layers.

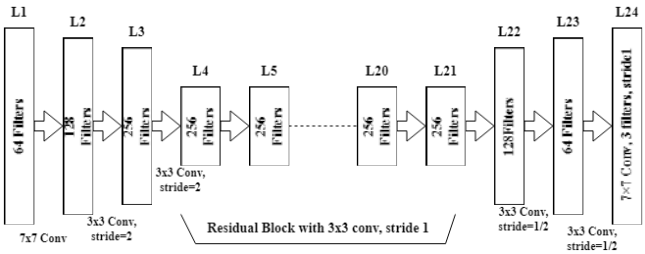

Figure 4. Architecture of the Generator Network

Discriminator Model: The Discriminator Neural Network Model is based on a 70x70 PatchGAN[3], with four 4x4 convolutional layers, with instance norm and leakyReLU with slope 0.2. Note that for the first layer, Instance-Norm is skipped and only taken into consideration for the other three layers.

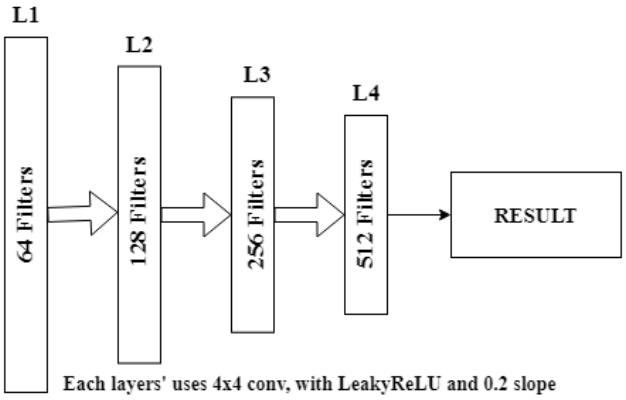

Figure 5. Architecture of the Discriminator Network

3.3. Adversarial Training

In an adversarial training process, the generator and discriminator networks are trained. The generator tries to produce realistic images that the discriminator can be tricked by, and the discriminator tries to tell the difference between real and fake images. This is primarily achieved by three main losses; Adversarial Loss, Cycle-Consistency Loss, and Identity Loss. These are then further combined to calculate Generator Loss and Discriminator Loss.

Adversarial Loss:A crucial part of the CycleGAN is the adversarial loss, which is utilized to train the discriminator network to tell the difference between real and fake images. The binary cross-entropy between the discriminator output and the ground truth labels is used to calculate the adversarial loss. When an image is input into the discriminator network, it outputs a probability score that indicates whether the image is real or fraudulent. A scalar value in the [0, 1] range, where 0 denotes a fake image and 1 denotes a real image, is the discriminator's output. This adversarial loss can be mathematically denoted as:

$$L_{adv} = -\big[z\log(D(x)) + (1-z)\log(1-D(G(y)))\big] \quad (1)$$

- L_{GAN} is the symbol for adversarial loss
- z is the ground truth label (1 for real images, 0 for fake images)
- $D(x)$ denotes the discriminator output for a real image x
- $G(y)$ is the generator which tries to look similar to the ground truth

The adversarial loss plays the role of the catalyst in a min-max game between the discriminator and the generator.

Whilst the generator tries to minimize this adversarial loss, the discriminator continuously tries to maximize this, as part of the cycle, which is the novelty of CycleGAN and makes it possible for the synthesis of realistic images.

Cycle-Consistency Loss: If a network is only based on adversarial loss, with large enough datasets, the network may end up translating the target images into any random domain, which could induce the cases where the target images end up characterizing a completely different domain instead of the one intended to be mapped. This is where cycle-consistency loss comes in, it enforces bijective mapping between two domains. This emphasizes that, if an image is translated from Domain 'A' into Domain 'B' if the new image is translated back to Domain 'A', it should be identical to the original image.

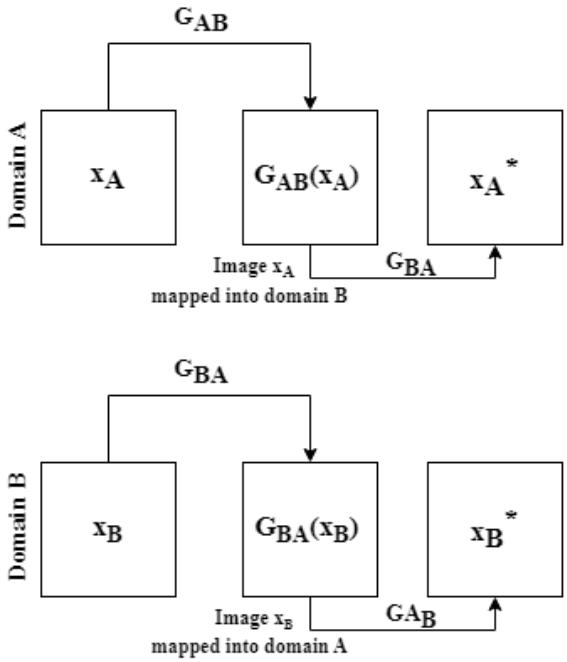

Figure 6. Shows the mapping of an input into the target domain, and reconstruction into x_A* or x_B*, depending on the mapping, where the cycle consistency loss would be the difference between the original input image and reconstructed output of the same image, through both generators.

Mathematically, the cycle-consistency loss is an L1 Loss, which can be denoted as:

$$L_{cyc} = ||G_{BA}(G_{AB}(x_A)) - x_A||_1 + ||G_{AB}(G_{BA}(x_B)) - x_B||_1 \quad (2)$$

where:

- L_{cyc} is the symbol for cycle consistency loss
- x_A is an image from Domain A
- x_B is an image from Domain B
- G_{AB} is the generator that maps domain A to domain B
- G_{BA} is the generator that maps domain B to domain A

Identity Loss: The identity loss is the final loss taken into consideration, with the goal of preservation of the identity of the input image. Zhu et al.[1] achieves this by mapping an image from a domain onto itself. Identity Loss prevents the introduction of false tints or shades into the constructed images, which would be a significant issue for translations involving paintings.

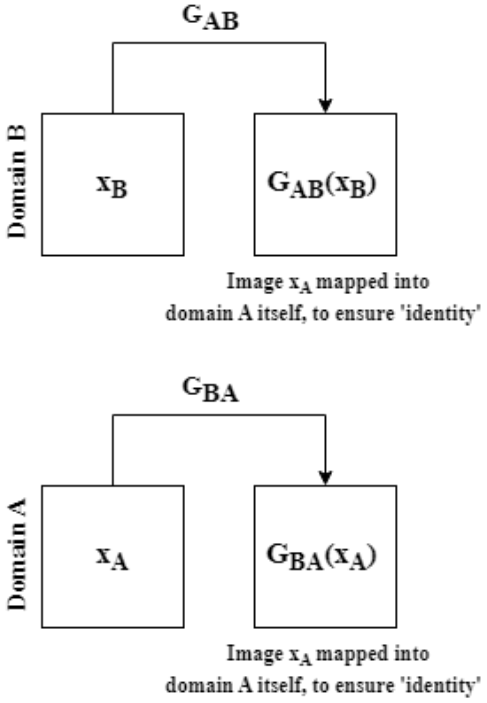

Figure 7. Shows the principle of calculation of identity loss, where an input image is put through a domain that maps images into the input images' domain itself. This is carried out because an accurate model will recognize the domain already matches and return the image unaltered, hence maintaining its 'identity'

The mathematical derivation for Identify Loss, which is also calculated as the L_1 distance, would be:

$$L_{identity} = ||(G_{BA}(x_A)) - x_A||_1 + ||(G_{AB}(x_B)) - x_B||_1 \quad (3)$$

where:

- $L_{identity}$ is the symbol for cycle consistency loss
- x_A is an image from Domain A

- x_B is an image from Domain B
- G_{AB} is the generator that maps domain A to domain B
- G_{BA} is the generator that maps domain B to domain A

Generator Loss:A complete generator loss, which is inferred by the model, through backward propagation to learn further, can be denoted as:

$$L_{Generator} = L_{adv} + (\lambda_{cyc} * L_{cyc}) + (\lambda_{id} * L_{identity}) \quad (4)$$

where:

- $L_{Generator}$ is the symbol for total generator loss
- L_{GAN} is the symbol for adversarial loss
- L_{cyc} is the symbol for cycle consistency loss
- $L_{identity}$ is the symbol for cycle consistency loss
- λ_{cyc} and λ_{id} are weights for cycle-consistency loss and identity loss respectively, which are defined in the hyperparameters.

3.4. Hyper-parameters and Implementation

Zhu et al.[1] introduced several variations for the plausible hyperparameters, while also discussing optimizers used, learning rates, and an image buffer within their implementation. We make use of similar techniques for the re-implementation. The re-implementation was carried out on a Google-Colab Notebook, using the packages within the PyTorch Module in Python. Once the architecture for the generator and discriminator networks were established, and the needed losses as denoted above were defined, Zhu et al.[1] additionally implemented an altered learning rate algorithm, where for the first 100 epochs, the learning rate was set at 0.0002, but for the next 100 epochs, the learning rate was decayed at a rate such that it eventually reaches zero by the end of a total of 200 epochs.

Additionally, to combat cases of over-fitting and improving the overall stability of the training, an image buffer, with the size of 50 images is introduced while training, where instead of using the immediate output from the discriminator, a random image from the set of 50 images is instead utilized to calculate the losses. This ensures more diverse outputs. These specific 50 images are maintained by push and pop mechanics to keep introducing newer images, with predefined probabilities, to maintain an unbiased set of 50 buffer images.

Finally, the complete network was set up in such a way that it kept on saving images for every epoch completion. Thus, for each epoch, for the last batch in that epoch, a set of 16 generated images, and their input images from which the generated images were mapped were saved simultaneously, with synced file names to create pairs and aid the process of evaluation.

Therefore to sum up some crucial hyper-parameters:

- The network was run over 200 epochs whenever possible, but reduced to 100 for larger data sets.
- A learning rate of 0.0002 was used for the first half of the total epochs, with the learning rate gradually reducing to 0 for the last half.
- Finally, for the weights for the identity loss and cycle-consistency loss were set as '5' and '10' respectively.

3.5. Evaluation Metrics

For the evaluation of the implementation, to achieve a comparison against a baseline, the evaluation methods of AMT score and FCN scores were implied from Zhu et al.[1]. But AMT scores were not viable for a re-implementation due to the need for human resources, which would be needed to be outsourced, available through the service of Amazon. So for the paper, only the accuracy calculation through FCN scores under the categories of per-pixel accuracy, per-class accuracy, and class IOU accuracy, based on the scores achieved by Zhu et al.[1] and introduced by Long et al.[7]. The FCN was carried out through the VGG-16 segmentation network, which showed the highest accuracies for Long et al.[7].

After the Cycle-GAN network had completed training, the generated images were tested on the aforementioned evaluation metrics, based on the real images for each of the two domains for all of our different experiments. Furthermore, the final Generator, Discriminator, Cycle-Consistency, Adversarial, and Identity Losses were also noted upon completion of training, to provide a context of how they correlate to actual accuracy, for all of the performed experiments.

4. Experiment

As part of the re-implementation, to verify the diversity of the Cycle-GAN, the network was experimented on various domains, across different types of image-to-image translations. To summarize these attempted experiments, the re-implementation achieved:

- *Van Gogh Paintings↔Photos*
- *Summer Yosemite Images↔Winter Yosemite Images*
- *Mapping Labels↔Photos*
- *Horses↔Zebras*
- *Apples↔Oranges*

Figure 8. Here are the results from the Cycle-GAN for mapping images from Van Gogh-style paintings to photos and vice versa

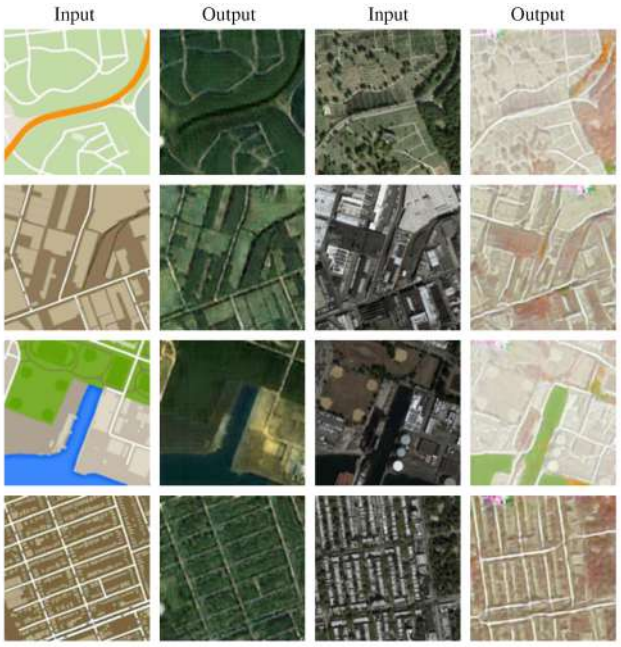

Figure 9. Here are the results from the Cycle-GAN for mapping labels to actual maps, and vice versa

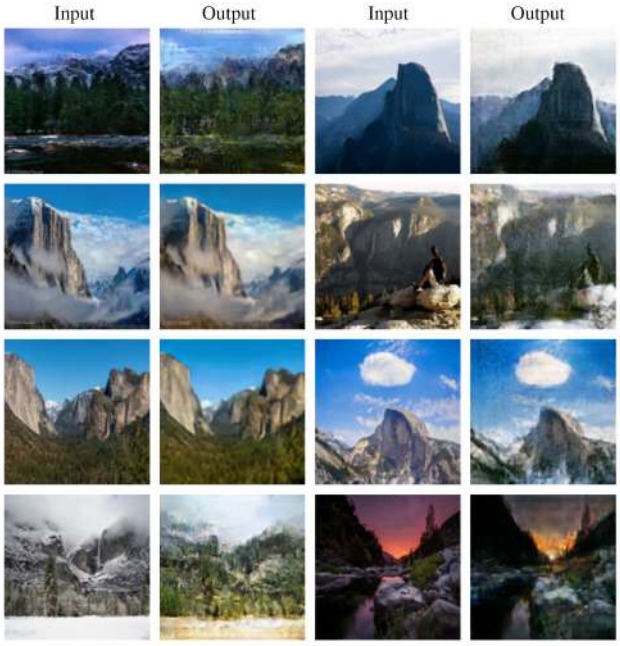

Figure 10. Here are the results from the Cycle-GAN for mapping images of Yosemite in Winter into Summer, and vice versa

Figure 11. Here are the results from the Cycle-GAN for mapping images of oranges to images of apples, and vice versa

Figure 12. Here are the results from the Cycle-GAN for mapping of images zebras to images of horses, and vice versa

For even more results, the Github address to this re-implementation is also posted at the bottom of references, showing more generated images for each of the five experiments, to provide a stronger idea of the range of results achieved.

DATA SET	L_{Gen}	L_{adv}	L_{cyc}	L_{id}
VAN GOGH↔PHOTOS	1.999	0.643	0.103	0.064
SUMMER↔WINTER	2.2982	0.741	0.116	0.079
MAP↔PHOTOS	1.977	0.975	0.070	0.059
APPLES↔ORANGES	2.902	1.257	0.218	0.190
HORSES↔ZEBRAS	2.169	0.735	0.105	0.076

Table 1. Generator/Discriminator Losses held by the Cycle-GAN, at the end of training 200 epochs for each of the 'five' experiments. This provides a brief perspective of how the goal of minimizing the established losses of 'identity loss', 'cycle-consistency loss', and 'adversarial loss' was worked towards through the entire run of the neural network, where one could notice that the experiments with the slightly worse results did end up having higher losses, perhaps signifying this failure in getting perfect translations between the domains.

Evaluation with a Baseline:

As a baseline to evaluate the success of the implementation, the results achieved by Zhu et al.[1], were used, who documented AMT scores for Maps↔Photos, and FCN scores for different variations for Cityscape-Labels↔Photos. While AMT scores were inaccessible for re-implementation due lack of resources to be able to access Amazon Mechanical Turk, this paper compares FCN scores by Zhu et al.[1], with FCN scores of 5 different datasets to provide a perspective of how the re-implementation came through.

Data-set	Per-pixel	Per-class	IOU
Cycle-GAN[1]	0.52	0.17	0.11
VanGogh↔Photos	0.33	0.14	0.06
Summer↔Winter	0.29	0.11	0.03
Map↔Photos	0.41	0.21	0.15
Horses↔Zebras	0.13	0.07	0.001
Apples↔Oranges	0.15	0.01	0.004

Table 2. A comparison for per-pixel, per class accuracies and class IOU, FCN scores between Cycle-GAN[1] on Cityscapes↔Photos and the five different dataset-based experiments carried out for this paper. FCN-VGG16[7] was made use of, to calculate scores for the tables, giving an idea of the level of success achieved by the re-implemented model, while also narrating a relation between different types of image-to-image translations and their success rates.

5. Conclusion

Cycle-GAN does indeed have strong capabilities of achieving unpaired and unsupervised image-to-image translation, that too across various types of translations, such as season transfer, object transfiguration, and photo generation from paintings and vice versa, as proven by the experiments carried out through the re-implementation. The learning based on Adversarial Loss L_{adv}, Cycle-Consistency Loss L_{cyc} and Identity Loss $L_{identity}$, proved suitable, especially for style transfer type of image-to-image translations, earning the 'Cycle' in Cycle-GAN.

Although, it is not all shiny and successful for Cycle-GAN.When working on the datasets from the University of Berkeley[9], running the neural network for each of the five performed experiments took an egregious amount of time, especially for 200 epochs, and altered 18 residual blocks for the Generator Model. Even on Google Colab Pro's, NVIDIA A100-SXM ended up taking several hours of time, which ranged from 12 hours to even multiple days to complete running the network on the decided architecture, which often caused run-time errors, causing restarting the whole network, losing any time so far invested. Furthermore, one can say with overt confidence that with a larger dataset, the accuracy for the re-implementation also increases, which is an aggravating thing when you take in the factor that larger datasets not only require more time to pre-process and pipeline but also ends up increasing the total time required by the model. Thus, during the experimentation for this paper, we found ourselves in a cat-and-mouse chase of trying to increase layers and learnable parameters, with

larger datasets but ending up getting lower accuracies due to the environment or GPU running out of memory. This also made it redundant to achieve high-definition results, from HD datasets, since lower-resolution training already takes a significant amount of resources and time.

Other than hardware bottlenecks, the reimplementation discovered that having a cycle-consistency loss and identity loss definitely struggles to provide strong results for object transfiguration type of image-to-image translation. Horse-to-Zebra translations or Apple-to-Orange translations are examples of object translation, and as seen above, results for them are definitely mixed and far from consistent. Cycle-GAN struggles to map a specific object in images between domains and provides very unrealistic images especially when the input object size and target object size are disproportional. For example, as long as the images have a whole orange, the result images are consistent but for instances where the oranges are cut open or if the coloration of the oranges is different based on different ripeness, the model struggles to define borders around what constitutes an orange and struggles significantly to identifying a cut-orange.

A similar observation is made for Horses-to-Zebras, where the network continuously struggles to define the edges of the animals, especially around the legs when they seem to be in motion. The saving grace here is that horses and zebras have a similar size and structure, but the drastic change of color from black-and-white stripes to brown often takes a toll on the background of the images, with lighter backgrounds looking darker and vice-versa, even if the final constructed image looks realistic.

As for a limitation observed for the season transfer, and paintings-to-photos types of image-to-image translations, some recurring issues, and limitations observed were that these translations often either infused the result images with a blur or speckle noise, which made them clearly distinguishable from the real images upon a human gaze, but offered significantly better translation results when compared to the other types of image-to-image translations, with the final results indeed being convincing a certain degree of having achieved the target domain's characteristics.

Based on the inferences and limitations observed, future work could entail two major plausible improvement techniques. First would be working towards calculating the sweet spots of reducing computational time, while maintaining or even improving the convolution structure chosen for the generator or discriminator networks, perhaps even trying different positions of use or altered use of the generator and discriminator losses, like for example DualGAN[5], which uses the losses in a different configuration.

The other technique focuses specifically on possibly improving object transfiguration type of image-to-image translation. Here, we would focus a bit more on the data-preprocessing aspect of the re-implementation, with the aim of helping the network define the objects better so that the translation focuses primarily on the defined object instead of the complete image. This approach would keep the unpaired datasets principle untouched, and would instead complement it with perhaps a mask or surrounding the object with a 3-D box label, which would highlight the part of the image we want the network to focus on, i.e. the animals for horse-to-zebra and fruits for apple-to-oranges. While this could indeed be successful, we still believe to be apprehensive about the accuracy of datasets with disproportionate-sized objects like cats-to-dogs, which could still be a running problem even with the highlighted objects.

That sums up the re-implementation of Cycle-GAN, learnings observed from the achieved re-implementation, its shortcomings, and their respective possible reasonings, and probable solutions for future implementations, on the work done so far in this paper.

Acknowledgements

The paper acknowledges help from LLM-GrammarlyGO, to help improve the language and grammar of the overall paper, and maintain a strict and formal tone throughout the paper.

References

1. Zhu, J. Y., Park, T., Isola, P., Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2223-2232).

2. Goodfellow, I., Bengio, Y., Courville, A., Bengio, S. (2016). Deep learning (Vol. 1). MIT Press.

3. Isola, P., Zhu, J. Y., Zhou, T., Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5967-5976).

4. A. Radford, A., Metz, L., Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In International Conference on Learning Representations.

5. Yi, Z., Zhang, H., Gong, D., Lai, J. (2017). DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2868-2876).

6. Johnson, J., Alahi, A., Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In European Conference on Computer Vision (pp. 694-711). Springer, Cham.

7. Long, J., Shelhamer, E., Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

8. PyTorch. (n.d.). Tutorials - PyTorch. Retrieved May 5, 2023, from https://pytorch.org/tutorials/

9. Park, T., Liu, M. Y., Wang, T. C., Zhu, J. Y. (2019). Semantic Image Synthesis with Spatially-Adaptive Normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2337-2346).

The GitHub Repository, containing relevant files for Reimplementation: https://github.com/ArnxbSaha/Cycle-GAN-Reimplementation