Neural style transfer

Deep learning project

Maxime Meurisse, Adrien Schoffeniels, Valentin Vermeylen

June 5, 2020

University of Liège

Neural style transfer

Ultimate goal : merge the style of an image and the content of another image.

Neural style transfer results (Gatys et al. technique)

Plan of the presentation

Development of the 2 techniques used :

- *A Neural Algorithm of Artistic Style* (Gatys et al., 2015)
- *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* (Zhu et al., 2017)

and a conclusion.

A Neural Algorithm of Artistic Style (Gatys et al., 2015)

Method

- Use a (pre-trained) convolutional neural network (CNN)
- **Key** : style and content of an image are separable in CNN
- Add new layers in the CNN to re-construct images

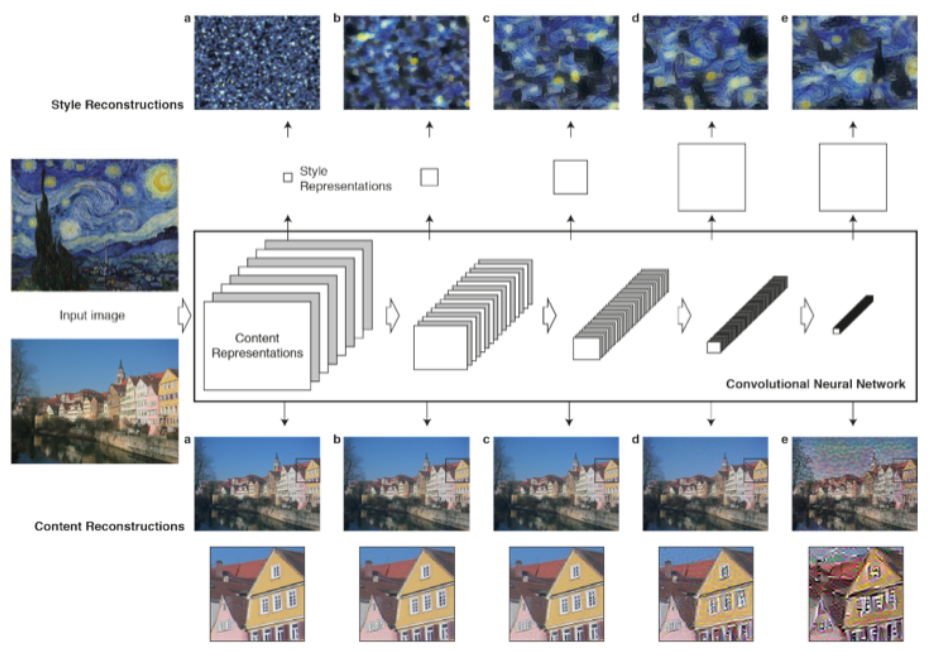


Image reconstruction - CNN

Source: `https://arxiv.org/gatys/1508.06576.pdf`

Method - Loss functions

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

where, for a layer l,

$$\mathcal{L}_{\text{content}} = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l} \right)^2$$

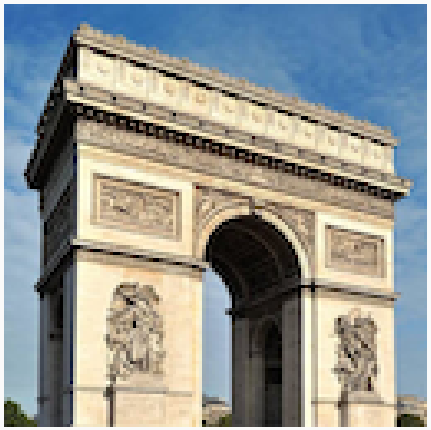
and

$$\mathcal{L}_{\text{style}} = \sum_{l=0}^{L} w_l \frac{1}{4 N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l} \right)^2$$

- F_{ij} and P_{ij}, $\in \mathcal{R}^{N_l \times M_l}$: activation of the i^{th} filter at position j in layer l
- G_{ij} and A_{ij} : Gram matrices

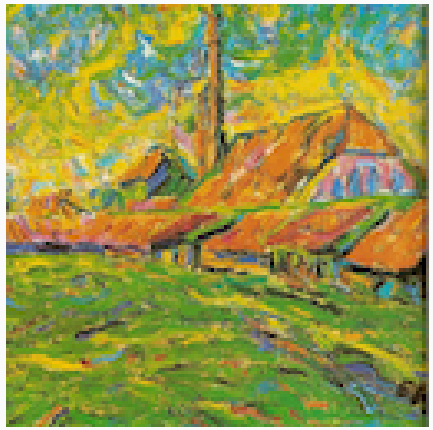
Results

Base images used :



(a) Content

(b) Style 1



(c) Style 2

Evaluation methods :

- **qualitative** (human eyes)
- quantitative (loss functions : content and style)

Different (significant) parameters :

- input image
- model used
- architecture of the model used
- weights
- added layers

Input image

2 ways to initialize the input image :

- copy of content image
- noisy content image

(a) Copy of content image

(b) Noisy content image

Very similar results obtained.

Copy of content image chosen because smoother results on average.

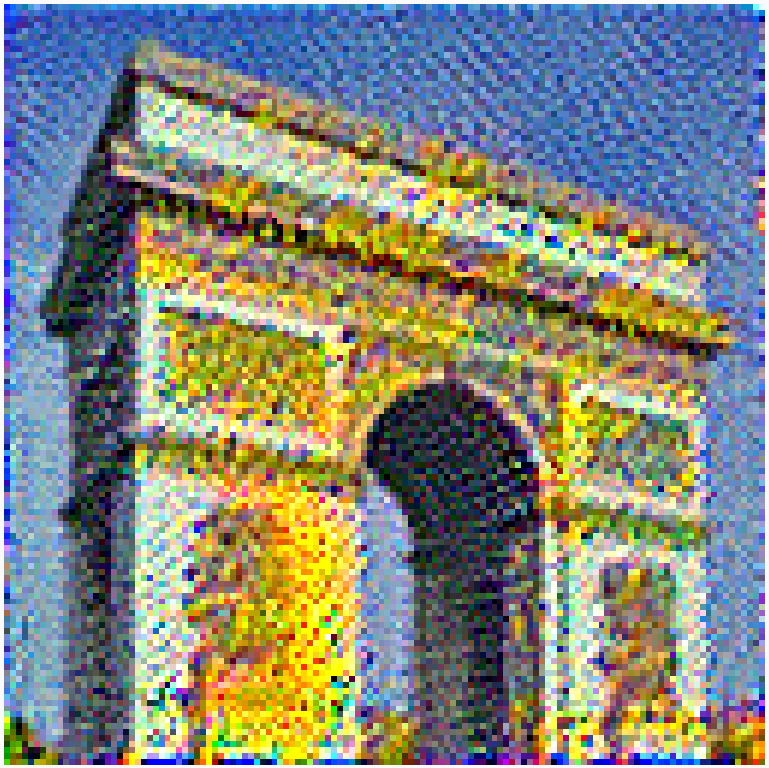
Model used

Base model used : **VGG19**

- not pre-trained
- pre-trained

We also tried :

- GoogLeNet
- AlexNet

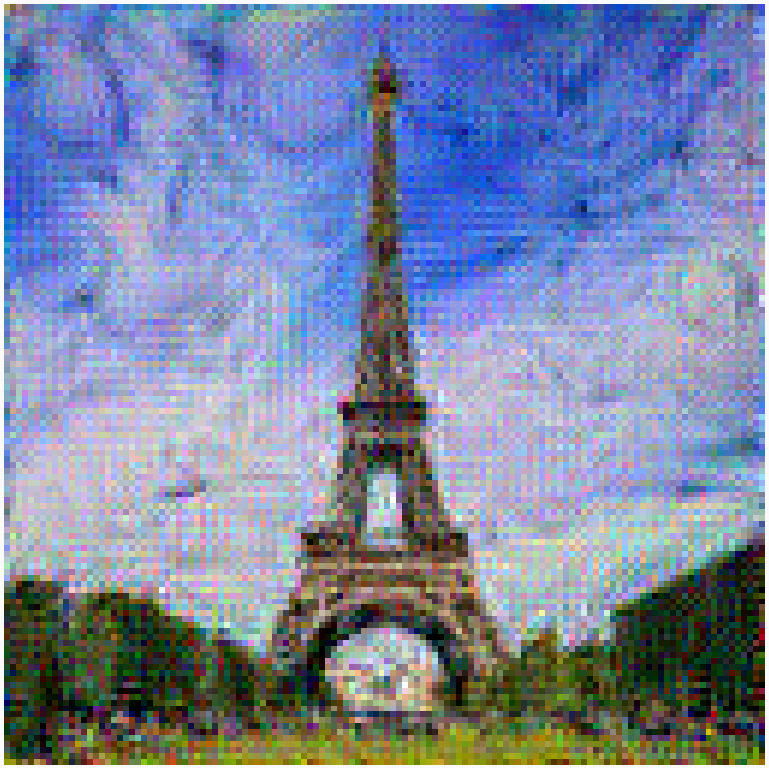


(a) VGG19 : not pre-trained

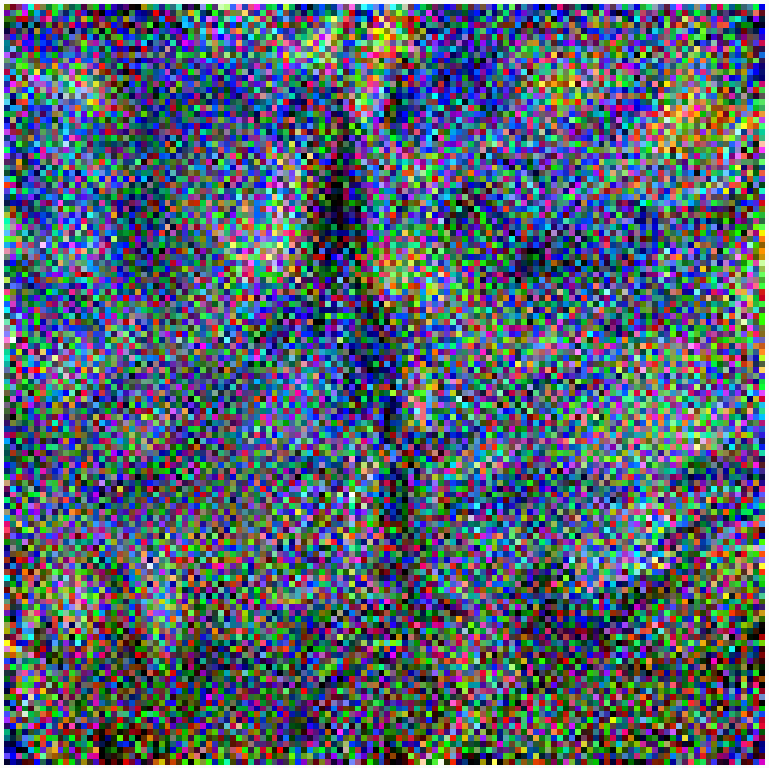
(b) VGG19 : pre-trained

Pre-trained model gives much better results than the non pre-trained model.

VGG19 pre-trained model chosen.



(a) GoogLeNet : pre-trained



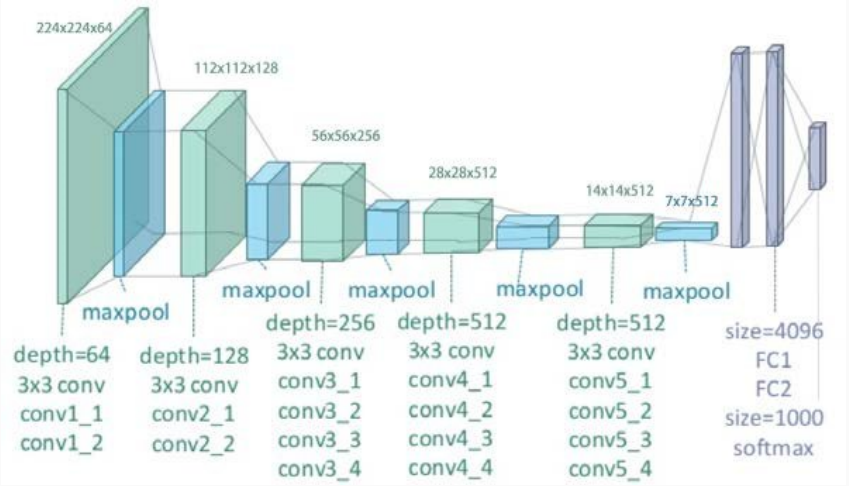
(b) AlexNet : pre-trained

Very bad results obtained.

Certainly due to a too simple (AlexNet) or too complex/unability to modify submodules (GoogLeNet) architecture.

Architecture of the model used

VGG19 uses **max** pooling layers by default.



VGG19 architecture

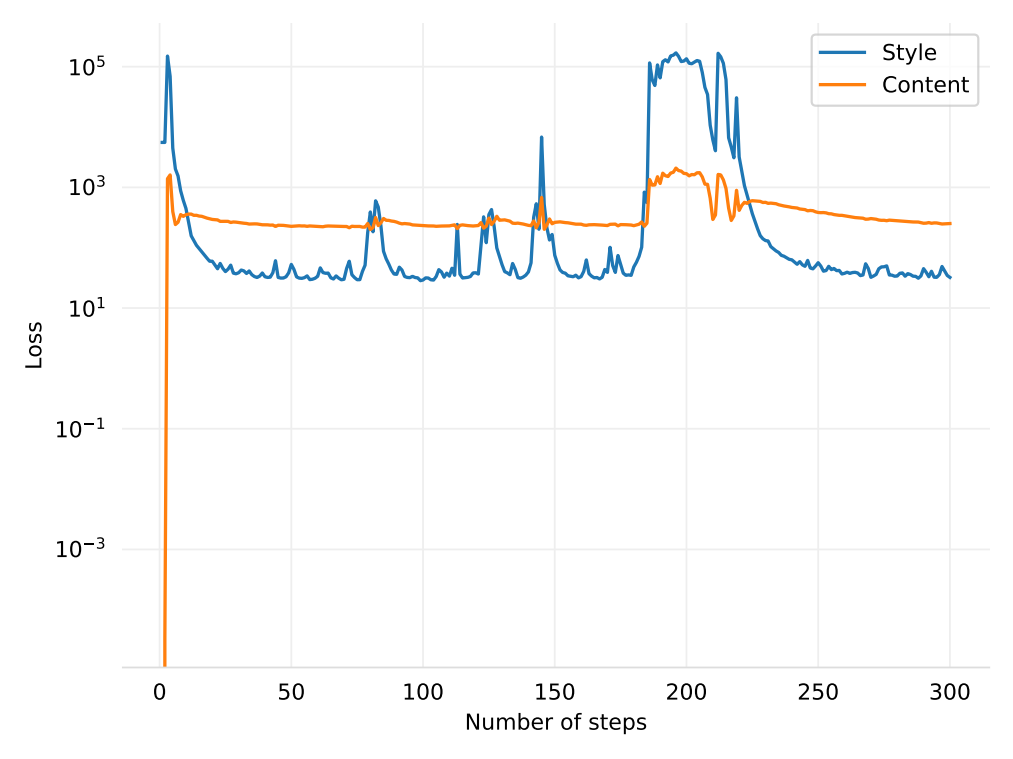
Source: `https://mc.ai/style-transfer-of-images-with-cnn-in-pytorch/`

Authors suggest to replace them by **average** pooling layers to get more appealing results.

(a) Max pooling layers

(b) Average pooling layers

Both results seems to be very good, but...

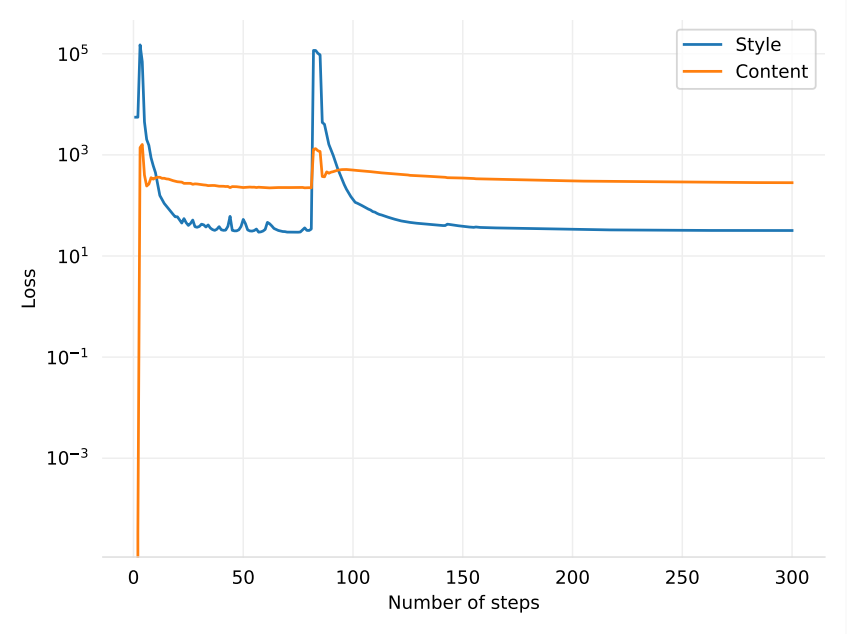


Average pooling layers : loss functions

...oscillations in loss functions ! (which sometimes leads to bad results...)

We try to add a **scheduler** to reduce oscillations.

(a) Average pooling layers



(b) Loss functions with scheduler

Scheduler seems to be efficient !

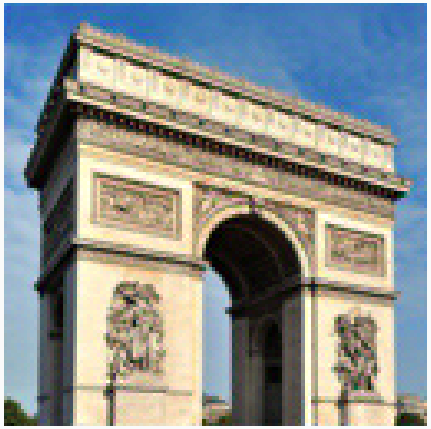
Average pooling layers with scheduler chosen.

Weights

The α and β weights in the loss function are used to distribute the style and content of the resulting image.

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

We set α to 10 and try different ratios α/β.



(a) 10^{-2}

(b) 10^{-3}

(c) 10^{-4}

(d) 10^{-5}

(e) 10^{-6}

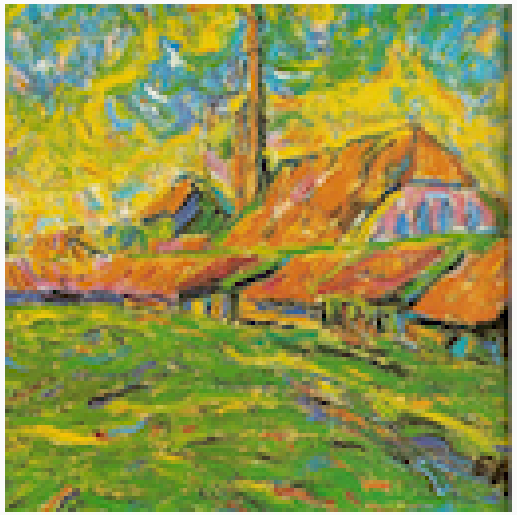
(f) 10^{-7}

Ratio $\alpha/\beta = 10^{-5}$ chosen (with $\alpha = 10$).

Added layers

The **positions of the construction layers in the network** have a significant influence on the result.

We tried to place them at different locations to try to understand how they influence the result.



Style used

(a) s: conv1
c: conv1

(b) s: conv3
c: conv1

(c) s: conv5
c: conv1

(d) s: conv9
c: conv1

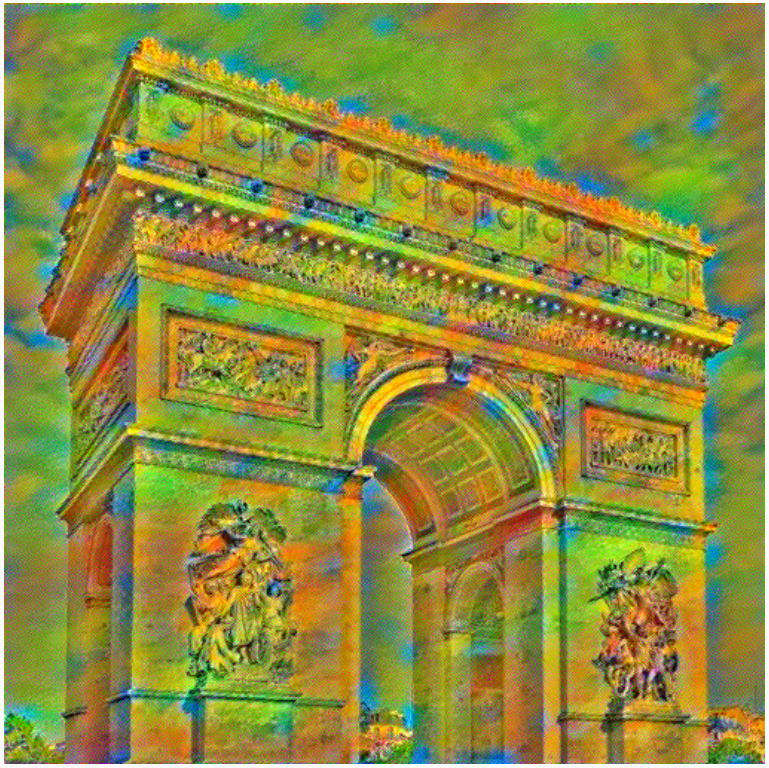
(e) s: conv1
c: conv1

(f) s: conv1
c: conv3

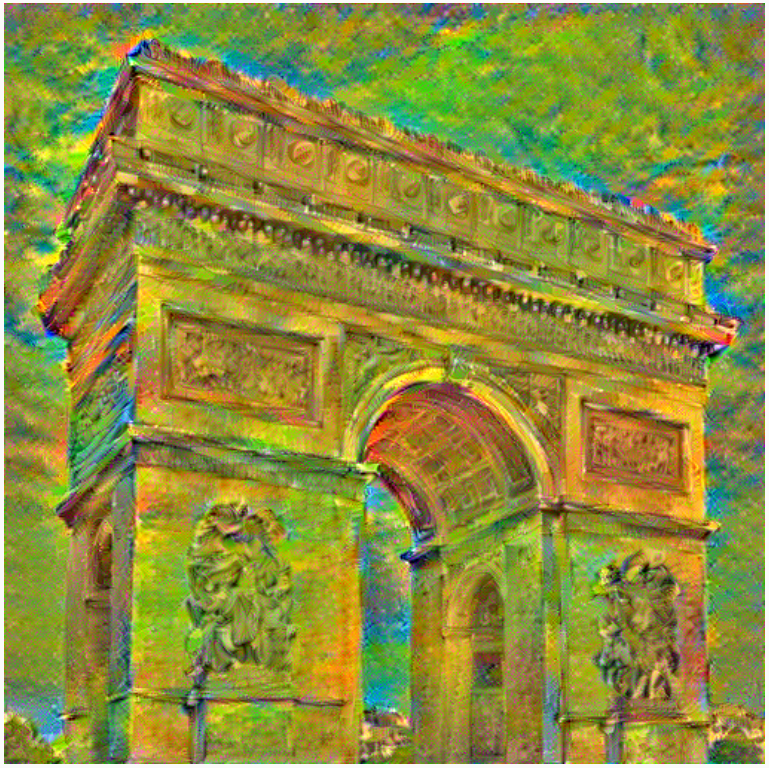
(g) s: conv1
c: conv5

(h) s: conv1
c: conv9

Style layers placed at the beginning and content layer around the middle seems to be the best combination.



(a) Starting architecture



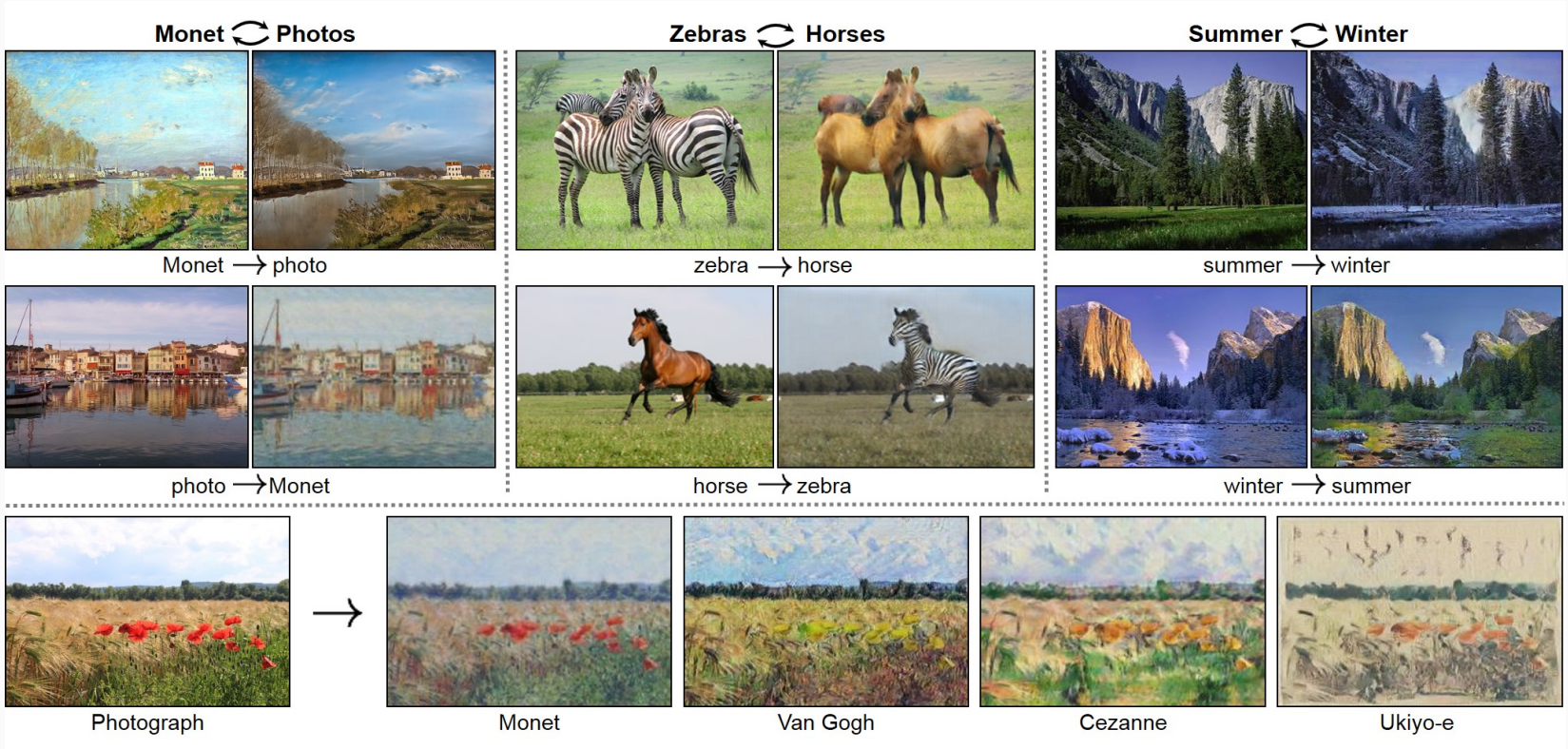
(b) Modified architecture

Result we obtained by modifying the locations of the construction layers.

CycleGAN (Zhu et al., 2017)

Introduction

CycleGAN allows *unpaired image-image translation* using two Generative Adversarial Networks.

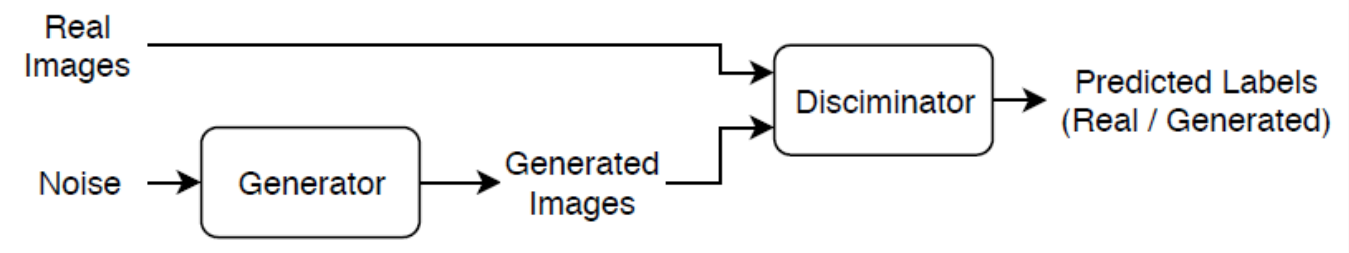


Examples of CycleGAN applications

Source: `https://junyanz.github.io/CycleGAN/`

A word about GANs

- **Generative** : Those models learn how to create data similar to what they have been fed.
- **Adversarial** : A generator and a discriminator compete.



GAN idea

Source: `https://www.mathworks.com/help/deeplearning/ug/train-conditional-generative-adversarial-network.html`

The idea of CycleGAN

Use two generators (one per domain) and two discriminators.

The two generators should lead to bijection.

$\longrightarrow$ **Cycle Consistency Loss** encouraging

$$F(G(x)) = x \text{ and } G(F(y)) = y.$$

This helps avoid mode collapse.

Losses

- Adversarial loss (MSE in actual training):

$$\mathcal{L}_{\mathrm{GAN}}\left(G, D_Y, X, Y\right) = \mathbb{E}_{y \sim p_{\mathrm{data}}\,(y)}\left[\log D_Y(y)\right] + \mathbb{E}_{x \sim p_{\mathrm{data}}\,(x)}\left[\log\left(1 - D_Y(G(x))\right)\right]$$

- Cycle Consistency loss :

$$\mathcal{L}_{\mathrm{cyc}}\left(G, F\right) = \mathbb{E}_{x \sim p_{\mathrm{data}}\,(x)}\left[\|F(G(x)) - x\|_1\right] + \mathbb{E}_{y \sim p_{\mathrm{data}}\,(y)}\left[\|G(F(y)) - y\|_1\right]$$

- Identity loss :

$$\mathcal{L}_{\mathrm{identity}}(G, F) = \mathbb{E}_{y \sim p_{\mathrm{data}}(y)}\left[\|G(y) - y\|_1\right] + \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}\left[\|F(x) - x\|_1\right]$$

Losses

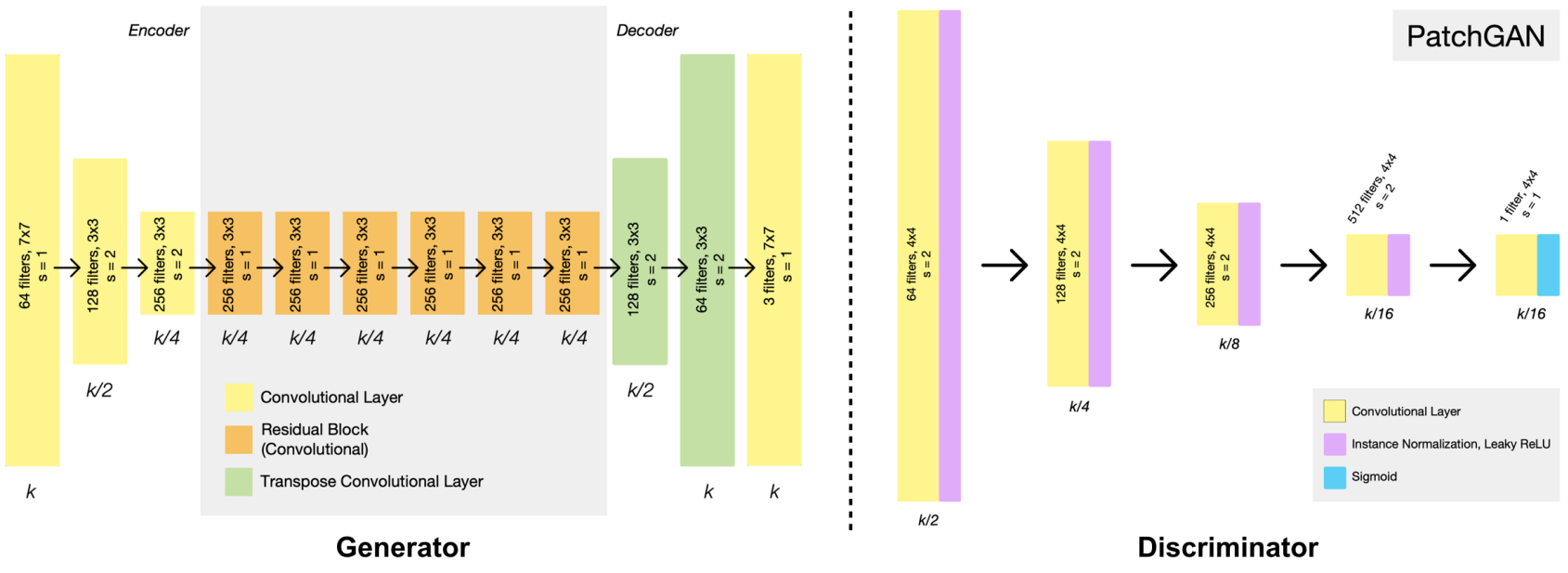
Full loss :

$$\mathcal{L}\left(G, F, D_X, D_Y\right) = \mathcal{L}_{\mathrm{GAN}}\left(G, D_Y, X, Y\right) + \mathcal{L}_{\mathrm{GAN}}\left(F, D_X, Y, X\right) + \lambda_{cyc}\mathcal{L}_{\mathrm{cyc}}(G, F)(+\lambda_{id}\mathcal{L}_{\mathrm{identity}}(G, F))$$

Goal :

$$G^*, F^* = \arg\min_{G,F} \max_{D_x,D_Y} \mathcal{L}\left(G, F, D_X, D_Y\right)$$

Architecture

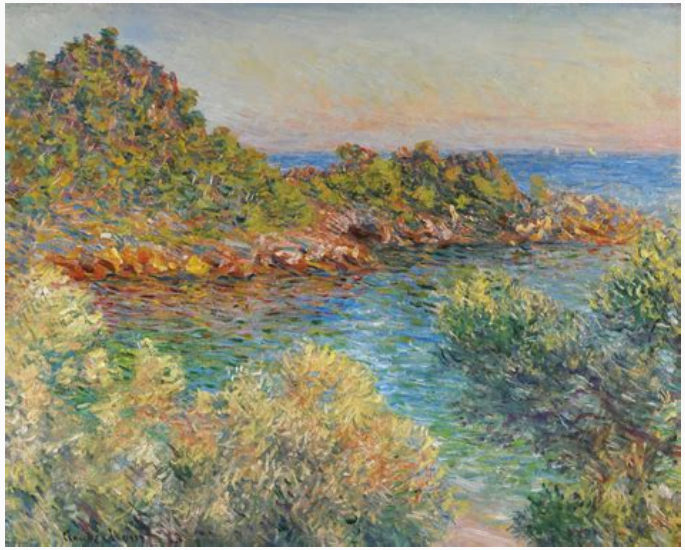


CycleGAN architecture

Source: `https://www.lyrn.ai/2019/01/07/instagan-instance-aware-image-to-image-translation/`

Input image

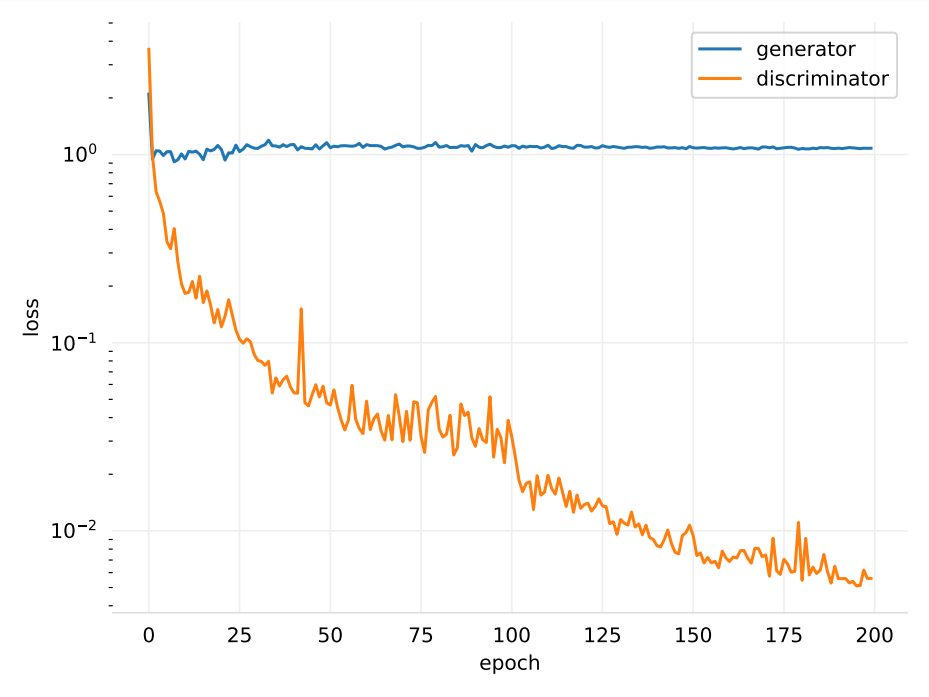
(a) Input image



(b) Claude Monet, Près Monte-Carlo

Results

All training dataset (6000 + 1000 images) but limiting to 20 batches of 3 + 3 images per epoch :



(a) Losses for the limitation of batches

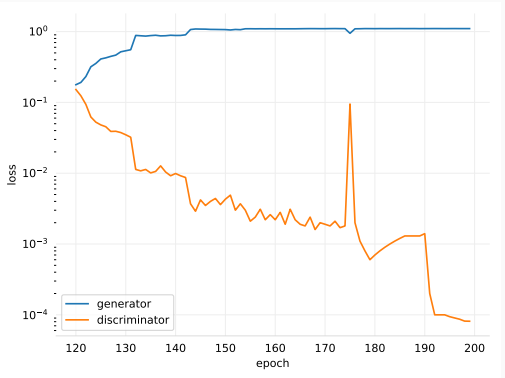
(b) Result obtained

Results

Using only one model with identity loss (to avoid mode collapse) and plateau LR (0.01 and 0.0001):[1]

(a) Result for 100 epochs

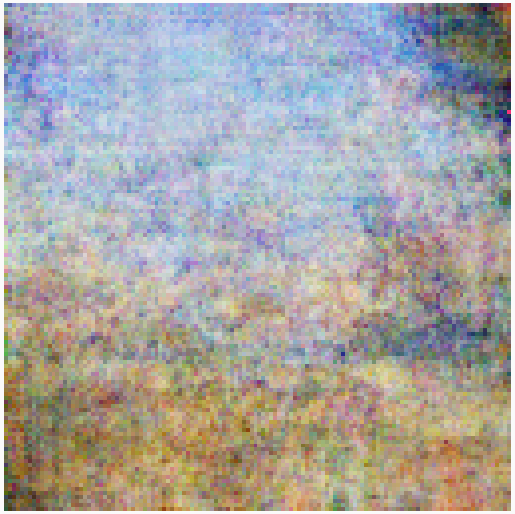
(b) Result for 200 epochs



(c) Losses

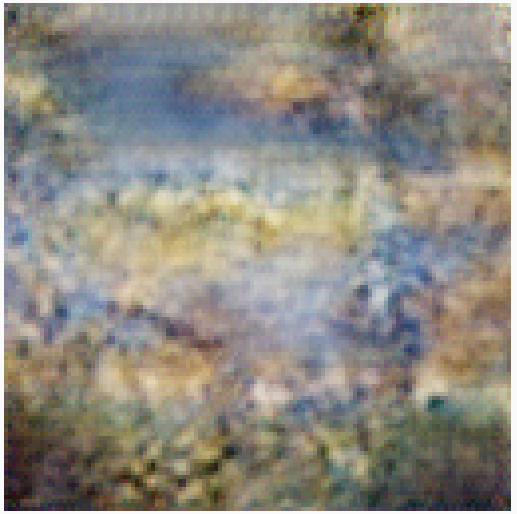
[1] Unless otherwise said, we used 120+750 images as training

Other results

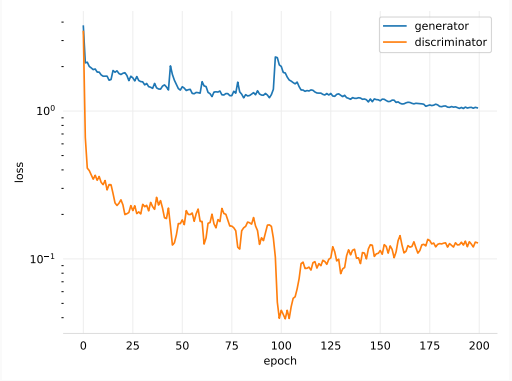


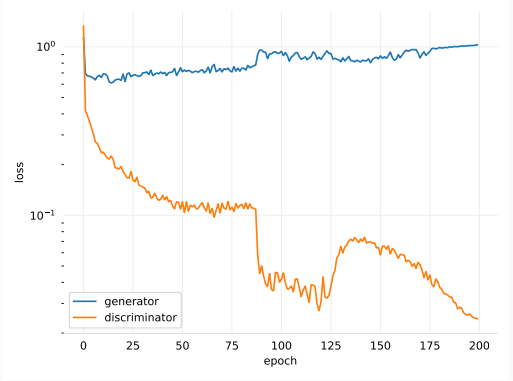
(a) $\lambda_{cyc} = 10$ and dataset (50+150)

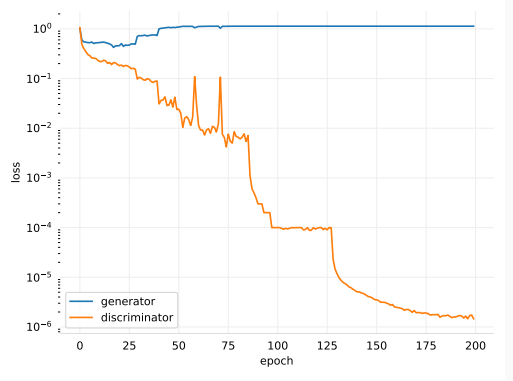
(b) $\lambda_{Id} = \lambda_{Cyc} = 1$



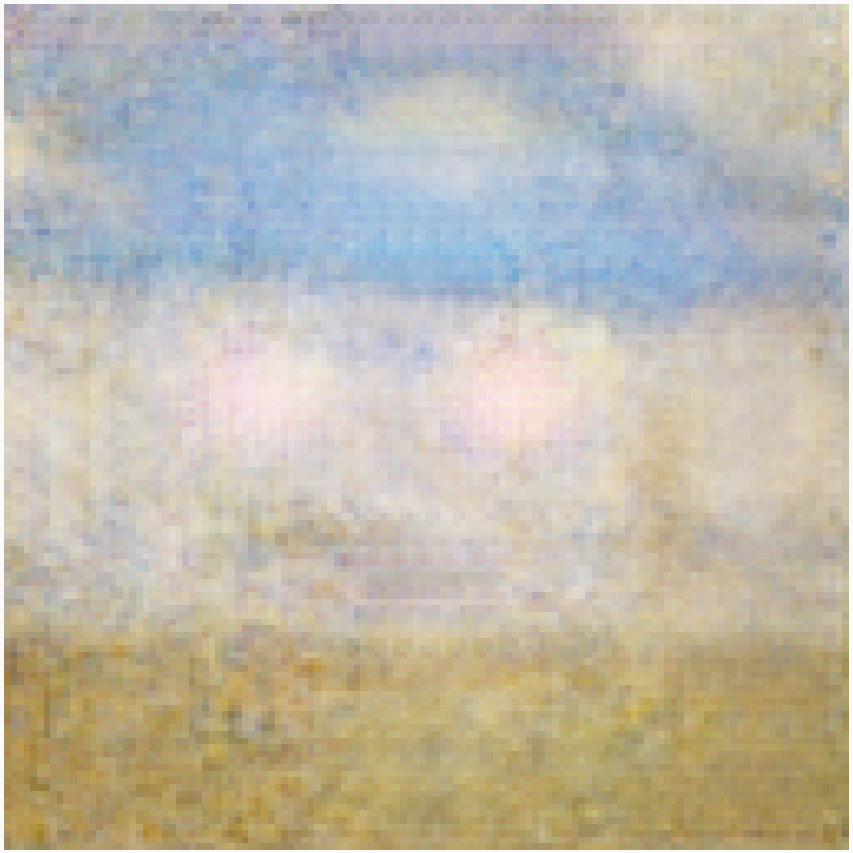
(c) Reduce LR on Plateau (0.001)







Other results



(a) Result at 160 epochs

- $\lambda_{id} = \lambda_{cyc} = 1$
- 5 training of discriminators per training of generators
- Clamping parameters of discriminators between -0.01 and 0.01

Conclusion

A Neural Algorithm of Artistic Style

- Results of Gatys et al.'s technique look very good
- Loss functions stabilize quickly
- Many improvements are now possible

- Does not generalize directly...
- ...but gives results very quickly !

CycleGAN

- Good results are very hard to obtain
- Probably caused by a dataset not big enough
- The long training time (3-5 days) made it lengthy to tweak
- Using only one GAN could be enough for the task

Comparison

	NAoAS[2]	**CycleGAN**
Scope of the tool	Specific to one style image but easily adapted	General (style of an artist) but limited (one style per training)
Training	Just needs a pre-trained model (instantaneous with tools like PyTorch)	Requires lots of time and resources
Results	Works on the fly, needs to tweak parameters each time	Better, more natural results than NAoAS (but harder to get...)

[2] A Neural Algorithm of Artistic Style

Our journey ends here... but we have only opened the first door to the world of neural style transfer !

Thank you for your attention !

References

- Jun-Yan Zhu. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. https://junyanz.github.io/CycleGAN/. 2017.
- John Glover. *An introduction to Generative Adversarial Networks (with code in TensorFlow)*. https://cutt.ly/VyTQ8jt. 2016.
- Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: *CoRR* abs/1508.06576 (2015). arXiv: 1508.06576. URL: http://arxiv.org/abs/1508.06576.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *CoRR* abs/1703.10593 (2017). arXiv: 1703.10593. URL: http://arxiv.org/abs/1703.10593.

Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, and Mingli Song. “Neural Style Transfer: A Review”. In: *CoRR* abs/1705.04058 (2017). arXiv: 1705.04058. URL: http://arxiv.org/abs/1705.04058.


Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. "Controlling Perceptual Factors in Neural Style Transfer". In: *CoRR* abs/1611.07865 (2016). arXiv: 1611.07865. URL: http://arxiv.org/abs/1611.07865.

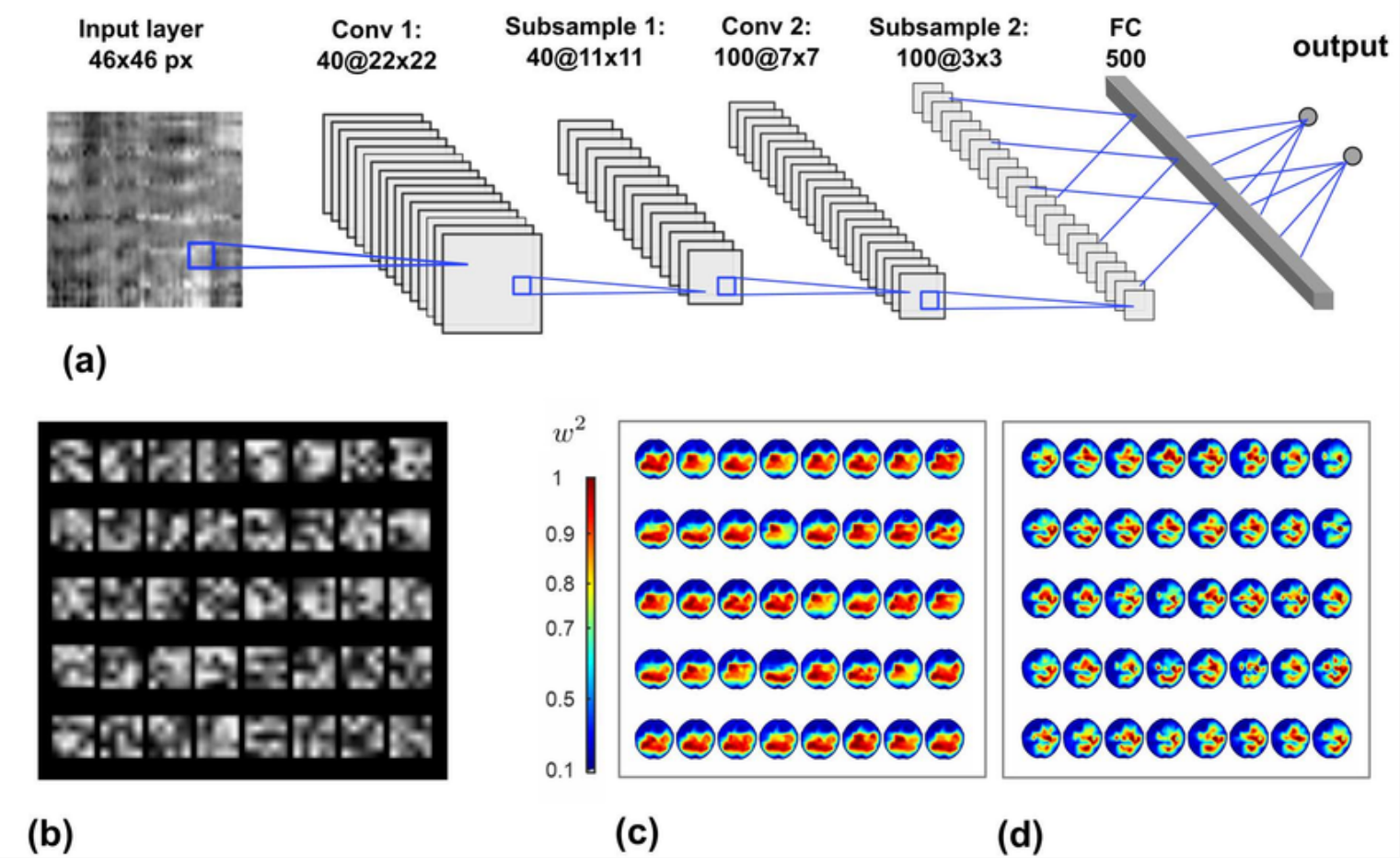

Justin Johnson, Alexandre Alahi, and Fei-Fei Li. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: http://arxiv.org/abs/1603.08155.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *CoRR* abs/1609.04802 (2016). arXiv: 1609.04802. URL: http://arxiv.org/abs/1609.04802.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012.

Additional resources

CNN structure



CNN structure

Source: `https://cutt.ly/nyL20Qo`

CNN layers

A CNN is generally composed of 3 types of layer :

- **convolutional layer** : apply a filter to its input (to get a so called *feature map*)
- **pooling layer** : aggregate values in a local region (non trainable, used to reduce size)
- **fully-connected layer** : combine features in a non-linear way

Gram matrices

In NAoAS, why the Gram matrices could represent style ?
Well, it's pretty mysterious...

This article[3] tries to demystify it.

In particular, they show that *matching the Gram matrices of feature maps is equivalent to minimize the Maximum Mean Discrepancy (MMD) with the second order polynomial kernel.*

$\longrightarrow$ The essence of neural style transfer is to match the feature distributions between the style images and the generated images.

[3] `https://arxiv.org/pdf/1701.01036.pdf`

Maximum Mean Discrepancy (MMD)

The MMD statistic can be used to **measure the difference between two distributions**.

$$
\begin{aligned}
\mathrm{MMD}^2[X, Y] &= \left\| E_x[\phi(x)] - E_y[\phi(y)] \right\|^2 \\
&= \left\| \frac{1}{n} \sum_{i=1}^{n} \phi\left(x_i\right) - \frac{1}{m} \sum_{j=1}^{m} \phi\left(y_j\right) \right\|^2 \\
&= \ldots \\
&= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{i'=1}^{n} k(x_i, x_{i'}) + \frac{1}{m^2} \sum_{j=1}^{m} \sum_{j'=1}^{m} k(y_j, y_{j'}) \\
&- \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, y_j)
\end{aligned}
$$

Auto-encoders

According to the theoretical course,[4]

An auto-encoder is a composite function made of

- an encoder f from the original space $\mathcal{X}$ to a latent space $\mathcal{Z}$,
- a decoder g to map back to $\mathcal{X}$,

such that $g \circ f$ is close to the identity on the data.

[4] https://glouppe.github.io/info8010-deep-learning/?p=lecture7.md#7

Auto-encoders and CycleGAN

According to the original paper,[5]

CycleGAN modal can be viewed as 2 auto-encoders :

- one auto-encoder $F \circ G : X \to X$ jointly with
- another auto-encoder $G \circ F : Y \to Y$

Special internal structures : they map an image to itself via an intermediate representation that is a translation of the image into another domain.

[5] `https://arxiv.org/pdf/1703.10593.pdf`

CycleGAN architecture

- Generator :

  `c7s1-64,d128,d256,[R256](6 or 9),u128,u64,c7s1-3(Tanh)`

- Discriminator :

  `C64-C128-C256-C512-C512(stride 1)-Conv2d`

`c` : `ConvInstanceNormRelu`

`d` : same but stride of 2

`R` : residual block

`u` : same as `c` but with stride 1/2

`C` : `ConvInstanceNormLeakyRelu` with stride 2