Generative Adversarial Network on Style Imitation

Rémy Sabathier
CentraleSupélec
Paris-Saclay University

Omar Souaïdi
CentraleSupélec
ENS Paris-Saclay (MVA)

Abstract

Generative Adversarial Networks (called GANs) is a type of deep learning models introduced by Ian Goodfellow and other researchers in 2014. In this type of models, two neural networks are competing against each other's in order to learn how to generate samples with the same statistics as the original distribution. On its early release, GAN were identified as highly unstable during the training (because of the competition between the two networks) and hard to evaluate. These two issues are still active domain in Artificial Intelligence: we propose to study the key aspects of these two problems by participating to an open competition hosted on Kaggle: 'I'm something of a painter myself'.

1. Introduction

The competition is a style transfer challenge based on Monnet paintings. As such, given a dataset of real-life images, we are asked to design a GAN that can give the style of Monnet paintings to an image of our choice. After implementing a model, we can send our proposition to measure the performance of our network compared to other participants (as we are writing this description, 221 teams are enrolled). A constraint on the training time of the model is imposed (less than 3 hours training time on TPUs or 5 hours on GPUs).

Evaluation In generative adversarial learning, evaluating a model is not as simple as in classification for instance where the progression of a model can be easily quantified witch metrics. Evaluating GANs performance is still considered as active research subject. This project is also motivated on understanding the current most used metrics for GAN evaluation (Fréchet Inception Distance [2], Inception Score [3]). The Kaggle competition propose to evaluate candidate on the MiFID: memorization informed Fréchet Inception Distance, which is a modified version of the Fréchet Inception Distance which takes training set memorization into account.

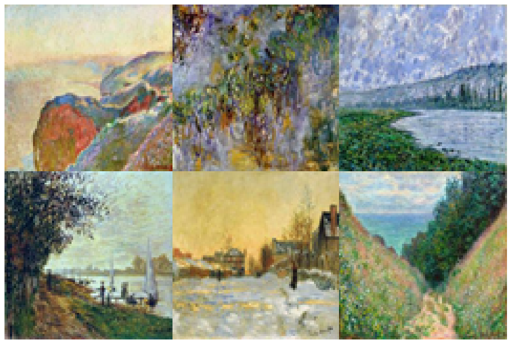

Figure 1. Example from the Monnet painting training set. (256x256x3) in TF records format.

Dataset Computing Units To train our GAN, the competition proposes a collection of 300 Monet paintings. We then have 7363 images that can be used to test the style transfer. One interesting auxiliary aspect of the competition is the use of the TF records format for the dataset and TPU processors for training.

2. Evaluation

Evaluate numerically the results of a generative model is not as straight-forward as classical neural networks performance evaluation where we have metrics (for classification or regression task for instance). In generative models, the objective of the **generator** is to learn how to generate, from a random latent vector z, a sample belonging to a target distribution. However, in the majority of cases, the target distribution is not well defined. In the competition context, it is Monet painting images. We analyzed two different metrics to understand how evaluation is performed in this context.

2.1. Inception Score

The *Inception Score* was introduced in [5] as a first metric that correlates with *human judgement* for GAN evaluation. They start by applying the Inception model on the generated images to get a label distribution y for each sample x. The loss is then based on the two principles:

1. The conditional probability $p(y|x)$ should have a **high entropy** since we want to generate realistic image that triggers only one label.
2. The sum of all conditional probability $\sum_i p(y_i|x_i)$ (called marginal probability) should have a **low entropy** since we want to generate diverse samples.

The inception score then computes the KL-Divergence between the marginal probability and each conditional probability and takes the exponential of the average. According to the 2 mentioned principles, if the GAN generates diverse and realistic samples, we should expect a very high score.

A first limit of the inception score is that it is only applicable on the training set used for the training of the inception network, which is *ImageNet*. As well, the score doesn't use the training images at all for evaluating how well the GAN is able to generate fake samples.

2.2. Frechet Inception Distance (FID)

In [3], the authors tries to address the inception scores issues by proposing the **Frechet Inception Distance (FID)**. Like the inception score, the method is based on the pretrained *Inception* model, but they use the output of an intermediate layer with *2048* features. For each training samples and a large batch of fake samples, they compute the activation features and approximate the distribution for fake and real by a multivariate gaussian $\mathcal{N}(\mu_{real}, \Sigma_{real})$ and $\mathcal{N}(\mu_{fake}, \Sigma_{fake})$. The score is then the *Frechet Distance* between the two distributions:

$$||\mu_{real} - \mu_{fake}||^2 + Tr(\Sigma_{real} + \Sigma_{fake} - 2\sqrt{\Sigma_{real}\Sigma_{fake}})$$

The FID score better correlates with the quality and diversity of fake images and it can also be used on GAN that are not trained on *ImageNet* dataset (as long as the features detected by Inception intermediate layers are meaningful on this training set).

2.3. Memorized FID (Mi-FID)

The main issue of the previous methods for evaluating GAN performance is the fact models can easily achieve strong performance by memorizing a small subset of the target distribution. To penalize the performance of these models, [2] introduced the **Memorization-Informed Frechet Inception Distance (MiFID)**. The score between the fake distribution $\mathcal{D}_f$ and the real distribution $\mathcal{D}_r$ is computed by

$$MiFID(\mathcal{D}_f, \mathcal{D}_r) = m_r(\mathcal{D}_f, \mathcal{D}_r).FID(\mathcal{D}_f, \mathcal{D}_r)$$
$$m_r(\mathcal{D}_f, \mathcal{D}_r) = \frac{1}{s(\mathcal{D}_f, \mathcal{D}_r) + \epsilon}$$

where m_r is the memorization penalty, and **s** is defined as 1 minus the average cosine similarity between a fake sample s_f and the full training set. As such, models that memorize the training set will have a strong memorization penalty. The MiFID is the metric used to evaluate models in the *Kaggle* competition we attended.

3. Models

We decided to focus only on generative model, which are specifically designed for the competition. Generative models are generally difficult to train due to the inherent instability of adversarial training. The *Kaggle* competition we attended is based on Monet painting generation from real life images, which is an **image-to-image translation** task on an *unpaired dataset*.

3.1. Cycle-GAN

Cycle GAN [7] is a standard generative model for image-to-image problems on unpaired dataset. In the original paper, Monet painting generation from standard image is presented as a application of their method.

Objective Let's consider P and M, the domain space associated to the real photos and the Monet paintings. The model includes two *Generators* G_P and G_M that maps a input to the domain P or M respectively. As an adversarial model, we also includes two discriminators D_P and D_M. The objective of D_M is to distinguish between real Monet painting and fake samples generated from G_M, and the objective of D_P is to distinguish between real pictures and fakes from G_P.

Loss is divided in 3 parts:

1. P **Adversarial Loss**. G_M tries to generate realistic fake Monet painting from picture input, whereas D_M tries to make the difference between real Monet m and fake ones $G_M(p)$:

$$L_{adv1} = \mathbb{E}_{p\sim P}[1 - log(D_M(G_M(p)))] + \mathbb{E}_{m\sim M}[log(D_M(m))]$$

2. M **Adversarial Loss**. Same, with real picture generation:

$$L_{adv2} = \mathbb{E}_{m\sim M}[1 - log(D_P(G_P(m)))] + \mathbb{E}_{p\sim P}[log(D_P(p))]$$

3. **Cycle consistency loss**. Mainly for regularization purposes, they force the generator to be able to reconstruct

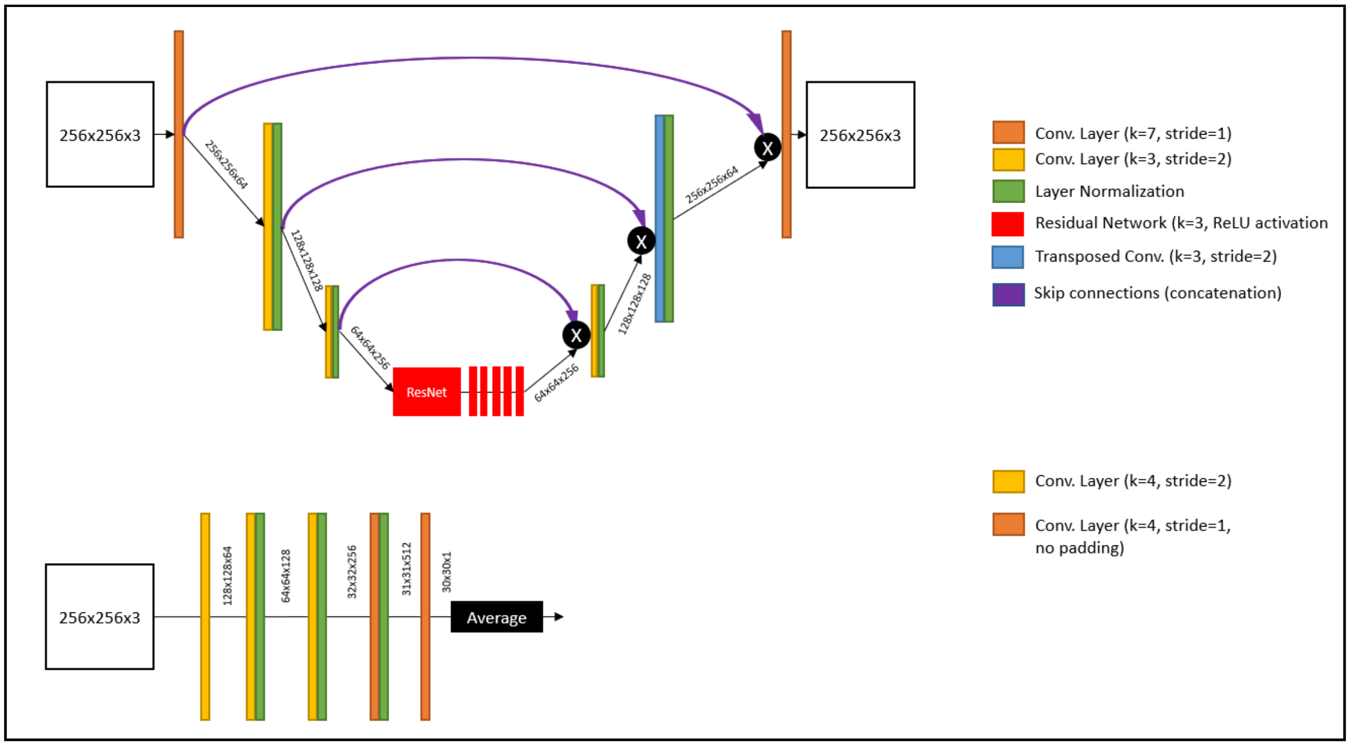
256x256x3
256x256x64
128x128x128
64x64x256
ResNet
64x64x256
X
128x128x128
X
256x256x64
X
256x256x3
Conv. Layer (k=7, stride=1)
Conv. Layer (k=3, stride=2)
Layer Normalization
Residual Network (k=3, ReLU activation
Transposed Conv. (k=3, stride=2)
Skip connections (concatenation)
256x256x3
128x128x64
64x64x128
32x32x256
31x31x512
30x30x1
Average
Conv. Layer (k=4, stride=2)
Conv. Layer (k=4, stride=1, no padding)

Figure 2. Cycle-Gan Architecture selected. top: Generator. bottom: Discriminator

an image, on the same principle that for **autoencoder** training:

$$L_{cycle} = \mathbb{E}_{m\sim M}[||G_M(G_P(m)) - m||] + \mathbb{E}_{p\sim P}[||G_P(G_M(p)) - p||]$$

The final loss is the sum of these three elements, with generally a λ coefficient applied on the cycle term.

Architecture In the original paper, the authors used for the generators a convolutional network with residual layers in the bottleneck (Fig.5). We add **skip connections** to the original implementation to achieve a better spatial reconstruction of the image and stabilize the gradient propagation during training. For the decoder, we keep the original architecture as well which is a simple sequential convolutional network with 5 stacked convolutions and a *2D average pooling* at the end. Layer normalization [1] is added after convolutional layers (except input and output).

3.2. Bayesian Cycle-Gan

To solve the stability issue in Cycle-GAN, Bayesian Cycle-GAN [6] were proposed. They introduce a latent space for exploring the full posteriors. Specifically, they combine source images with corresponding certain volume of sampled latent variables to construct variant-concatenate inputs, transferring network sampling to input domain sampling. By exploring full posteriors, Bayesian CycleGAN enhances generator training to resist crazy learning discriminator, and therefore alleviates the risk of mode collapse, boosting realistic color generating and multimodal distribution learning. The proposed Bayesian CycleGAN models the true data distribution more accurately by fully representing the posterior distribution over the parameters of both generator and discriminator. More details about the model can be found in the original paper [6].

3.3. U-GAT-IT

U-GAT-IT (Unsupervised Generative Attentional Networks With Adaptive Layer-Instance Normalization for Image-to-Image Translation) [4] is a novel method for unsupervised image-to-image translation, which incorporates a new attention module and a new learnable normalization function in an end-to-end manner.

This model guides the translation [Source Image (Normal Picture) to Target Image (Monet Painting)] to focus on more important regions and ignore minor regions by distinguishing between source and target domains based on the attention map obtained by the auxiliary classifier. These attention maps (figure 3) are embedded into the generator and discriminator to focus on semantically important areas, thus facilitating the shape transformation. While the atten-

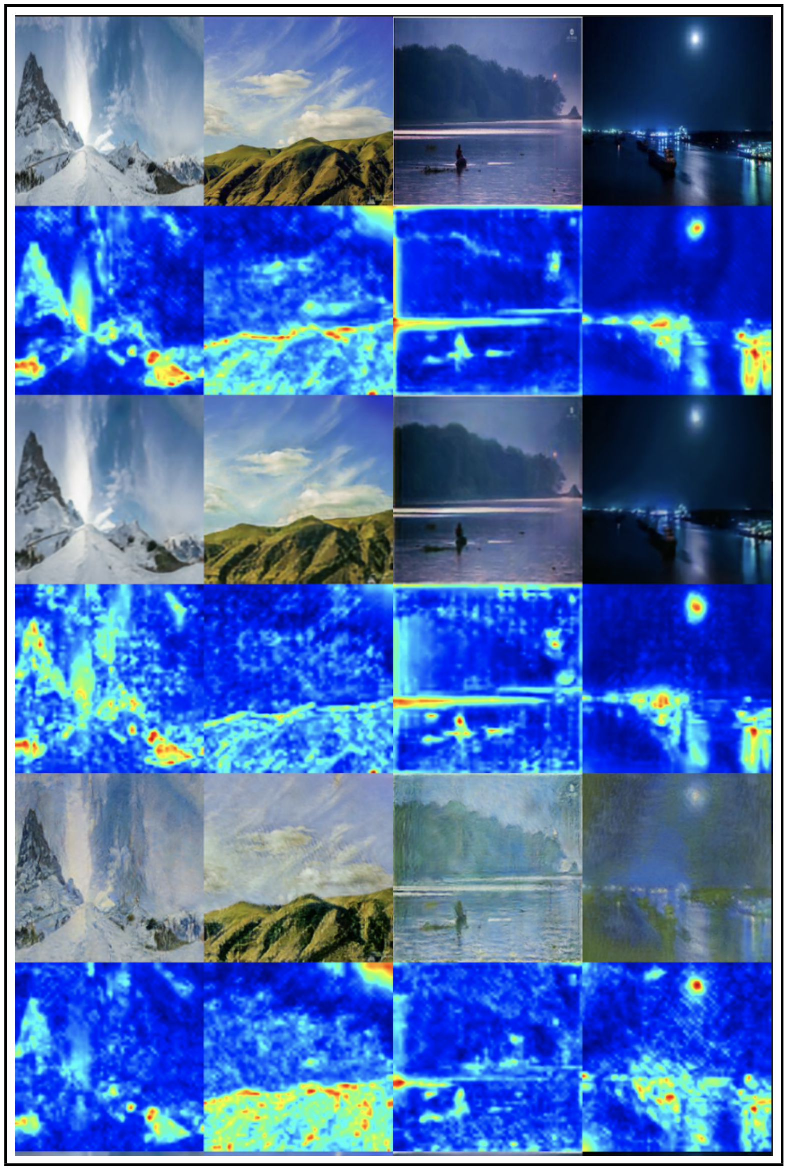

Figure 3. Visualization of the attention maps for 4 differents normal pictures. The first one (second row) represent the attention map related to the identity A2A. The fourth one to the translation A2B and the sixth one to the cycle A2B2A

tion map in the generator induces the focus on areas that specifically distinguish between the two domains, the attention map in the discriminator helps fine-tuning by focusing on the difference between real image and fake image in target domain.

In addition to the attentional mechanism, they find that the choice of the normalization function has a significant impact on the quality of the transformed results. They propose an Adaptive LayerInstance Normalization (AdaLIN), whose parameters are learned from datasets during training time by adaptively selecting a proper ratio between Instance normalization (IN) and Layer Normalization (LN). The AdaLIN function helps their attention-guided model to flexibly control the amount of change in shape and texture. As a result, the model, without modifying the model architecture or the hyper-parameters, can perform image translation tasks not only requiring holistic changes but also requiring large shape changes.

3.3.1 Architecture

Description is mainly taken from the original paper [4]. The architecture of the model can be seen in figure 4.

Generator Let $x \in \{X_s, X_t\}$ represent a sample from the source and the target domain. The translation model

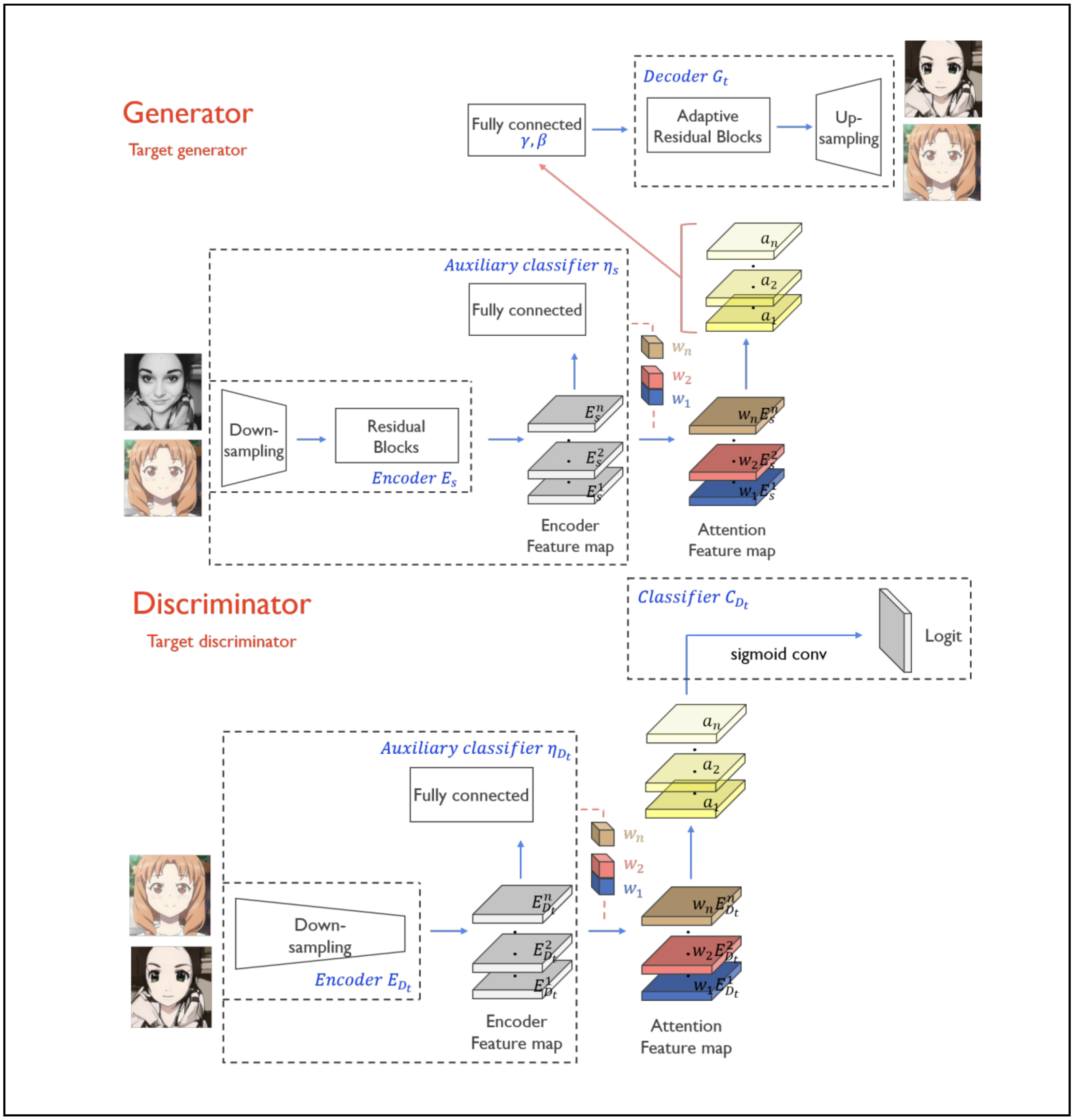

Figure 4. The model architecture of U-GAT-IT. Taken from [4].

$G_{s\to t}$ consists of an Encoder E_s and a decoder G_t and an auxiliary classifier η_s where $\eta_s(x)$ represent the probability that x comes from X_s. Let $E_s^k(x)$ be the k-th activation map of the encoder. The auxiliary classifier is trained to learn the weight of the k-th feature map for the source domain, w_k^s by using the global average pooling and global max pooling. By exploiting w_k^s, we can calculate a set of domain specific attention feature $a_s(x) = w_s * E_s(x) = \{w_s^k * E_s^k/1 \leq k \leq n\}$ where n is the number of encoded features maps. Then, the translation model $G_{s\to t}$ becomes equal to $G_t(a_s(x))$. The residual blocks with AdaLIN whose parameters γ and β

are dynamically computed by a fully connected layer from the attention map.

$$AdaLIN(a,\gamma,\beta) = \gamma * (\rho * \hat{a_I} + (1-\rho) * \hat{(a_L)}) + \beta, \quad (1)$$

$$\hat{a_I} = \frac{a-\mu_I}{\sqrt{\sigma_I^2+\epsilon}}, \hat{a_L} = \frac{a-\mu_L}{\sqrt{\sigma_L^2+\epsilon}} \quad (2)$$

$$\rho \leftarrow clip_{[0,1]} * (\rho - \tau * \Delta\rho) \quad (3)$$

where $\hat{a_I}$, $\hat{a_L}$ and σ_I and σ_L are channel-wise, layer-wise mean and standard deviation respectively, γ and β are parameters generated by the fully connected layer, τ is the learning rate and $\Delta\rho$ indicates the parameter update vector (the gradient) determined by the optimizer. The values of ρ are constrained to the range of [0, 1] simply by imposing bounds at the parameter update step.

Discriminator Let $x \in \{X_t, G_{s\to t}(X_s)\}$ represent a sample from the target domain and the translated source domain. Similar to other translation models, the discriminator D_t which is a multi-scale model consists of an encoder E_{D_t}, a classifier C_{D_t}, and an auxiliary classifier η_{D_t}. Unlike the other translation models, both $D_t(x)$ and $\eta_{D_t}(x)$ are trained to predict whether x comes from X_t or $G_{s\to t}(X_s)$. Given a sample x, $D_t(x)$ exploits the attention feature maps $a_{D_t}(x) = w_{D_t} * E_{D_t}(x)$ using w_{D_t} on the encoded feature maps $E_{D_t}(x)$ that is trained by $\eta_{D_t}(x)$. Then, our discriminator $D_t(x)$ becomes equal to $C_{D_t}(a_{D_t}(x))$.

Note : More information about the model can be found in the original paper [4].

4. Results

Our study of generative models was oriented towards the *Kaggle* competition for Monet painting generation. The results we present in this section were directly computed from the competition host in order to rank propositions. There was a computational time constraint of maximum 5 hours of training on a single GPU (3 hours if the model is trained on TPU).

Optimizer The original CycleGan architecture is trained with *Adam* optimizer ($l_r = 0.0002$). We tested a recent new optimizer $AdaBelief$ [8] on our CycleGan architecture. The method is supposed to stabilize the training. However, we were unable to get any improvement with this optimizer (best score : **80.48**).
The Bayesian CycleGan was also trained with *Adam* optimizer ($l_r = 0.0002$).
U-GAT-IT was trained with *Adam* optimizer ($l_r = 0.0001$).

Hyperparameters The CycleGAN model is trained with a batch size of **1** and **100** epochs.
The Bayesian CycleGAN model is trained with a batch size of **1** and **40** epochs. For every 10 epochs completed, we generated the target images for the kaggle competition and we took the best score.
The U-GAT-IT model is trained with a batch size of **1** and **60** epochs. For every 10 epochs completed, we generated the target images for the kaggle competition and we took the best score.
The implementation of Bayesian CycleGAN and U-GAT-IT was based on the original code (https://github.com/ranery/Bayesian-CycleGAN and https://github.com/znxlwm/UGATIT-pytorch respectively).
Unfortunately, we were unable to make the implementation TPU-compatible, then we decided to run the training on Colab Pro with a TESLA V100. It took 40 hours to train the Bayesian CycleGAN for 40 epochs and 65 hours to train the U-GAT-IT model for 60 epochs.

Competition results We sent **42** submission on the *Kaggle* competition. We performed our best result with a **UGATIT** model scored 17^{th} out of 221 teams (at the moment we write the report). The best score on the leaderboard is 37.72 . Our fine-tuned cycleGAN architecture achieved as well reasonably good performance compared to the other competitors.

	CycleGan	Bayesian CycleGAN	UGATIT
Mi-FID	48.39	55.08	**39.32**

Table 1. Best score obtained on the competition for the different models (lower is better).

Visual Interpretation The visual results presented in Fig.5 illustrates that our models were able to capture the general style that *Monet paintings* have in common and reproduce it on a real-life picture. Through the study of the visual results, we noticed that input pictures from landscape are generally better converted compared to city or people since the distribution of *Monet painting* is mainly composed of this category.

5. Conclusion

In this project, we investigated classical ([7]) and state of the art ([4],[6]) models for style transfer on a *Monet Painting* dataset. The study was motivated through the participation of an open competition hosted on *Kaggle*. We analyzed the current methods for generative model evaluation (Inception Score, FID, Mi-FID) and the limits to their interpretation. We noticed how unstable generative models

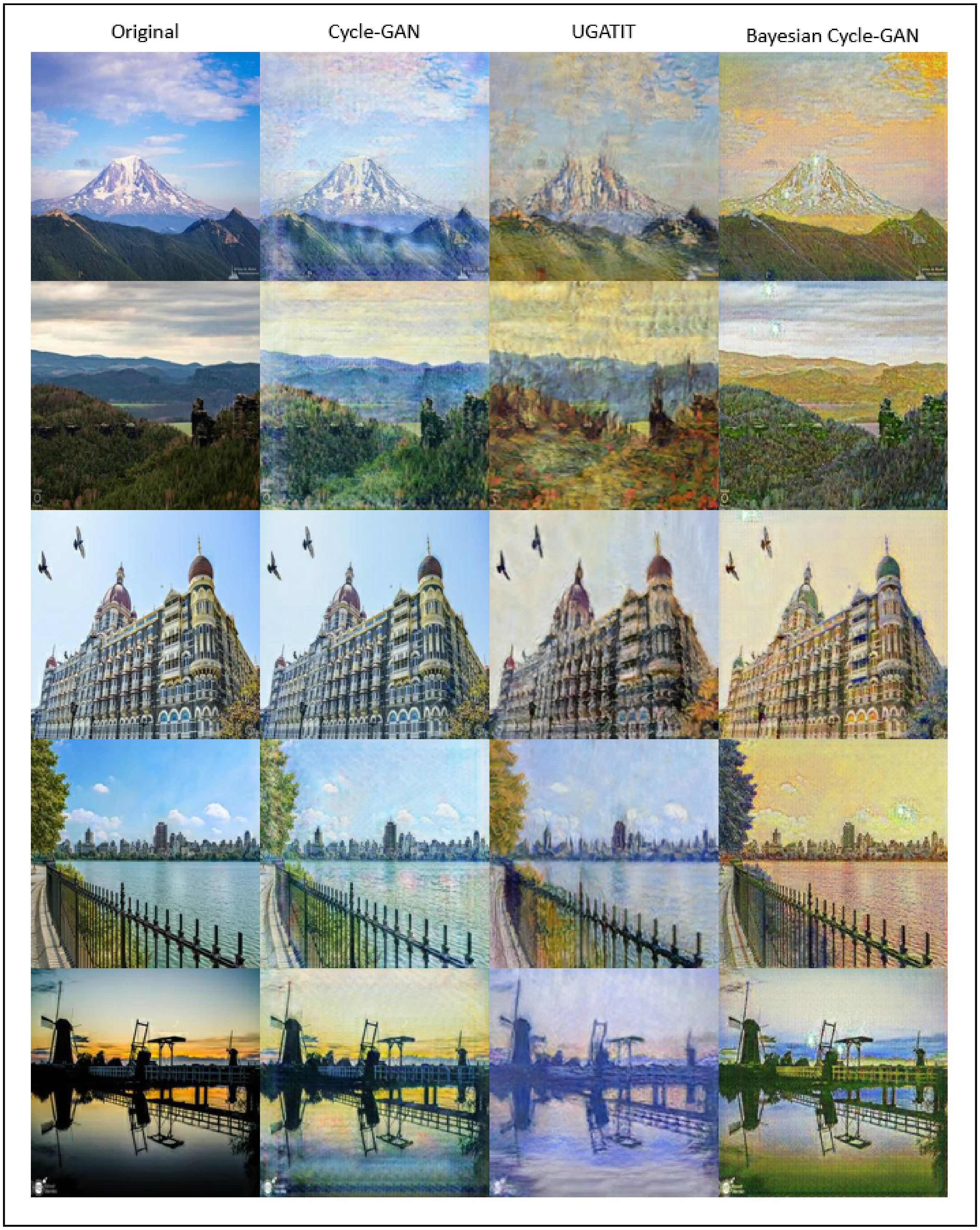

Figure 5. Comparison of models with different input pictures. As the Mi-FID score suggested, we consider that **UGATIT** achieves the best results

are because of the adversarial loss used for the training. Our implementation of CycleGAN and recent UGATIT model achieved good performance on the leaderboard, which is confirmed by our visual analysis of their outputs.

Code The implementation of our custom Cycle-GAN and the other models used is available on the Github repository of our project.

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. 3

[2] C.-Y. Bai, H.-T. Lin, C. Raffel, and W. Kan. A large-scale study on training sample memorization in generative modeling, 2021. 2

[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. 2

[4] J. Kim, M. Kim, H. Kang, and K. Lee. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation, 2020. 3, 4, 5, 6

[5] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. 1

[6] H. You, Y. Cheng, T. Cheng, C. Li, and P. Zhou. Bayesian cycle-consistent generative adversarial networks via marginalizing latent sampling, 2020. 3, 6

[7] 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. 2, 6

[8] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. S. Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients, 2020. 6