



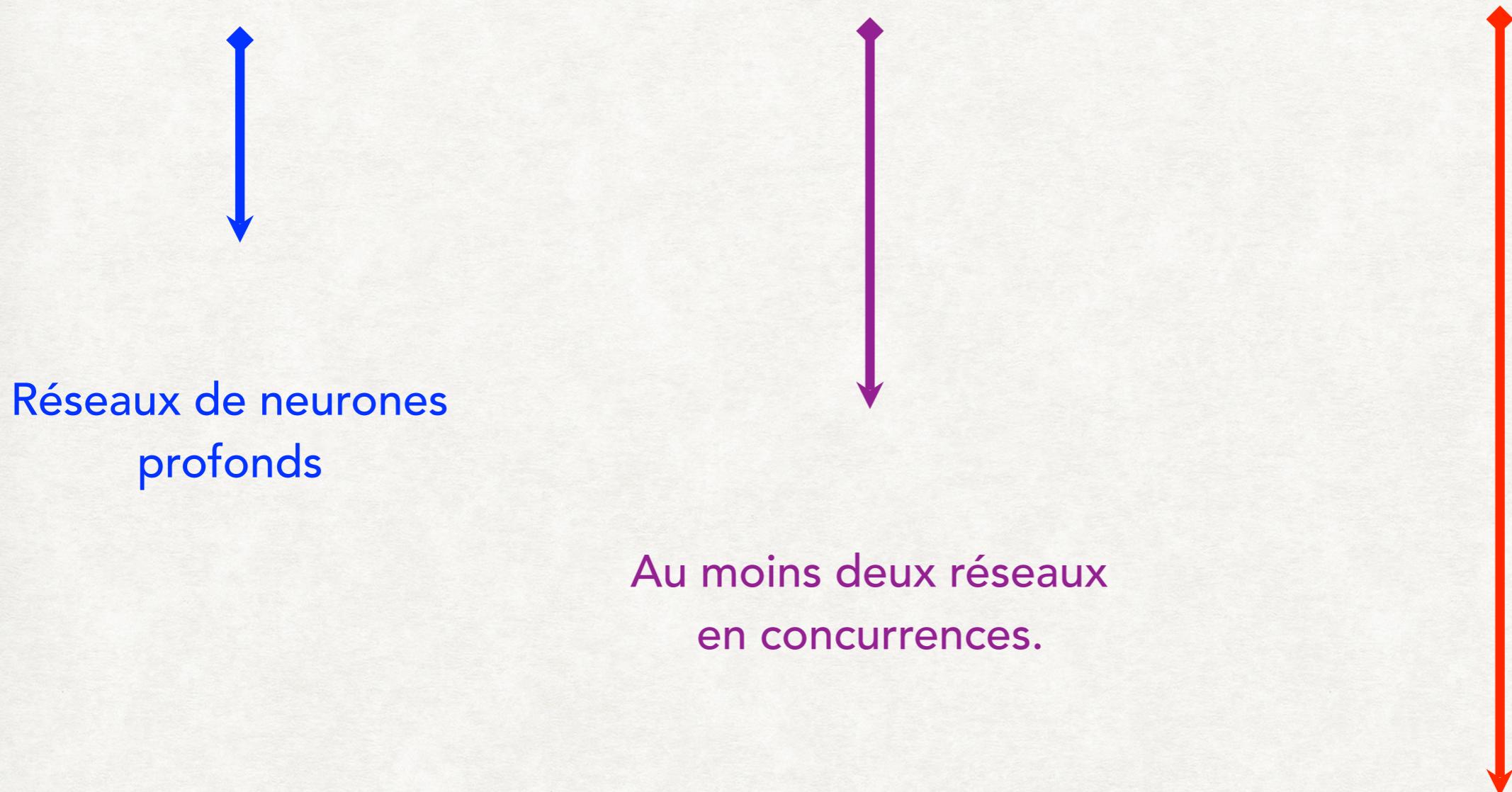
GENERATIVE ADVERSARIAL
NETWORKS (GAN)

RÉSEAUX ADVERSES
GÉNÉRATIFS

SOMMAIRE

- Présentation des GAN (principe et exemple)
- Cycle GAN
 - Exemple
 - Principe
 - Architecture
 - Mise en place
- Conclusion

RÉSEAUX ADVERSES GÉNÉRATIFS

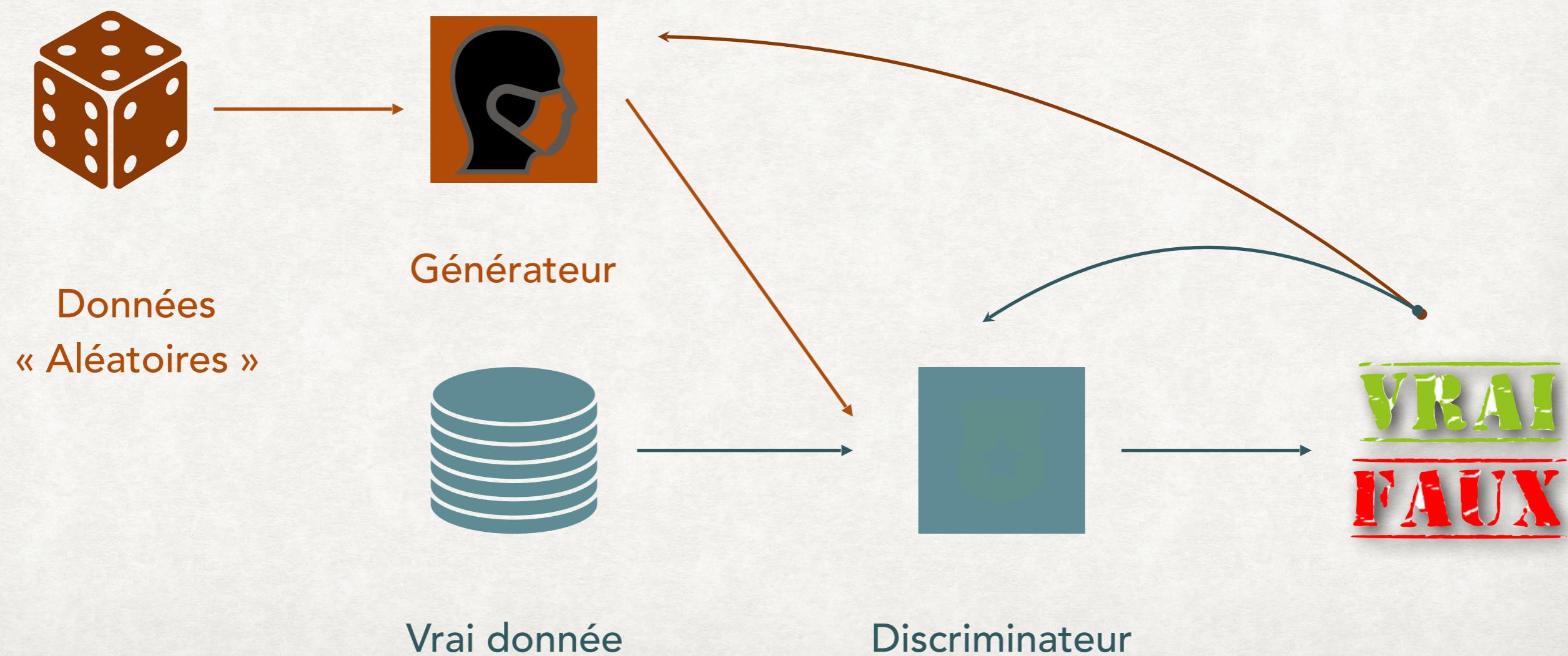


- Introduit en 2014 (Par Ian Goodfellow)
- Classe d'algorithme non-supervisé

Permet de faire de la génération (par exemple d'image)

RAG : PRINCIPE

- Deux réseaux :
 - Un discriminateur, qui doit différencier le vrai du faux
 - Un générateur, qui génère des images aussi vrai que possible (= pour que le discriminateur classe l'image comme étant vraie).



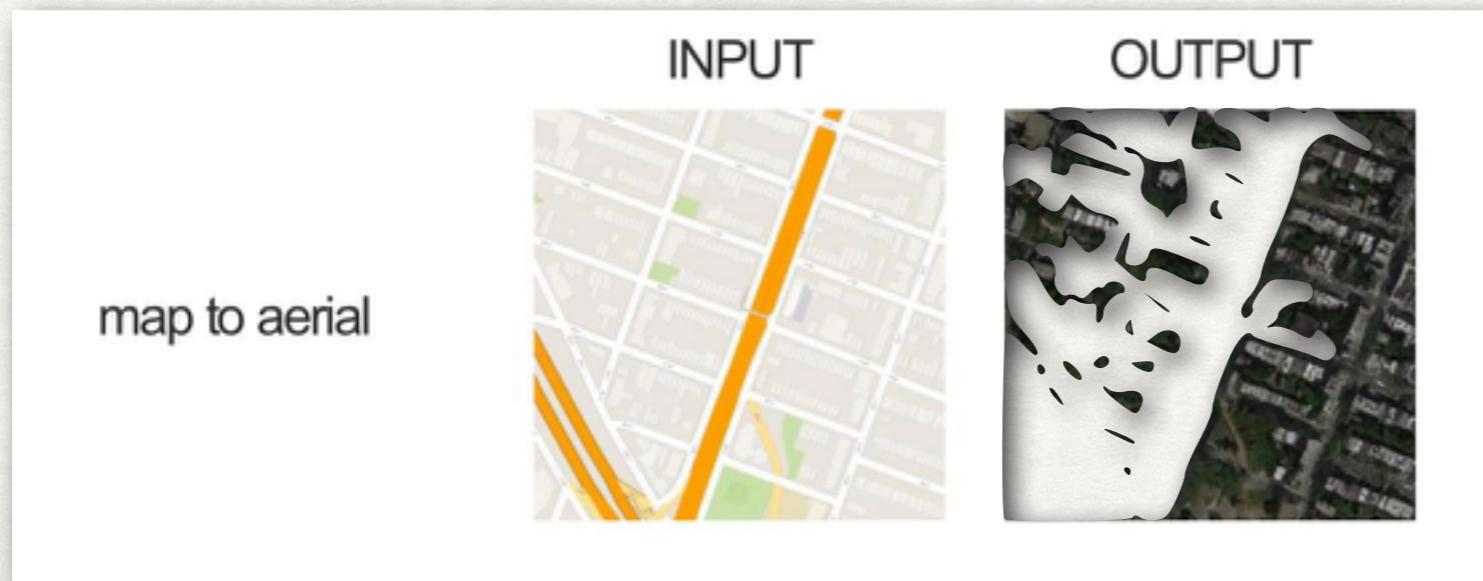
RAG : EXEMPLE D'APPLICATION

- « Tendance »:
 - thispersondoesnotexist.com —> Génération de visage
 - faceswap...

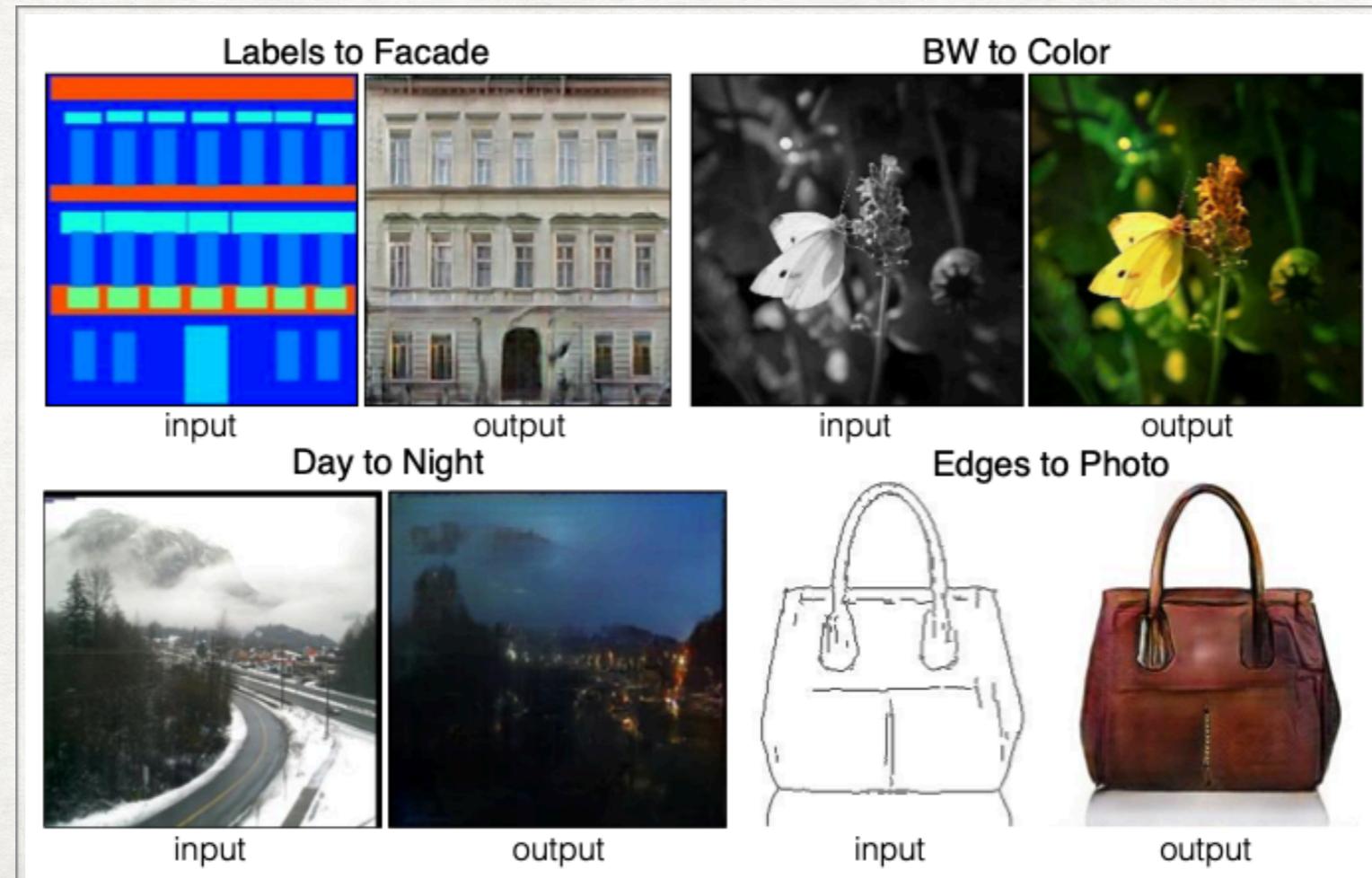
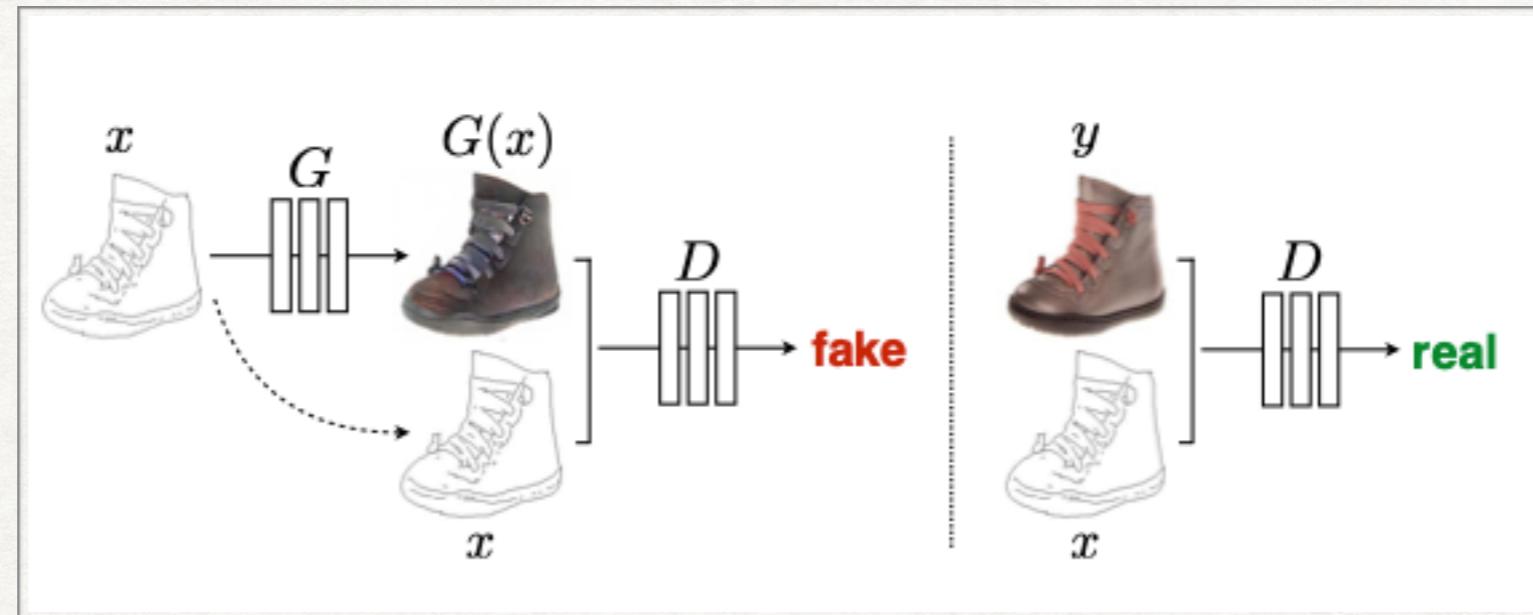
RAG : EXEMPLE D'APPLICATION

DONEC QUIS NUNC

- Plus terre à terre:
 - Génération de molécules
 - Augmentation de résolution d'image (SRGAN)
 - Génération 3D pour le design (industriel ou mode) (DCGAN ou cGAN)



VARIANTE RAG : CONDITIONAL RAG (CGAN)



CYCLE RAG

Monet ↪ Photos



Monet → photo

Zebras ↪ Horses



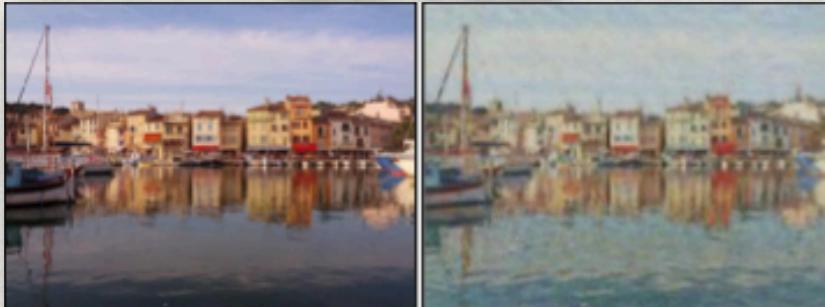
zebra → horse

Summer ↪ Winter

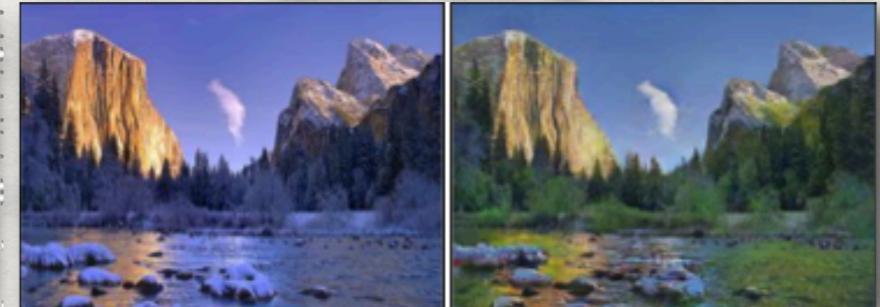


summer → winter

photo → Monet



horse → zebra



winter → summer

Photograph



Monet



Van Gogh



Cezanne

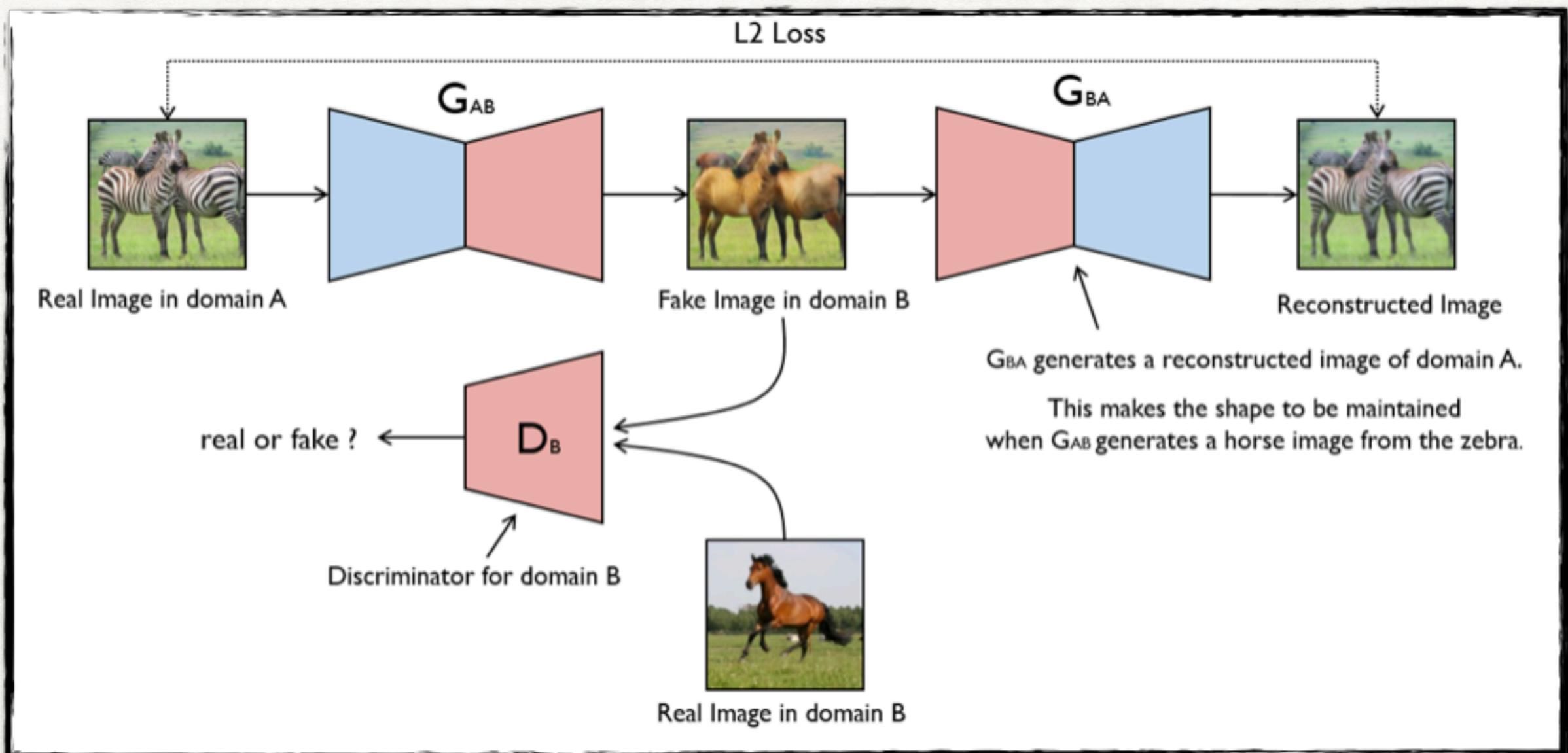


Ukiyo-e

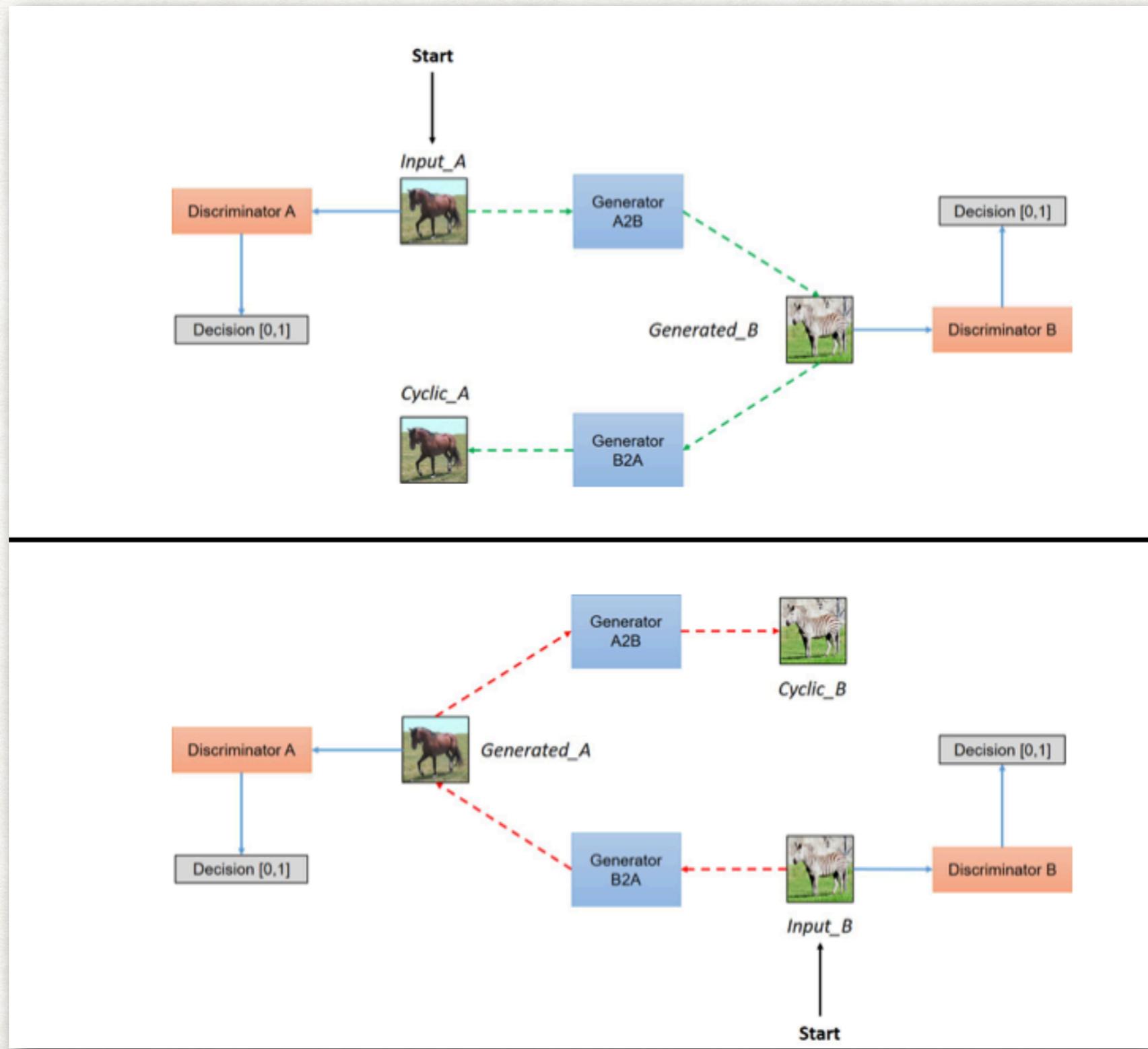
Jun-Yan Zhu Taesung Park Phillip Isola Alexei A. Efros Berkeley
AI Research (BAIR) laboratory, UC Berkeley

CYCLE RAG : PRINCIPE

4 réseaux différents : 2 générateurs et 2 discriminateurs

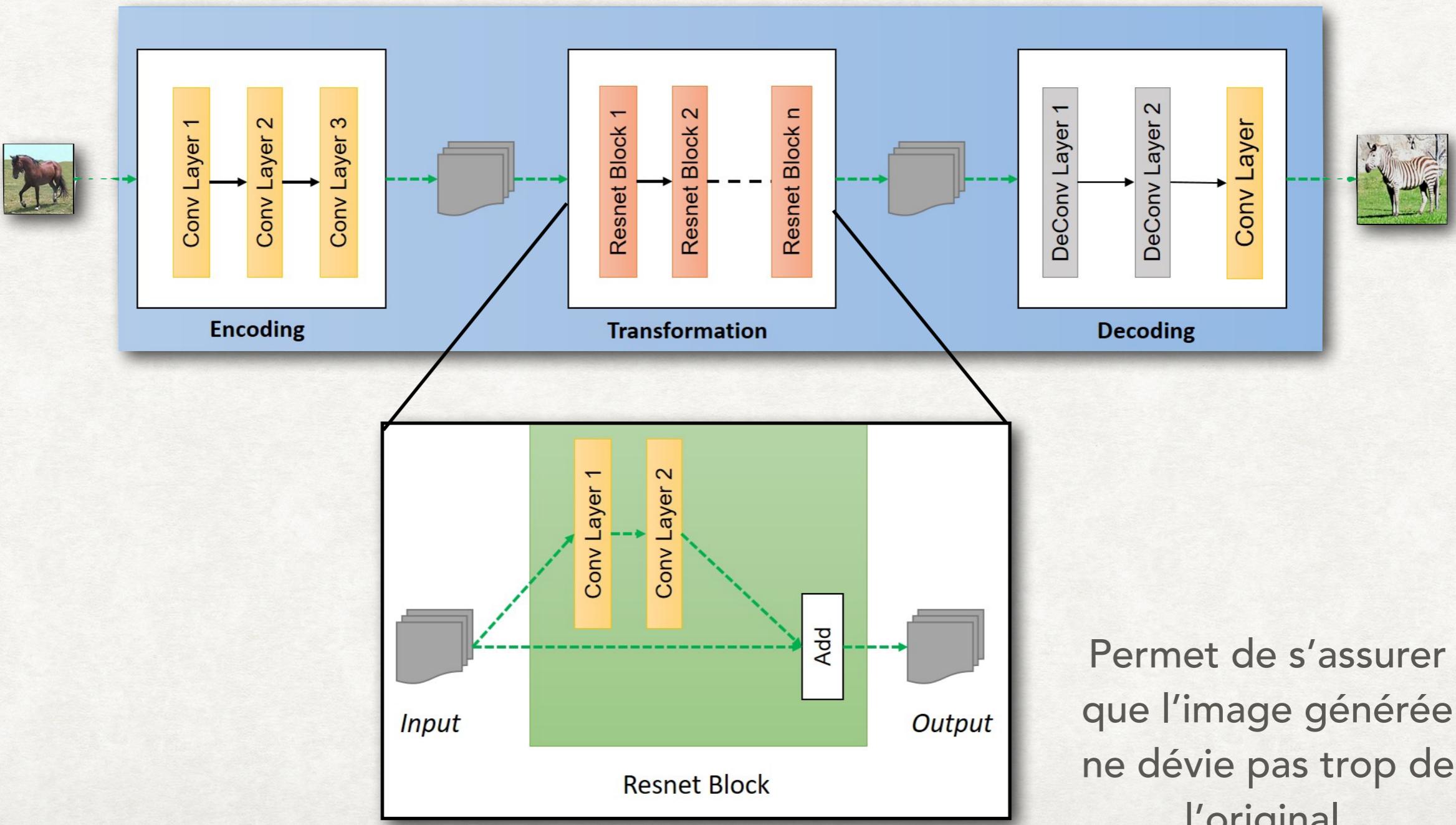


CYCLE RAG : ARCHITECTURE



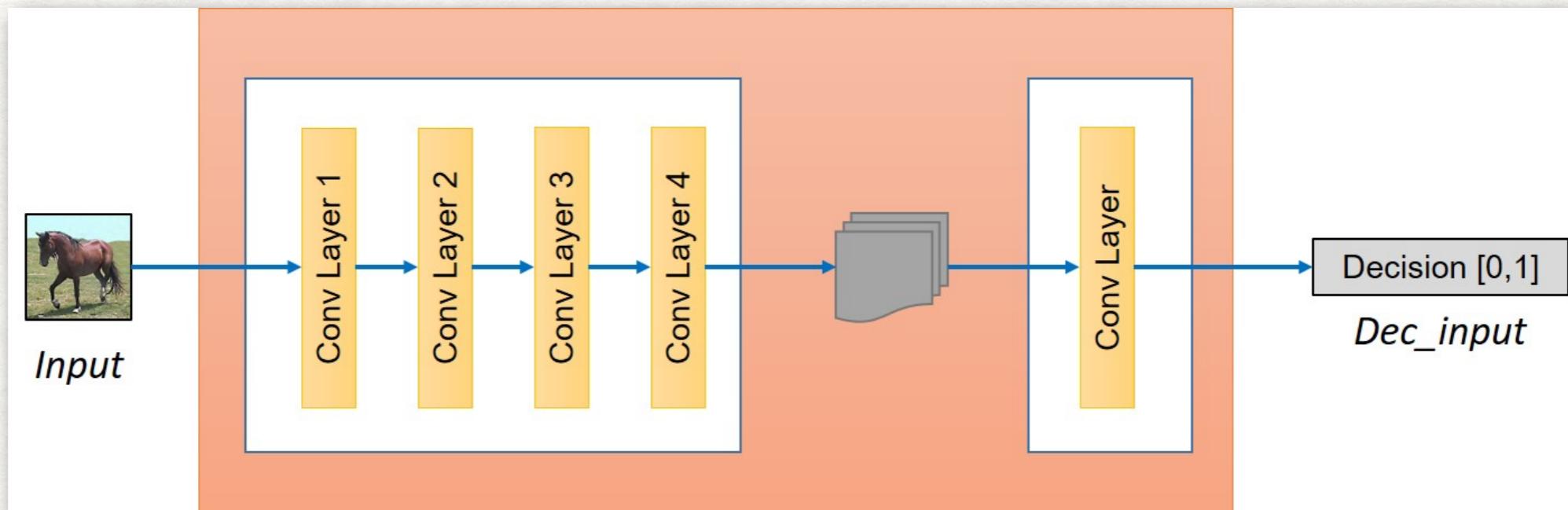
CYCLE RAG : ARCHITECTURE

- Générateurs:



CYCLE RAG : ARCHITECTURE

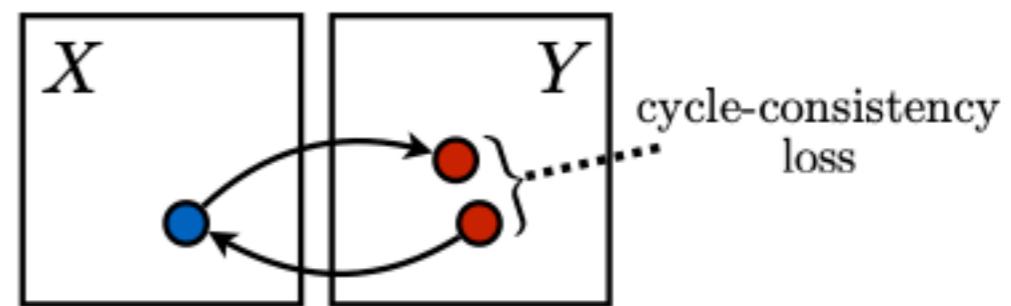
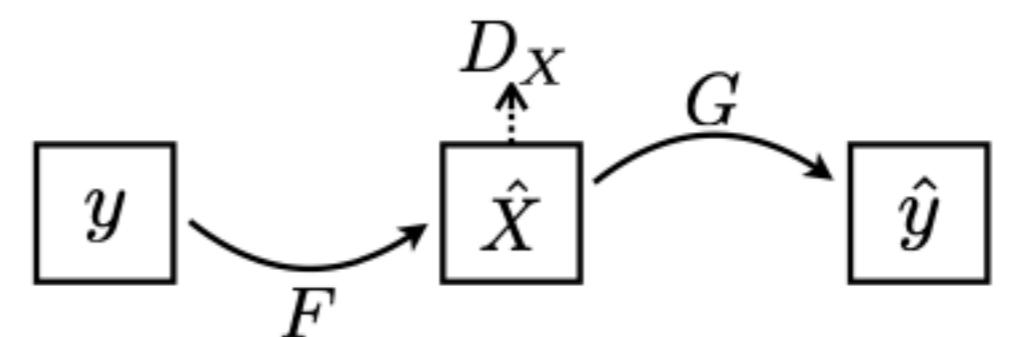
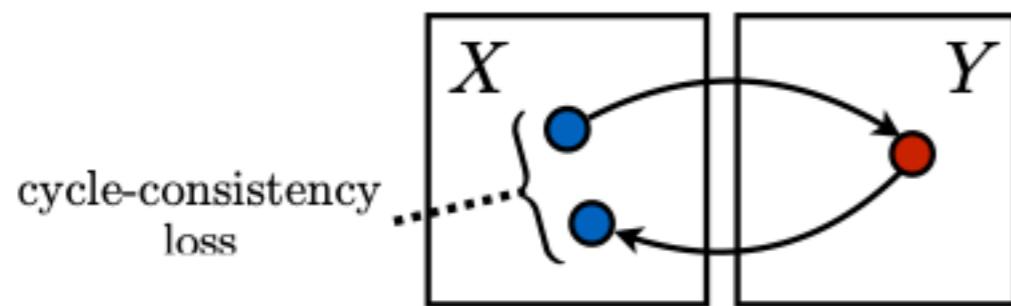
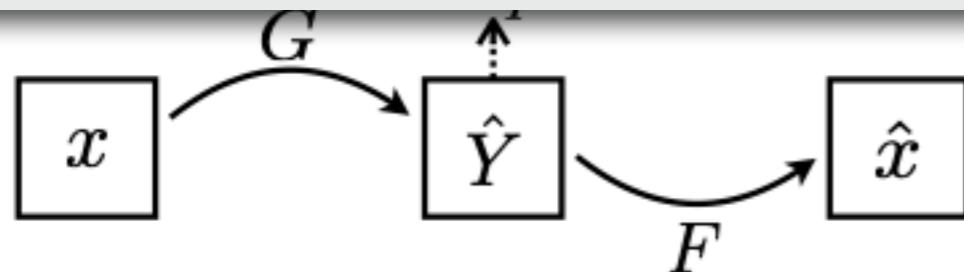
- Discriminateurs:



- LeakyRelu pour les fonctions d'activations.

CYCLE RAG : FONCTIONS DE PERTE

$$\text{Identity loss} = |G(Y) - Y| + |F(X) - X|$$

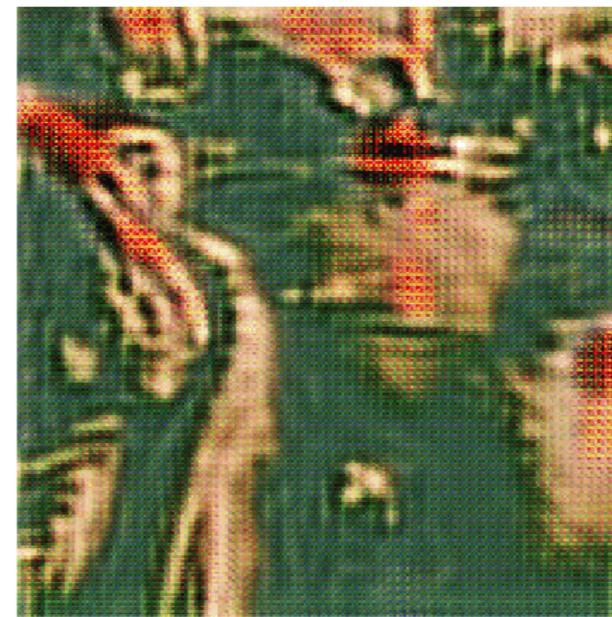
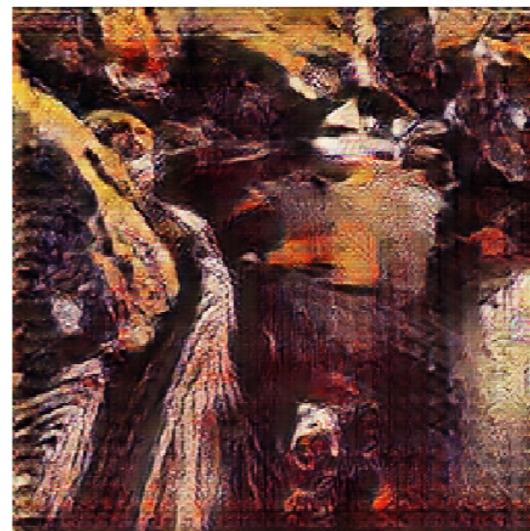
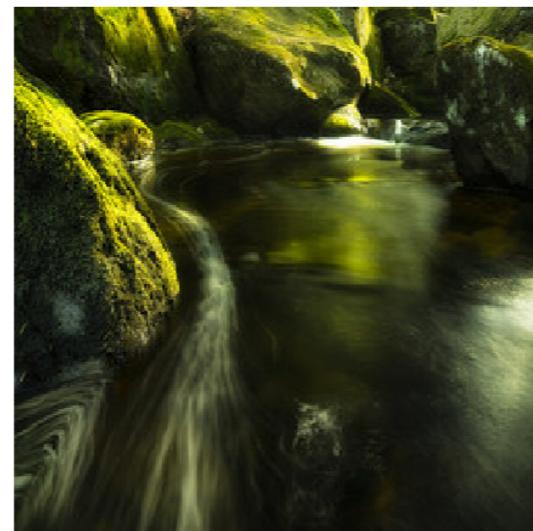


CYCLE RAG : MISE EN PLACE

- (1) Transformer des tableaux d'artistes en photo et vice-versa.
- (2) Flouter l'arrière plan d'une image et vice versa
- Implementation en Keras
- Test sur respectivement 41 et 100 epochs
- Temps par epoch variable mais de l'ordre de 5-10 min

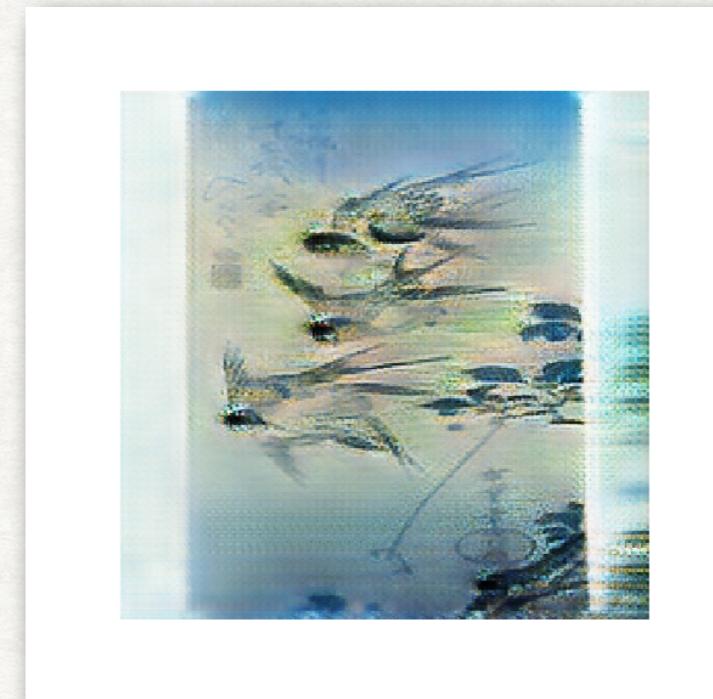
CYCLE RAG : MISE EN PLACE

- Evolution du rendu au cours du temps (photo vers dessin)



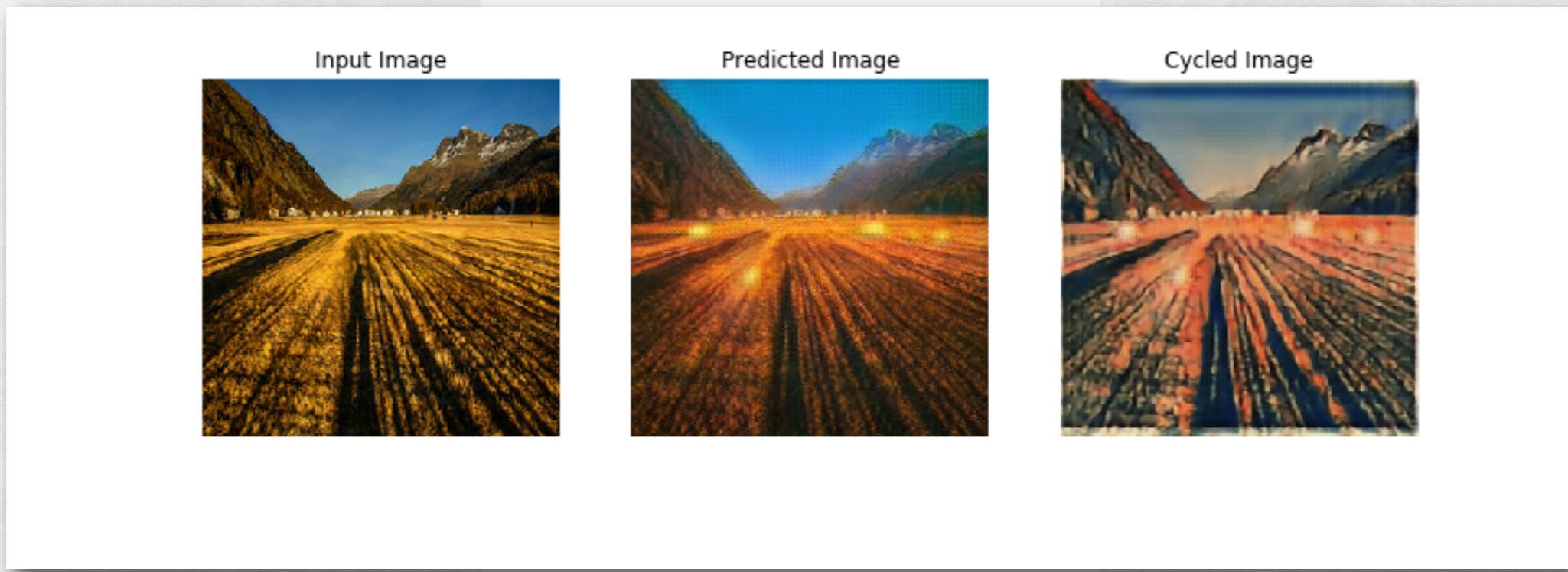
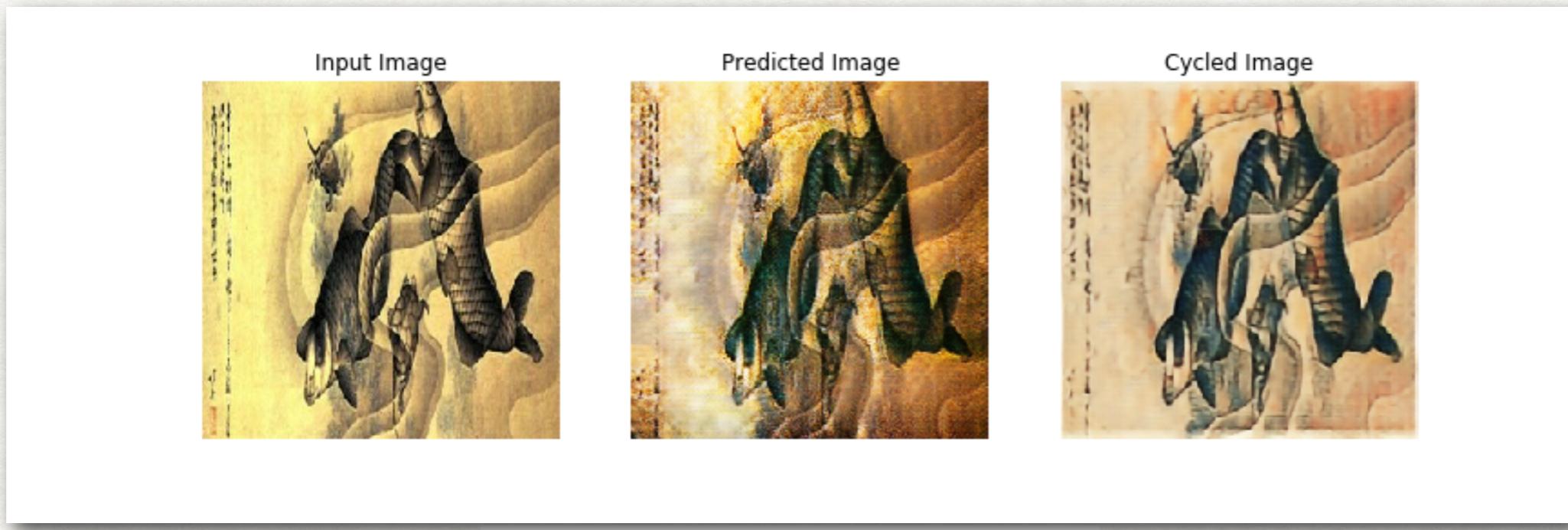
CYCLE RAG : MISE EN PLACE

- Evolution du rendu au cours du temps (dessin vers photo)



CYCLE RAG : RESPECT DES FONCTIONS DE PERTES?

- Cycled losses



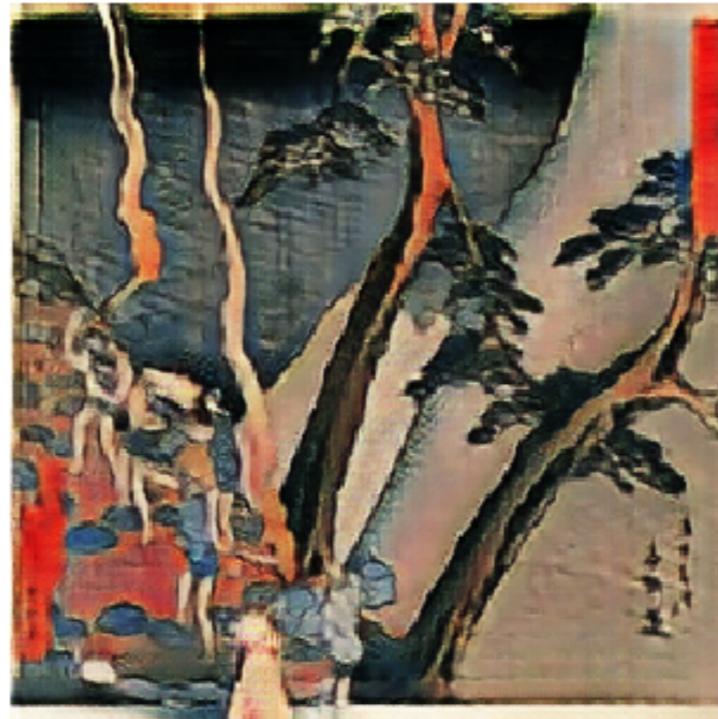
CYCLE RAG : RESPECT DES FONCTIONS DE PERTES?

- Identity losses

Input image

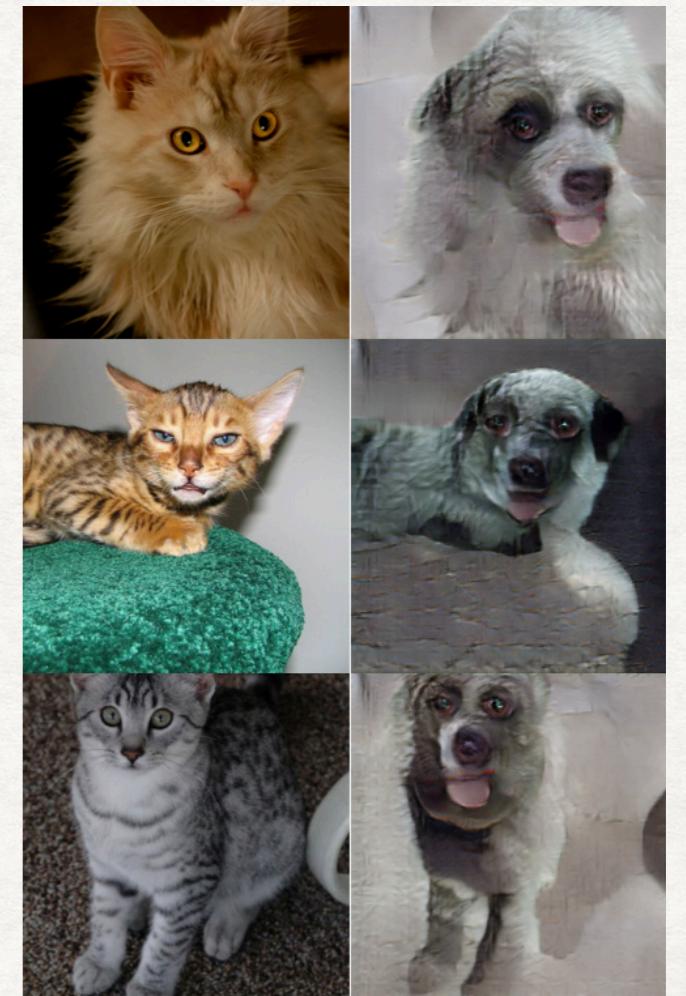


Identity image



CONCLUSION

- Des applications prometteuses, mais des résultats pas toujours satisfaisant.
- Seulement des changements d'apparence, pas de changement de structure



- Résultats non -uniforme, stabilité de la solution?
- Temps d'exécution relativement long

CONCLUSION

- Choix technologiques :
 - Tensorflow
 - Numpy, Matplotlib ...
 - Gcloud : une instance Tensorflow & Jupyter
 - Environ 1,8 \$/h
 - GPU Nvidia Tesla P100

CONCLUSION

- Choix du dataset : « iphone2dslr_flower »
- Objectif : appliquer de l'effet flou sur l'arrière plan d'une photo



SOURCE

- DCGAN : <https://arxiv.org/pdf/1511.06434.pdf>
- CycleGAN: <https://arxiv.org/pdf/1703.10593.pdf>
 - <https://hardikbansal.github.io/CycleGANBlog/>
 - https://qiita.com/itok_msi/items/b6b615bc28b1a720afd7?fbclid=IwAR3vGDvBQUjJK_-H3kkLRjz_S8D66um8uH9wV_9Q6lbl1yrK8YLqpkp1RF8#追加実験結果20170614追記7
- cGAN : <https://arxiv.org/pdf/1611.07004.pdf>