Project Machine Learning
— Milestone 2 —

Hazem Lahoual, Shashi Durbha, Wai Tang Victor Chan

January 23, 2020

1 Outline

In this milestone, our work is about model selection and evaluation. We start in section 2 by giving an overview of the CycleGAN model followed by a formulation of the problem of unpaired images translation. Section 3 describes the different components of the networks that will be tested. After that, we discuss in section 4 the different evaluation metrics, and explain our choice of the best metric to use. Section 5 describes the large-scale study we performed for the model selection phase. Moreover, we conduct in section 6 two case studies, where our model is validated on the chosen metric and giving good results. Finally, in the last section, we talk briefly about the future work to be done in the next milestone.

2 CycleGAN Overview

CycleGAN is a class of generative adversarial networks for image translation between domains that maintains cycle consistency. Unlike style transfer, where we apply a style image to an image content, or paired image translation, where training data are in pairs of corresponding domains, CycleGAN works on unpaired images from two domains. This section explains what are the improvements added by CycleGAN and formulates the problem of domain translation learning.

2.1 Adversarial Loss

CycleGAN is built on a generative adversarial network (GAN). Thus, We want to train on one hand a generator $G_{X\to Y}$, which transforms elements in domain X to domain Y, where the output should be as similar as possible to those in the target domain. On the other hand, we want to train simultaneously a discriminator D_Y, which can distinguish real elements in Y from generated images (where a score of 1 represents real inputs while a score of 0 represents fake inputs).

The default loss to use is called the adversarial loss, and is expressed using the cross-entropy loss (log loss) as:

$$L_{GAN_X} = E_y[\log D_Y(y)] + E_x[\log(1 - D_Y(G_{X\to Y}(x))].$$

where $x \in X$ and $y \in Y$. The training of the generator involves minimizing this loss, where only the second term is affected by the choice of $G_{X\to Y}$, and the training of a discriminator involves maximizing this loss.

2.2 Cycle-consistency Loss

The adversarial loss does not prevent the generator from generating images with little reference to the input images (the problem is under-constrained). For example, a network that generates a permutation of some memorized training data may be sufficient to obtain a low adversarial loss. The implementation of CycleGAN therefore involves the training of both generators $G_{X\to Y}$ and $G_{Y\to X}$ and makes sure that the output image, supposedly in domain Y, can be reverted to an image, supposedly in domain X, which will be as

similar as possible to the input image. Such property is called cycle consistency and the loss can be expressed by:

$$L_{cycle} = E_x[\|G_{Y\to X}(G_{X\to Y}(x)) - x\|_1] + E_y[\|G_{X\to Y}(G_{Y\to X}(y)) - y\|_1].$$

2.3 Identity Loss

In applications such as photo-to-painting style transfer, we would like the output image to have the same color composition as the input image, and therefore, we need to add another constraint that allows preserving the color distribution. The idea behind identity loss is that if we use an image x from domain X as an input to the generator $G_{Y\to X}$, the output should be identical to x. The following formula expresses the formulation of the identity loss:

$$L_{identity} = E_x[\|G_{Y\to X}(x) - x\|_1] + E_y[\|G_{X\to Y}(y) - y\|_1].$$

2.4 Problem Formulation

The training of the generative network is equivalent to finding (fine-tuning parameters for) two generators $G_{X\to Y}$ and $G_{Y\to X}$ which minimize the following maximization problem:

$$L = \max_{D_X, D_Y} L_{GAN_X} + L_{GAN_Y} + \lambda(L_{cycle} + \gamma L_{identity}).$$

The weight λ controls the balance between the adversarial loss and the additional loss for CycleGAN (cycle-consistency loss and identity loss).

Since the identity loss is less important than the cycle-consistency loss, we assign $\gamma = 0.5$ as the default value for all models (as in Zhu et al. (2017)). In fact, a small change of the color composition can be tolerated, as long as the outputs are similar to the images in the target domain. We will discuss in section 6 the importance of λ and γ for two different tasks, namely image transfiguration (change of appearance without change of shape) and image style transfer (photo to painting and painting to photo).

3 Model Selection

In this section, we give an overview of the different components of the CycleGAN, which will be evaluated in section 5.

3.1 Loss Function

Generally for working with GANs, there are two types of loss functions to consider, namely, logarithmic (log) loss and least-squares loss. Both functions are commonly used in related research works. They can be expressed as the following:

- **Log Loss:** $L_{GAN_X} = E_y[\log D_Y(y)] + E_x[\log(1 - D_Y(G_{X\to Y}(x))]$
- **Least-Squares Loss:** $L_{GAN_X} = E_y[D_Y(y)^2] + E_x[(1 - D_Y(G_{X\to Y}(x))^2]$

Our model selection process will involve the comparison of the performance of the networks trained under these two loss functions.

3.2 Discriminator Normalization

We want to test different types of normalization for the discriminator to see how it affects the model. In the experiments, we limit our choice to only two types of normalization: batch normalization and instance normalization. The former normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. However, in the case of instance normalization, this operation is done across each channel in each training example (instance).

3.3 Networks Architecture

In this section, we describe the architecture of the networks (such as the depth of the network and the length of the residual layers) that we will build and test in 5.4. We will use the same naming convention as used in the original paper (Zhu et al. (2017)).

3.3.1 Generator Architecture

Let $c7s1\text{-}k$ denotes a 7×7 Convolution-InstanceNorm-ReLU layer with k filters and stride 1. dk denotes a 3×3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. Reflection padding is used to reduce artifacts. Rk denotes a residual block that contains two 3×3 convolutional layers with the same number of filters on both layers. uk denotes a 3×3 fractional-strided-Convolution-InstanceNorm-ReLU layer with k filters and stride $\frac{1}{2}$. Table 1 summarizes the architectures based on ResNet.

network	architecture
9-blocks resnet	c7s1-64, d128, d256, (R256)×9, u128, u64, c7s1-3, tanh
6-blocks resnet	c7s1-64, d128, d256, (R256)×6, u128, u64, c7s1-3, tanh

Table 1: Architecture of the generators based on ResNet

Moreover, we test two types of generators based on Unet architecture (for the description of the architecture of Unet, we refer the reader to Ronneberger et al. (2015)): Unet 256 and Unet 128. The difference between Unet 256 and Unet 128 is that the former has one more step of downsampling and upsamling (input size for Unet 256 is 256×256 px, while the input size for Unet 128 is 128×128 px).

3.3.2 Discriminator Architecture

We use as a discriminator a 70×70 PatchGAN, which gives for each 70×70 overlapping input image patches a probability that they are real or fake.

We denote by Ck a 4×4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. Instance normalization is not used in the first $C64$ layer. We apply a convolution after the last layer to produce a 1-dimensional output. The slope of LeakyReLU is 0.2. The discriminator architecture is then: $C64\text{-}C128\text{-}C256\text{-}C512$. The dimensions of the output are [Batch size, number of channels = 1 , 30, 30]

4 Evaluation Metrics

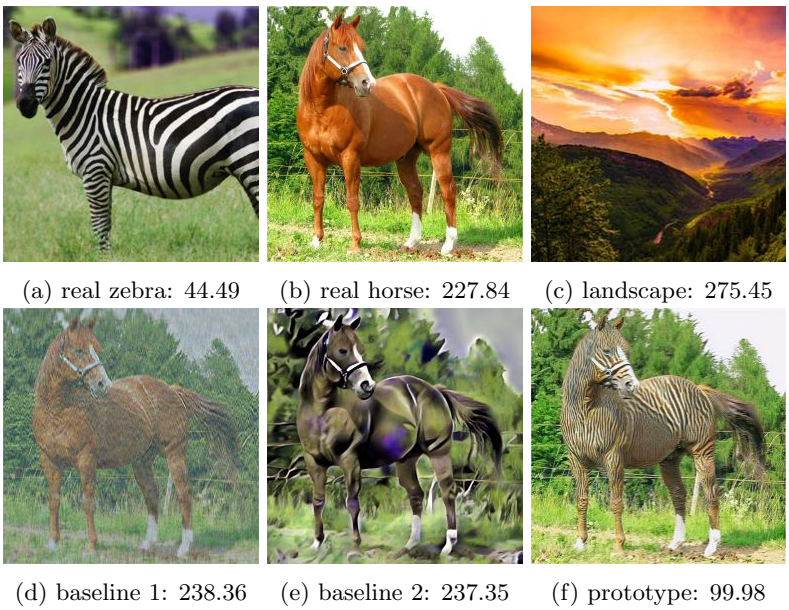
To perform model selection or parameters tuning, one needs to assess the quality of a model, trained with a specified learning rate, model architecture and hyperparameter, by evaluating the quality of the output images. We describe the qualitative and quantitative evaluation metrics, and which one will be used for our experiments.

4.1 Qualitative Evaluation

As our model generates images, the output can be evaluated qualitatively, that means a user can judge directly the performance of our generator. But, the qualitative evaluation is very expensive in practice. For a large dataset and with many classes, it is not possible for a human to check every image manually and assess its quality. Also, the assessment is impartial and subjective, and changes from one person to another, so it's not a good metric to rely on when evaluating GANs. This is why we look for an automated way to evaluate the generator performance.

4.2 Quantitative Evaluation

To evaluate GANs automatically, we wish to make use of metrics that agree with the perceptual judgments of human beings. Borji (2018) gives an overview and a benchmark of different metrics used in literature. We will only consider three evaluation metrics, which are the most used in recent papers about GANs.



(a) real zebra: 44.49 (b) real horse: 227.84 (c) landscape: 275.45

(d) baseline 1: 238.36 (e) baseline 2: 237.35 (f) prototype: 99.98

Figure 1: FID of different images compared to a set of real zebra images.

4.2.1 Inception Score

Proposed by Salimans et al. (2016), Inception Score (IS) is a metric that evaluates the quality of the generated images. To compute the score, the generated images are fed to a pre-trained Inception Classifier. The probabilistic distributions obtained will then be assessed to produce a score that reflects (1) how well identifiable or easily classifiable the individual image is (conditional label distribution has low entropy $p(y \mid x)$, where x is the generated image and y is the predicted label) and (2) how diverse the generated images are (marginal label distribution has high entropy $p(y)$). A high score reflects a high quality of the assessed GAN model. The IS is computed using the following formula:

$$IS = \exp\left(\mathbb{E}_{x \sim Q}[D_{KL}(p(y \mid x) \mid\mid p(y))]\right).$$

Barratt and Sharma (2018) discuss a few drawbacks of the Inception Score. It is noted that the score, based on an Inception Classifier trained on ImageNet, does not properly assess GAN models trained on alternative image sets. Optimization with such metric also produces models that tend towards adversarial examples and often generates unnatural images that achieve good scores.

4.2.2 Fréchet Inception Distance

One main drawback of Inception Score (IS) is that it does not take into account the real images to evaluate the performance of the GAN. As an alternative, the "Fréchet Inception Distance" (FID), proposed by Heusel et al. (2017), solves this problem by defining a similarity measure between generated and real images.

The process is the following: Samples from domain X (generated images) and domain Y (real images) are first embedded into a feature space using a specific layer of InceptionNet, which is a pretrained classification model on ImageNet. Then, assuming that the embedded data follow a multivariate Gaussian distribution, the mean and covariance are estimated. Finally, the Fréchet distance between these two Gaussians is computed using the following formula:

$$FID = \|\mu_X - \mu_y\|_2^2 + Tr(\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{\frac{1}{2}}).$$

where (μ_X, Σ_X) and (μ_Y, Σ_Y) are the mean and covariance of the embedded samples from X and Y. The FID is consistent with human judgment and robust to noise. The lower the score, the more the outputs of the generator are similar to the real images.

To justify the use of FID as an evaluation metric, we computed the FID of several images, shown in Figure 1, reflecting how far each image is from a set of genuine zebra images. As expected, a real zebra image has the smallest distance, while the image that is

semantically the most different (a landscape photo) corresponds to the largest distance. We also see that FID better reflects the quality of the models in Milestone 1 (Figures 1d, 1e and 1f), than the score from a discriminator did.

4.2.3 Kernel Inception Distance

The Kernel Inception Distance (KID) was proposed by Bińkowski et al. (2018) as an alternative metric to FID that removes the unpredictable biasedness of FID.

Kurach et al. (2018) and Lucic et al. (2017) show that KID and FID are strongly correlated, with Spearmann rank-order correlation coefficient of over 0.9 (where the coefficient value lies between 1 and -1: 1 means correlation, 0 no correlation and -1 inverse correlation). So, for the sake of simplicity, we will limit ourselves to only FID score, as the results found using it will be the same as those obtained with KID score.

5 Ablation Study

In this section, we describe first how the experiments are done, then we show the results obtained from different models. We explain then our model selection choices.

5.1 Experiments Description

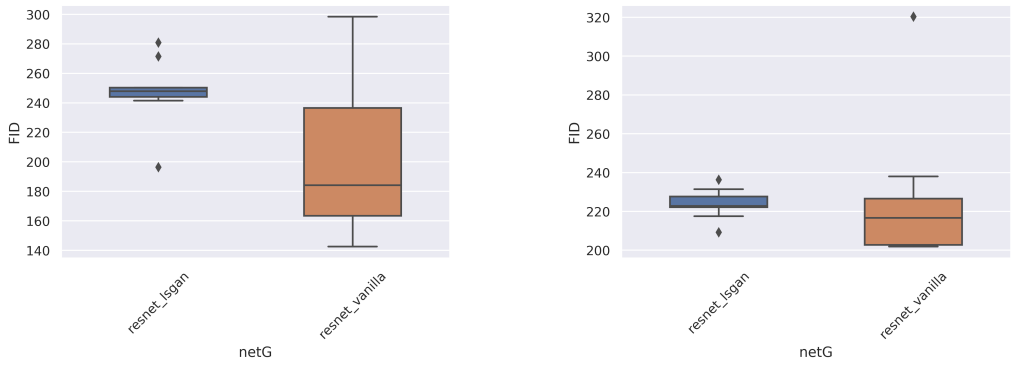
Due to resources limitations, we can not test all possible combinations of generator architectures, losses and discriminator normalization, that's why we will only test a limited choice of them. We conduct an ablation study: each time, we fix all components mentioned above except one. Then, we train different models using different types of the tested component, and using the same set of hyperparameters combinations (using different hyperparameters and not only one specific combination will allow us to have a fair comparison between models. In fact, these experiments will test the stability of the models, which is a desirable property for the GANs). After that, we compute their FID score and find the best type of component to use, which is the one that is giving in average lower FID with small variance (more stable).

For all models, we will use Adam optimizer with batch size 4, so the hyperparameters are the learning rate α, the momentum decaying rate β_1 and scaling decaying rate β_2. Table 2 contains the set of hyperparameters chosen for the training phase. We decided to test the most used values in the recent works about GANs (Kurach et al. (2018)), as the choice is narrowed by the researchers to a small set of hyperparameters. But, there is still no agreement on the best choice, because the training of GANs is sensitive to any change: the random seeds for the initialization of the network, the dataset used for the training, the number of batches, etc. The triplet J is chosen to compare the behavior of different models using an aggressive learning rate.

index	α	β_1	β_2
A	0.0002	0.5	0.9
B	0.0002	0.5	0.999
C	0.0002	0.9	0.999
D	0.0001	0.5	0.9
E	0.0001	0.5	0.999
F	0.0001	0.9	0.999
G	0.001	0.5	0.9
H	0.001	0.5	0.999
I	0.001	0.9	0.999
J	0.1	0.9	0.999

Table 2: The hyperparameters combinations used for the training

As an example of the experiments to make things clear: we fix the architecture of the generator, the normalization type for the discriminator, and test two different losses. For this, we train two models: one with log loss and one with least-squares loss. Each model will be trained 10 times, each time with one combination of α, β_1 and β_2 (see table 2 for all possible combinations of hyperparameters). We compute the FID score from these



(a) horse to zebra translation

(b) zebra to horse translation

Figure 2: FID scores distributions of the models resnet_lsgan and resnet_vanilla

experiments (10 FID scores for each type of model), and we plot the boxplots of these scores to have an overview of the models performances (the best model will have a lower median and smaller variance of the FID scores).

The dataset horse2zebra is used for the training. The preprocessing pipeline is the following: resizing to 283×283 px, random cropping of size 256×256 px, random horizontal flip and normalizing to [-1,1] range (see milestone 1 for more details about dataset and pipeline).

Since we want to train a large number of models (70 models), we decide to limit the number of the training set to only 64 images for each domain. This small dataset will not give good outputs, but, at this step, we want to have an idea on the best combination of losses, normalizations, and architectures. So, by training different models using the same set of hyperparameters and the same size of dataset, we will have a fair benchmark on their performance. In the next section, we will train our best model using the full dataset, and evaluate its performance again with the best triplet of the hyperparameters (to be decided in section 5.5).

We train the models for 200 epochs, with linear weight decay to 0 starting from epoch 100. Furthermore, to reduce the risk of mode collapse, we follow the same strategy as in Zhu et al. (2017): we use a buffer to store the last 50 generated images by the generator, and we update the discriminator using a randomly chosen image from this buffer, instead of using the last generated one.

We chose 120 images as the test set for each domain. Therefore, giving the fact that we have two domain A and B, and a generator that translates images from domain A to domain B: The computation of FID score uses 120 generated images (translated test images from domain A) and 120 real images (test images from domain B).

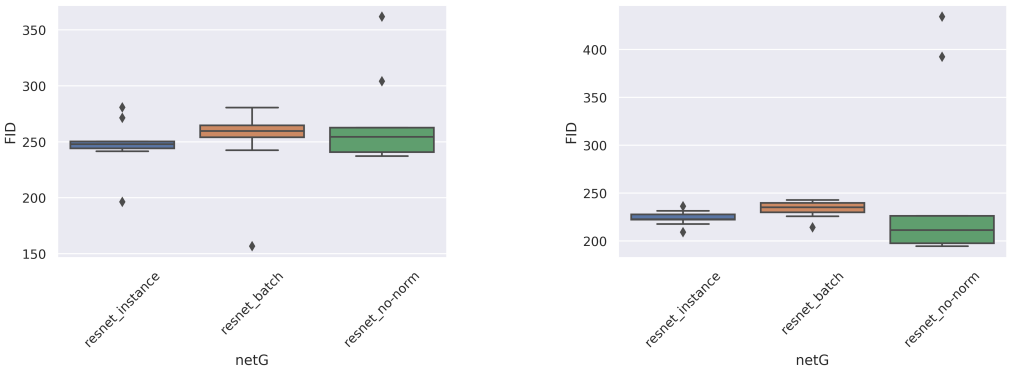
The networks trained with the combination J are always diverging (high FID score). Therefore, to keep the visualization clear, we decide to not include it in the results of sections 5.2, 5.3 and 5.4.

5.2 Loss Function

Two models are trained with a 9-blocks resnet generator, patchGAN discriminator and instance normalization using two losses: log loss (resnet_vanilla) and least-squares loss (resnet_lsgan). Figure 2 shows the FID scores obtained after training the models for the given hyperparameters in table 2. In figure 2a (zebras generation), we can see that the mdoel resnet_lsgan has bigger variance (which means less stability), but almost all the FID scores are lower than the average of resnet_lsgan scores. In figure 2b (horses generation), the resnet_vanilla model still gives better results, but this time the performances are quite similar. So, we will use log loss for our final model.

5.3 Discriminator Normalization

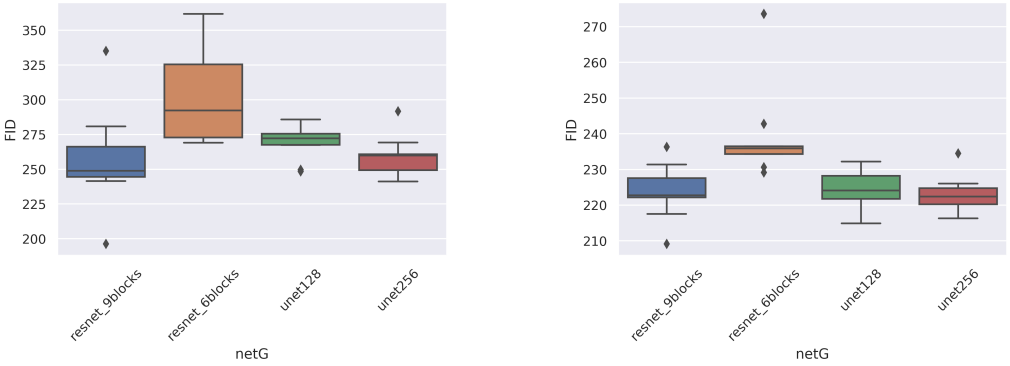
We train three models with a 9-blocks resnet generator, patchGAN discriminator and least-squares loss using three different types of normalization for the discriminator: no normalization (resnet_no-norm), instance normalization (resnet_instance) and batch nor-



(a) horse to zebra translation

(b) zebra to horse translation

Figure 3: FID scores distributions of the models resnet_instance, resnet_batch and resnet_no norm



(a) horse to zebra translation

(b) zebra to horse translation

Figure 4: FID scores distributions of the models resnet_9blocks, resnet_6blocks, unet256 and unet128

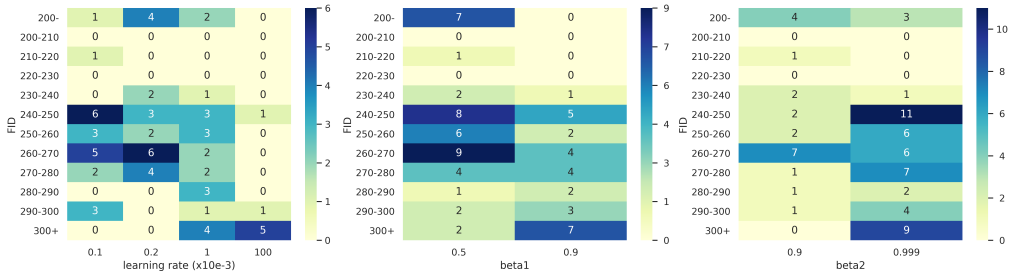
malization (resnet_batch). Figure 3 shows the results of the experiments. For the case of zebras generation (figure 3a), resnet_instance and resnet_no-norm are giving similar results, but the latter with bigger variance and outliers far from the mean (unstable model). The same thing happens in the case of horses generation (figure 3b), where resnet_no-norm is giving slightly better results over resnet_instance, but with more outliers and bigger variance. Resnet_batch is giving the worst results in both cases. Thus, we will choose the instance normalization for our final model.

5.4 Generator Architecture

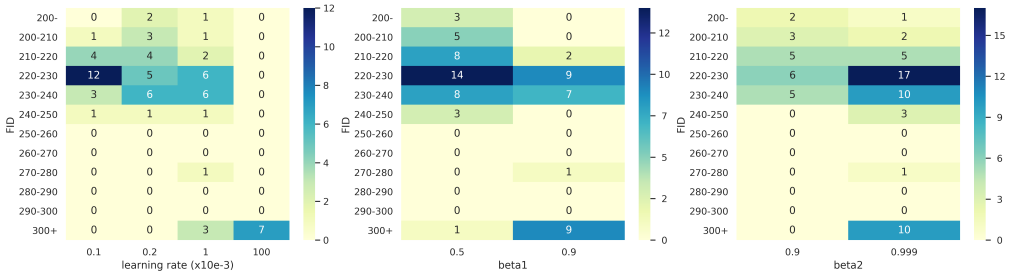
Here, four models with patchGAN discriminator, instance normalization and least-squares loss are compared using four different architectures for the generator: 9-blocks resnet (resnet_9blocks), Unet 256 (unet256), 6-blocks resnet (resnet_9blocks) and Unet 128 (unet128). For the last two models, we resize the input to 128×128 px. Figure For the case of zebras generation (figure 4a), resnet_9blocks and unet256 are giving the best results, with the former having a bigger variance (less stable). Also the same for the horses generation (figure 4b), where resnet_9blocks, unet256 and unet128 are giving similar FID scores. In both cases, resnet_6blocks is the worst performer. We will choose then resnet_9blocks as it has less parameters (11×10^6) compared to unet256 (54×10^6). 4 shows the results.

5.5 Hyperparameters

For the selection of the hyperparameters, we plot in figure 5 the heatmaps of the learning rate α, β_1 and β_2, using the results from the trained models above (70 models). The value of the hyperparameter that is giving the largest number of models with a lower



(a) horse to zebra translation



(b) zebra to horse translation

Figure 5: heatmaps of the learning rate α, momentum decaying rate β_1 and scaling decaying rate β_2. The number of models is also shown.

FID score is the one to be chosen.

The learning rate 0.0001 and 0.0002 are giving similar results, with 0.0002 having more models with a score lower than 200. With 0.1 as learning rate, we have the worst results, as the models are almost always diverging. Thus, our choice for the learning rate is 0.0002.

In case of β_1, we can see clearly that 0.5 is better than 0.9, that means we have more models with lower FID score using 0.5. Finally, for β_2, the results are not decisive about which rate is better (if we remove the results of the seven models trained with the hyperparameters triplet J), so more experiments should be done. For our final model, we will use $\beta_2 = 0.999$ as it is the most used value in the recent papers.

6 Case Studies

Having chosen the architecture of the CycleGAN and the hyperparameters, we train the model using the full dataset this time: 9-blocks resnet with instance normalization, first with least-squares loss (res9-inst-ls) and log loss (res9-inst-vn) to compare them again (as we want to check the claim of the original implementation, which says that the log loss is not performing better than the least-squares loss), and second, using different values of λ and γ.

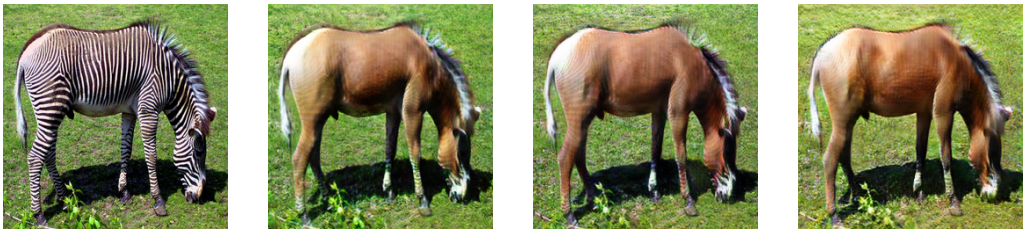
index	model	α ($\times 10^{-3}$)	β_1	β_2	λ	γ	FID H2Z	FID Z2H
M1	res9-inst-ls	0.2	0.5	0.999	10	0.5	77.2	135.72
M2	res9-inst-vn	0.2	0.5	0.999	10	0.5	**56.75**	140.25
M3	res9-inst-vn	0.2	0.5	0.999	5	0.5	59.05	**134.59**
M4	res9-inst-vn	0.2	0.5	0.999	0	0.5	379.80	159.62

Table 3: FID scores of the models trained using the complete dataset. Different losses and λ are tested.

Table 3 summarizes the results using the horse2zebra dataset. The experiments support our choice in section 5.2, as the model with log loss outperforms least-squares loss in the case of horse to zebra translation (H2Z), and the two models have similar FID score in the case of zebras to horses translation (Z2H). Figures 6 and 7 show samples from the test set used for generating the scores. Here, we can see clearly that the model

(a) real horse (b) M1 (77.2) (c) M2 (56.75) (d) M3 (59.05)

Figure 6: Example of H2Z generator output of the best three models.



(a) real zebra (b) M1 (135.72) (c) M2 (140.25) (d) M3 (134.59)

Figure 7: Example of Z2H generator output of the best three models.

with the lowest FID score is the model generating the best outputs.

We also test the art style transfer task using landscape2monet dataset. Table 4 contains the FID score of different trained models.

index	model	α ($\times 10^{-3}$)	β_1	β_2	λ	γ	FID M2P	FID P2M
P1	res9-inst-ls	0.2	0.5	0.999	10	0.5	**122.20**	146.68
P2	res9-inst-vn	0.2	0.5	0.999	10	0.5	125.45	146.89
P3	res9-inst-vn	0.2	0.5	0.999	10	0	135.75	**140.98**
P4	res9-inst-vn	0.2	0.5	0.999	5	0.5	131.03	145.44

Table 4: FID scores of the models trained using the complete dataset. Different losses, λ and γ are tested.

For Monet to photo (M2P), adding the identity loss gives more realistic images. Figure 8 shows also that using least-squares loss is giving better results. However, in the photo to Monet translation (P2M), the model without identity loss gives better outputs (figure 9). This is because the models with the identity loss try to keep the color of the input, and don't focus on the artistic style of the output. This is not the case of the model without identity loss, which focuses on generating images as similar to Monet painting as possible, even with changes to input color distribution.

7 Conclusion

We discussed the CycleGAN model and what are the best architectures, losses and normalizations to use through a large-scale study. We then conducted a case study of two different tasks, namely image transfiguration and art style transfer, and discussed the results. We found that log loss is outperforming the least-squares loss proposed in the original implementation. Moreover, removing the identity loss in the case of photo to painting translation is giving better results. In the next step, we will study the model limitations and failure cases, and what are the proposed improvements proposed in the literature. Also, we will see the drawbacks of the FID score as a metric evaluation, and how precision and recall can be a better alternative to it. Finally, we will extend our model to real-life applications.

(a) real monet (b) P1 (122.20) (c) P2 (125.45) (d) P4 (131.03)

Figure 8: Example of M2P generator output of the best three models.

(a) real photo (b) P1 (146.68) (c) P3 (140.98) (d) P4 (145.44)

Figure 9: Example of P2M generator output of the best three models.

References

S. Barratt and R. Sharma. A Note on the Inception Score. *arXiv e-prints*, art. arXiv:1801.01973, Jan 2018.

M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD GANs. *arXiv e-prints*, art. arXiv:1801.01401, Jan 2018.

A. Borji. Pros and Cons of GAN Evaluation Measures. *arXiv e-prints*, art. arXiv:1802.03446, Feb 2018.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30*, pages 6626–6637. Curran Associates, Inc., 2017.

K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly. A Large-Scale Study on Regularization and Normalization in GANs. *arXiv e-prints*, art. arXiv:1807.04720, Jul 2018.

M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? a large-scale study, 2017.

O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.