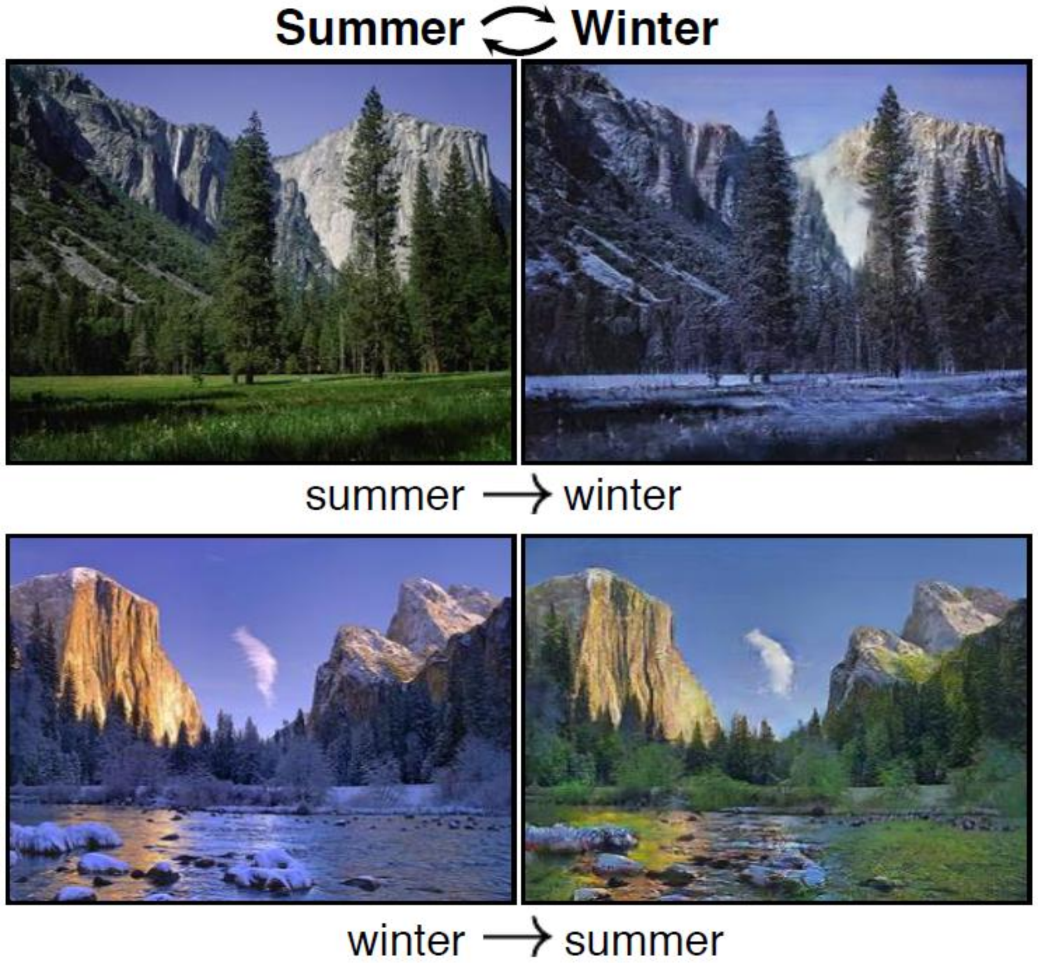


Final course paper on:

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

JUN-YAN ZHU, TAESUNG PARK, PHILLIP ISOLA, ALEXEI A. EFROS

Palumbo Marco

Outline

- Introduction of the topic
- CycleGAN Architecture
- Objective functions
- CycleGAN implementation V1
- CycleGAN implementation V2
- Results
- Conclusions

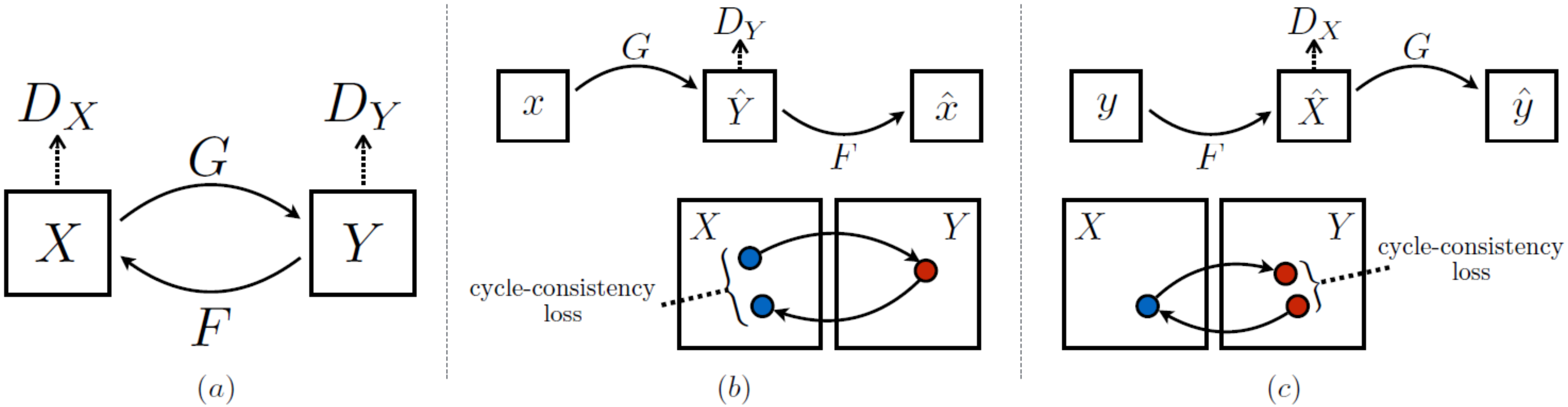
Introduction of the topic

- The goal of the authors in this paper is to create a GAN that is able to translate image samples from class X to class Y and vice versa, using an **unsupervised approach**.
- In practice, the goal is to learn a **mapping function *G: X → Y*** that is able to generate images of class Y starting from images of class X, so that the generated images are indistinguishable from those of the target class (the same is true for the inverse function ***F: Y → X***).
- The authors present qualitative results on various tasks, such as neural style transfer, object transfiguration and photo enhancement, so their method represents a **general-purpose solution** for various vision and graphics tasks. In our case the focus is on the **season transfer task** (i.e. transforming an image of a summer landscape into an image of a winter landscape and vice versa).

CycleGAN Architecture

- In general, obtaining datasets containing paired images is a difficult, time-consuming and costly task.
- We want to train a **generator** *G: X* → *Y* such that it produces images ŷ indistinguishable from the real ones $y \in Y$, making use of an **adversary** (**discriminator**) trained to classify real samples y and generated samples ŷ.
- Although the algorithm follows an unsupervised approach (due to the **lack of paired images**), it is possible to take advantage of the "**cycle consistency**" property to correctly address the work of the generators.
- In other words, we want the mapping functions *G* and *F* to be **inverse** of each other and both bijective, so that we obtain $F\big(G(x)\big) \approx x$ and G$\big(F(y)\big) \approx y$.

CycleGAN Architecture



Adversarial losses

Forward cycle-consistency loss

Backward cycle-consistency loss

Objective functions

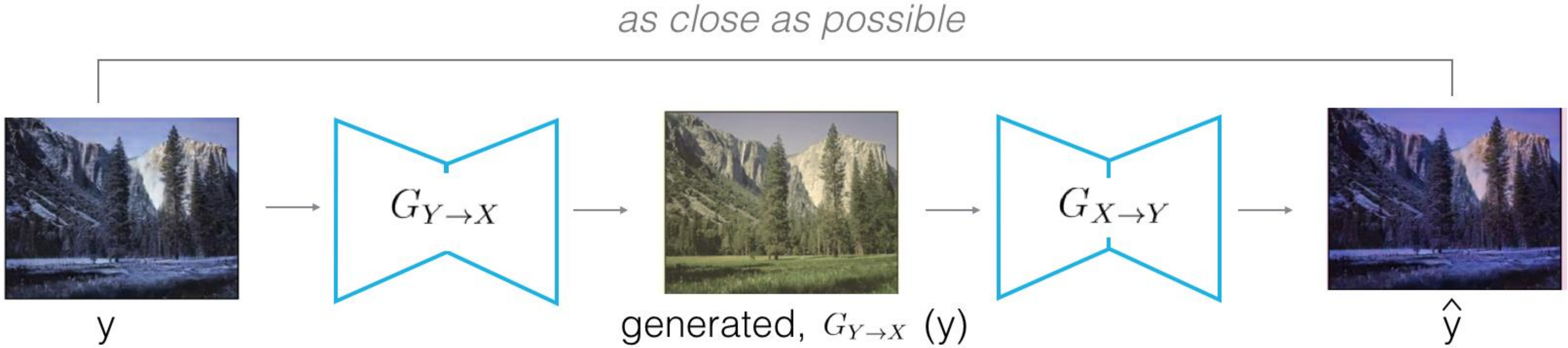
- The objective function (**loss function**) contains two terms: one of **adversarial loss** to make the generated images match those of the target domain, and one of **cycle consistency loss** to prevent the mapping functions *G* and *F* from contradicting each other.

- $Adversarial\ loss = MSE\big(D(x_{real})\big) + MSE\left(D\left(x_{fake}\right)\right) = \mathbb{E}[(D(x_{real}) - 1)^2] + \mathbb{E}[\big(D\big(x_{fake}\big) - 0\big)^2]$→ same applies for class Y
- $Cycle\ consistency\ loss = \lambda_{cycle} \cdot [\mathbb{E}(||x - F(G(x))||) + \mathbb{E}(||y - G(F(y))||)]$
- (Optional) $Identity\ loss = \lambda_{identity} \cdot [\mathbb{E}(||x - F(x)||) + \mathbb{E}(||y - G(y)||)]$

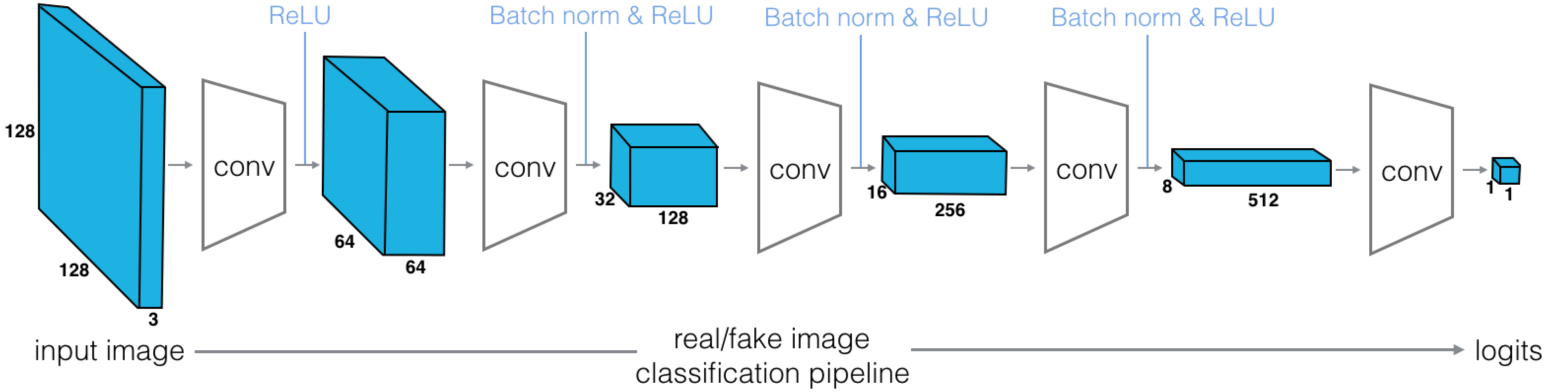
- $\boldsymbol{CycleGAN\ loss = Adversarial\ loss + Cycle\ consistency\ loss +\ (Identity\ loss)}$

- D_X and D_Y try to minimize the adversarial loss, while the generators *G* and *F* try to minimize the real part when the input x_{real} is substituted with x_{fake} in order to fool the discriminators. At the same time, we want to minimize the cycle consistency loss term.

Example of reconstructed image

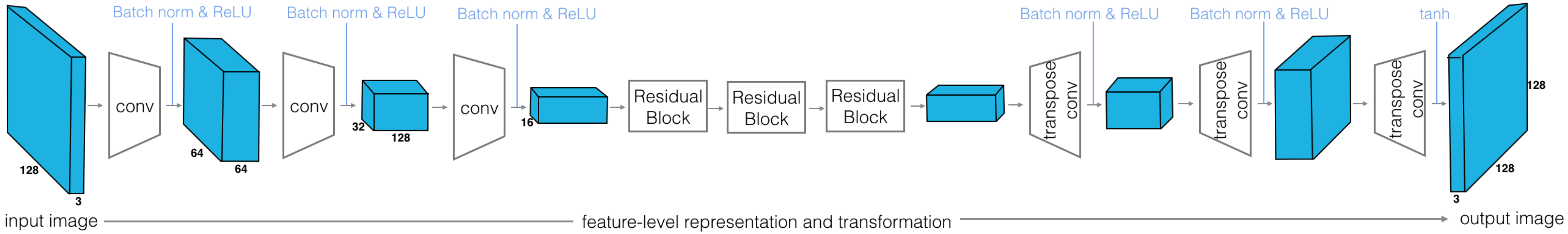


CycleGAN implementation V1 - Discriminator



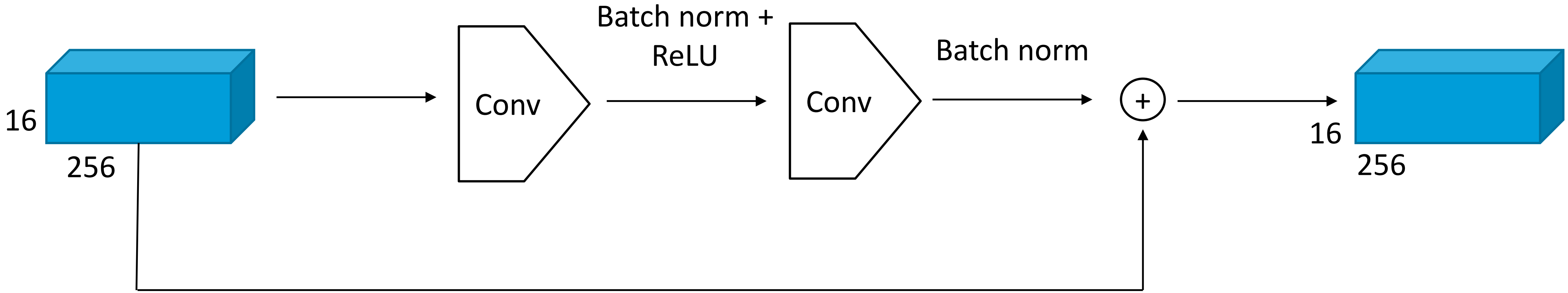
- **4 convolutional layers with ReLU activations** which downsample the image by a factor of 2 at each step.
- **3** intermediate **batch normalization layers** to standardize images.
- **1 final convolutional layer** for classification.

CycleGAN implementation V1 - Generator



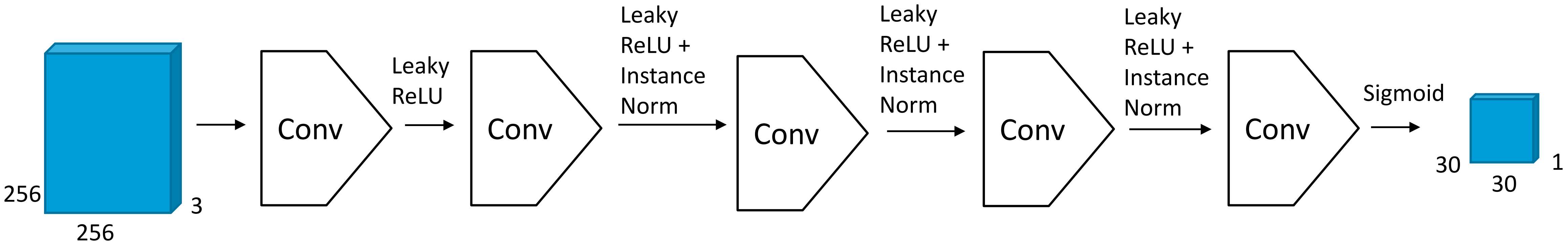
- **1 encoder** which transforms the input image, with dimensions 128x128x3, into an intermediate representation (a tensor with dimensions 16x16x256).
- **3 residual blocks**, each of which is composed by 2 convolutional layers and 2 batch normalization layers.
- **1 decoder** which transforms the output of ResBlocks into an image of dimension 128x128x3 like the original one.

CycleGAN implementation V1 - ResBlocks



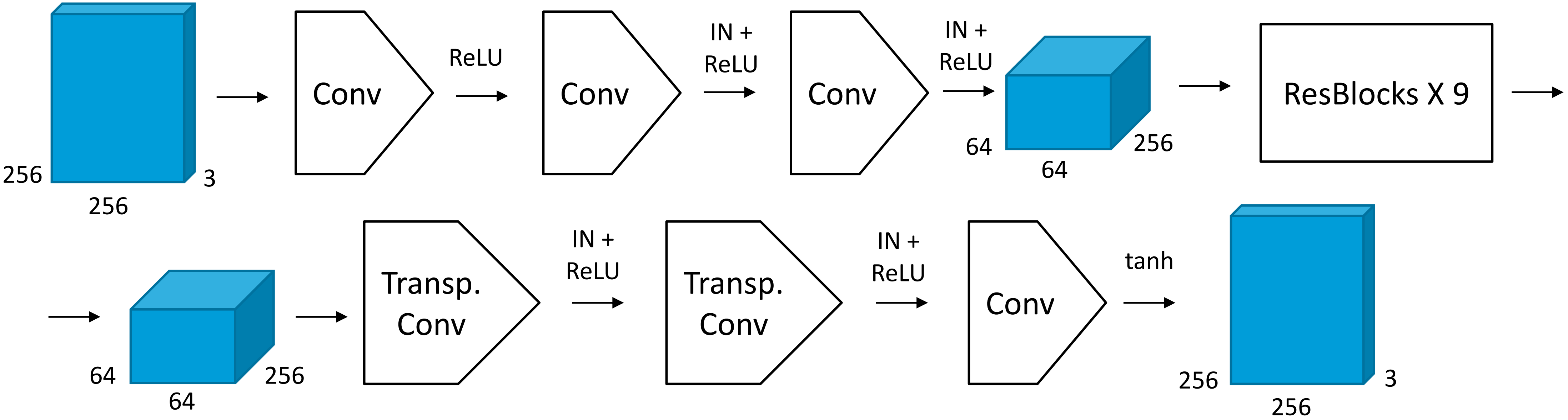
- Resnet blocks are based on connecting the output of one convolutional layer with the input of the previous convolutional layer. These layers are used in the case of neural networks with many layers (deep neural networks) to solve the problem of evanescent or "explosive" gradients, in order to avoid problems of instability during the training phase (when accuracy saturates or even worsens in subsequent epochs).
- In this case, the two convolutional layers must have the same number of inputs and outputs. The output of a ResBlock is equal to $y = F(x) + x$, where ***F(x)*** is called the "**residual function**". This function simplifies the generator training procedure during the parameter optimization phase.

CycleGAN implementation V2 - Discriminator


256
256
3
Conv
Leaky ReLU
Conv
Leaky ReLU + Instance Norm
Conv
Leaky ReLU + Instance Norm
Conv
Leaky ReLU + Instance Norm
Conv
Sigmoid
30
30
1

- Unlike the previous model, in this one the discriminator classifies images of size 256x256x3 always using convolutional layers, in which, however, the activation function is the **Leaky ReLU** (with a negative slope of 0.2), and **Instance Normalization** is used instead of Batch Normalization. In "*Instance Normalization*", mean and variance are calculated ***for*** each individual channel ***for*** each individual sample ***across*** both spatial dimensions.
- The output of the discriminator is not a single scalar but a 30x30 size matrix, where each value (0 or 1) is responsible for classifying a **70x70 size patch** of the input image (this type of net is taken from the model called **PatchGAN**).

CycleGAN implementation V2 - Generator

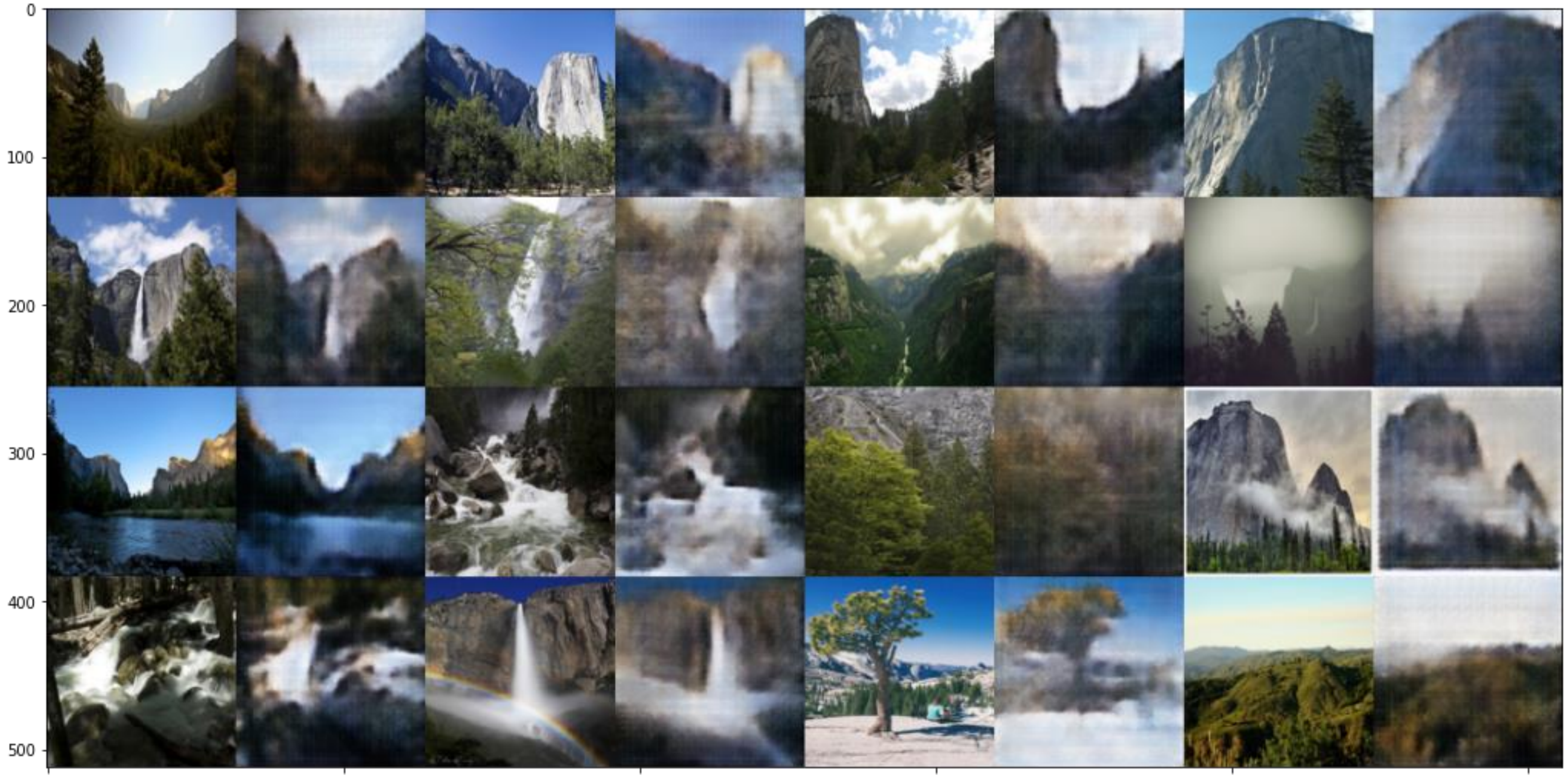


- The generator uses 9 residual blocks (since the input size is 256x256), whose internal architecture is similar to the one seen previously, with the only exception given by the use of Instance Normalization instead of Batch Normalization.

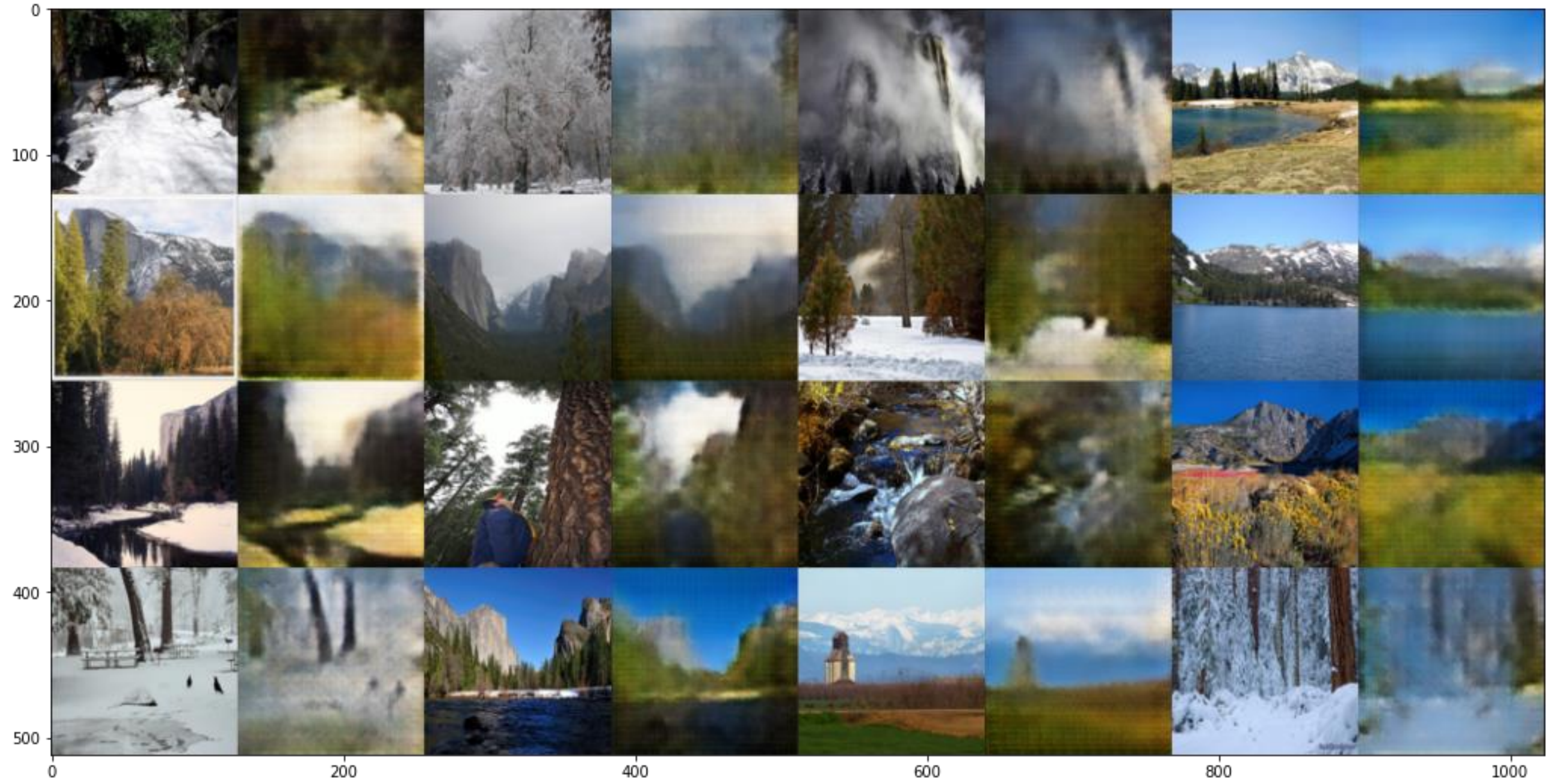
Results V1

- The following results were obtained using the Adam optimizer with the following hyperparameters:
  - LR = 0.0002
  - beta1 = 0.5
  - beta2 = 0.999
- For the cycle consistency loss $\lambda_{cycle} = 10$
- Batch_size = 16
- Number of epochs = 4000

Summer → Winter



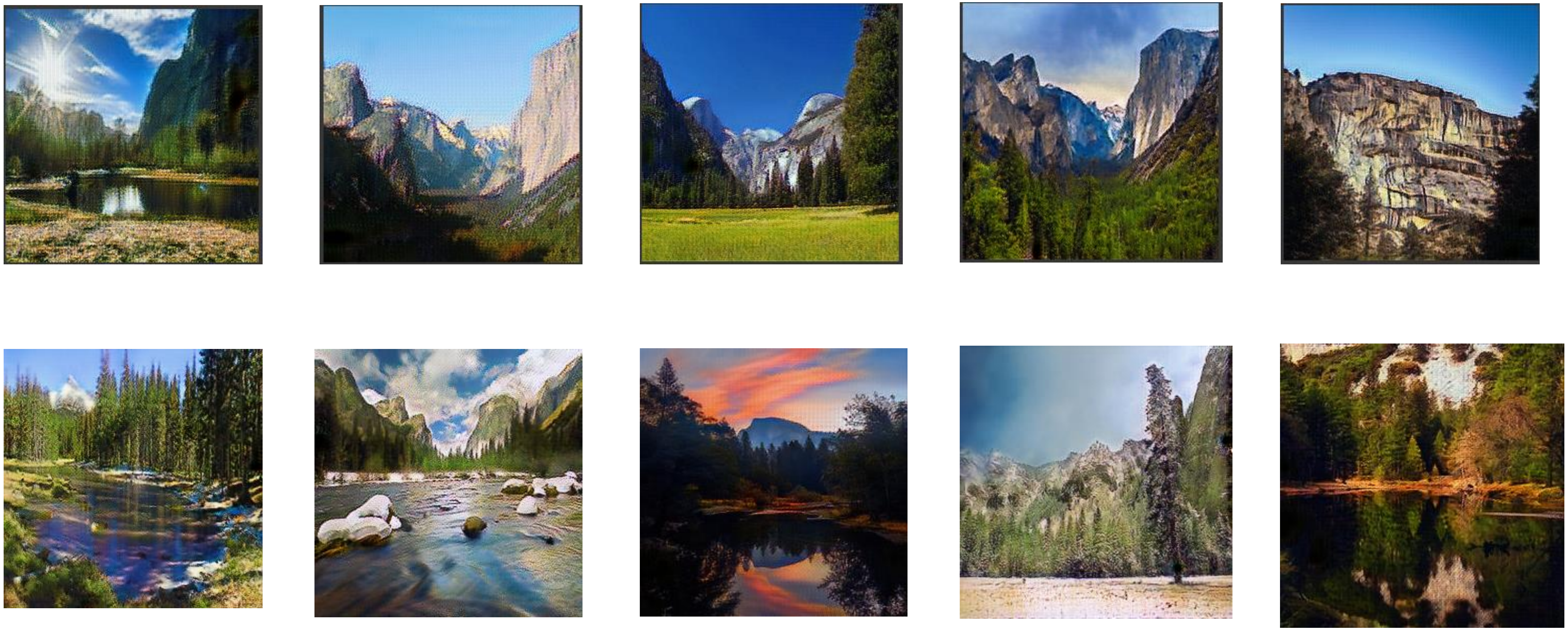
Winter → Summer



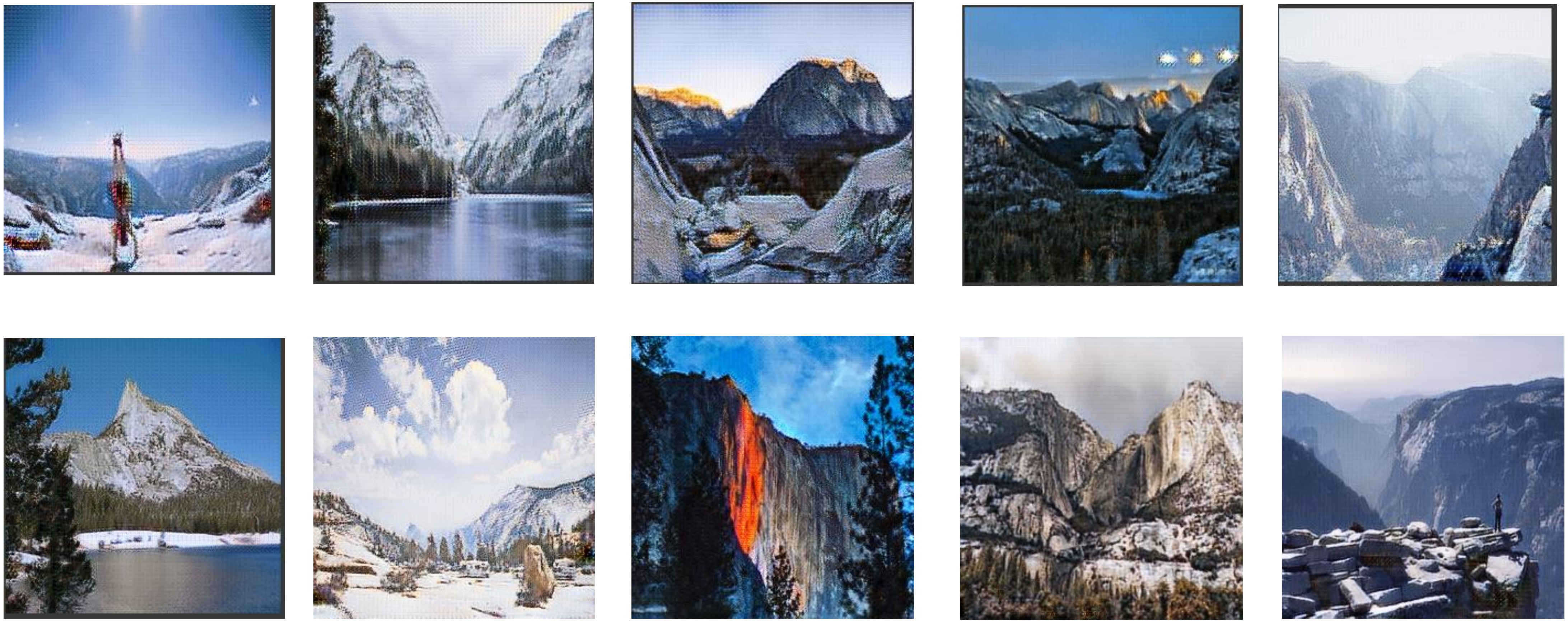
Results V2

- The following results were obtained using the Adam optimizer with the following hyperparameters:
  - LR = 0.0002
  - beta1 = 0.5
  - beta2 = 0.999
- For the cycle consistency loss $\lambda_{cycle} = 10$
- Batch_size = 1 → images are larger in dimension with respect to the V1 Net
- Number of epochs = 25

Fake summer images



Fake winter images



Conclusions

- The resulting V2 model images were generated from the training set. Nevertheless, since the training set does not include paired images, obtaining images translated from one domain to another is still an excellent result.
- In any case, the CycleGAN authors specify that their model works well for tasks involving apparent image changes (color and texture mainly), but they get poor results when they have to transform images by changing geometric shapes (for example, dog → cat) . According to their hypotheses, this is due to the fact that the generator model used, copied by the author Justin Johnson, is calibrated for neural style transfer, therefore for tasks in which the main objective is to change the coloring of the images.
- Furthermore, the observed gap between the results obtainable through paired training data and those obtained with the unsupervised approach is in some cases unbridgeable. This problem could be solved by adding some form of semantic supervision to the model. In any case, the purpose of the paper is to push the potential of this unsupervised approach to its limits, showing the quality of the results that can be obtained.

Critical analysis

- In my opinion, the work done by the authors of CycleGAN presents excellent results that I was able to replicate at least partially.
- The use of pre-trained models with already optimized weight values can help to speed up the training phase of the CycleGAN and slightly improve the quality of the images generated.
- It has been seen how the use of images of size 256x256 in the V2 model has allowed to obtain more accurate results, despite the training phase taking much more time.
- In my opinion, an improvement that can be made to the model concerns the use of the **identity loss function** (discussed above) which allows to direct the model to the generation of images without changing the original tint or coloration. The idea is to compare the real images (*x* or *y*) with the corresponding generated images (x_{fake} and y_{fake}) making sure that the difference between the two is minimal. However, it should be noted that the use of this loss function leads the model to use a supervised approach, and in some cases, it may not lead to significant improvements.

Thank you!