

# Deep Learning : Génération d'image à partir de schéma

Mathieu Chambard, Amine Boujemt, Yiyao Zhai, Hichem Djeghri

Lien github : Deep Learning Project

## 1 Introduction

Ce projet s'intéresse à la génération d'images de synthèse. C'est un domaine très étudié, et très en vogue en ce moment. Les différentes méthodes génératives d'images utilisent des outils de deep learning (réseaux de neurones). La génération d'images de synthèse a de nombreuses applications et elle peut se faire à partir de différents formats.

Tout d'abord, la génération peut se faire à partir de texte comme l'outil proposé par Bowen Li [1]. Le réseau prend une phrase en entrée, et ce dernier générera une image correspondant au mot clé de celle-ci.

De plus, il existe également des outils qui génèrent des prolongement d'images. C'est le cas de l'outil DALL-E qui va permettre de prolonger une image en spécifiant des informations sur ce prolongement.

Enfin, dans le cas de notre projet, on va s'intéresser à la génération d'images de synthèse à partir d'un croquis. Un outil permettant de générer des images de paysage est disponible en ligne (Test GAUGAN). L'utilisateur va pouvoir faire un croquis de son paysage, voir côté gauche de la figure 1. Il a accès à différentes couleurs qui sont reliées à des labels (montagne, ciel, rivière etc..). A partir de ce croquis, l'outil va générer une image de synthèse d'un paysage vraisemblant, voir côté droit de la figure 1.

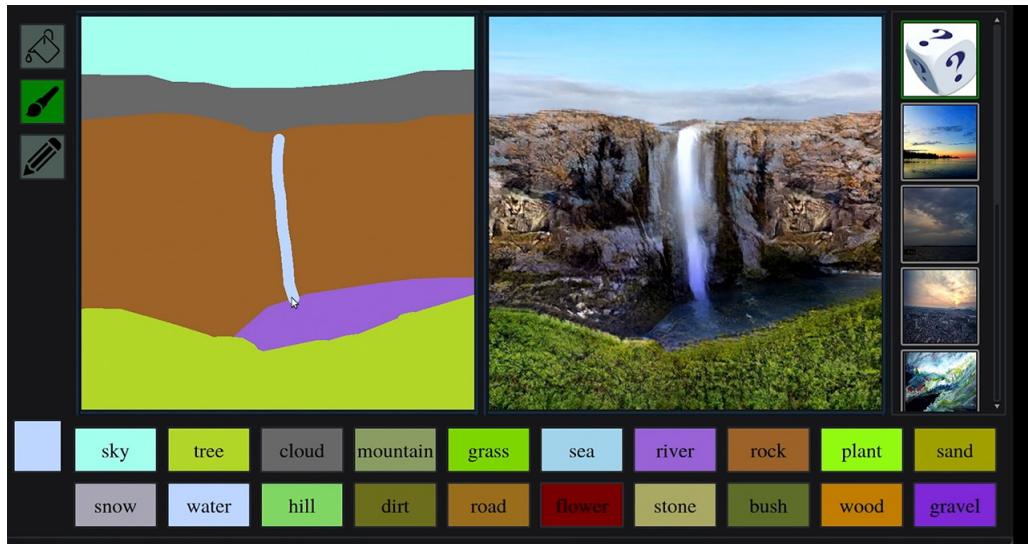


Figure 1: GAUGAN : Outil de génération d'image

L'objectif de notre projet, est donc de générer des paysages avec un style d'animé japonais à partir d'un croquis. Les paysages d'animés sont similaires dans leur conception mais possèdent un style graphique très éloigné des paysages réels, voir figure 2. Pour générer ces images, on va utiliser un réseau antagoniste génératif (GAN) qui utilise l'architecture SPADE [2] (SPatially-Adaptive (DE)normalization). Il existe déjà des modèle pré-entraîné utilisant l'architecture SPADE, cependant notre problème étant très spécifiques, nous ne pouvons pas nous contenter de simplement réutilisé un model préentraîné. Nous avons le choix entre deux options, soit on entraîne notre modèle

avec nos propres données, soit on réentraîne un modèle pré-entraîné avec notre propre dataset(fine tuning).

Ce rapport est constitué de trois parties. Une première partie où on abordera l'architecture du modèle offerte par GAUGAN, puis on détaillera le processus de génération de nos données ainsi que la phase d'entraînement. Pour finir, on fera un bilan de notre méthode et on affichera les différents résultats.



Figure 2: Paysage d'animé

## 2 Méthode SPADE (GAUGAN)

Dans cette partie, nous nous concentrerons sur la méthode choisie GAUGAN et les raisons de ce choix. Dans nos recherches nous nous sommes concentré sur les modèles génératif de référence (generative adversarial network GAN). Lors de nos recherches nous avons fait la découverte des GAN conditionnelles et plus spécifiquement des modèles de type GAUGAN.

### 2.1 Présentation général de la méthode

GAUGAN est un modèle génératif d'image. En entrée du modèle, il faut fournir des fragmentations (des croquis). Un exemple de fragmentation est donné dans la figure 3. Une fragmentation est une image avec un nombre de couleur égale au nombre de label défini dans le modèle. Chaque couleur correspond donc à un certain label. Dans la segmentation de la figure 3, le rouge correspond au label "personne", le bleu au label "voiture", le vert au label "arbre" etc.. A partir de cette image, GAUGAN va générer une image réaliste en remplaçant la couleur du label par la représentation qu'il a appris dans la phase d'entraînement.

Pour qu'il soit capable de générer une image, un réseau de neurones doit être entraîné. Il existe des modèles pré entraînés. Deux options s'offrent à nous, soit nous faisons du fine tuning sur un modèle déjà entraîné sur des paysages réels, soit nous entraînons un nouveau modèle. En vue de notre tâche spécifique, nous avons décidé d'entraîner notre propre modèle.

Pour entraîner le réseau de neurones de GAUGAN, il faut les entrées et les sorties du problèmes. Il faut donc des images réelles avec leurs fragmentations labellisées.

Nous faisons donc face à notre premier problème. En effet, il est dur de trouver des jeux de données avec des images et leur segmentation. On peut trouver ce genre de jeux de données



Figure 3: Segmentation d'une image

pour des paysages, des photos de rue, de maison etc..( Voir ADE20k, Coco stuff, Cityscape,...). Cependant, il n'y a pas de base de données de paysage d'animé avec les segmentations associées. On a trouvé des bases de données de paysages d'animé (DataSet Anime Landscape) mais pas la fragmentation associée. Dans notre cas, nous n'avons pas le temps de faire la fragmentation à la main. Nous avons donc chercher une méthode de segmentation d'image pour préparer notre jeu d'entraînement. La création de ce jeu de données est expliquée dans la partie 3. Dans la suite de cette partie, nous allons étudier plus précisément GAUGAN.

## 2.2 GAN et GAUGAN

Une des principales différences entre un GAN classique et GAUGAN est que GAUGAN utilise un GAN conditionnelle au lieu d'un modèle GAN classique. La différence entre un GAN classique et conditionnelle est expliquée sur la figure 4. Un GAN classique a besoin seulement de bruit (image/son/text...) en entrée du modèle. En apprenant les distributions de probabilité, le modèle sera alors capable de générer une sortie ressemblant plus ou moins aux données qu'il a vues pendant l'entraînement.

Dans notre cas, un GAN classique aurait pu suffire pour que l'on puisse générer une image à partir d'un croquis. Cependant, les GAN conditionnelles vont permettre d'être plus précis. En effet, le GAN conditionnel a besoin d'une information supplémentaire(condition y sur la figure 4). Dans notre cas, la condition y sera les labels des différentes zones (chaque couleur sur notre croquis). Cette information supplémentaire va forcer le réseau à générer telle chose dans telle zone. On évite ainsi les erreurs dues à l'interprétation de l'image par le modèle (Il ne pourra pas nous générer de l'eau alors qu'on a demandé de l'herbe). Cela permet également au modèle de converger plus vite.

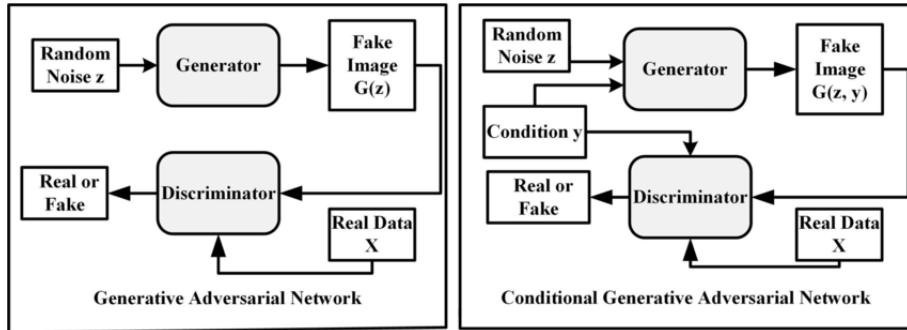


Figure 4: GAN vs Conditional GAN

### 2.3 SPADE normalization

Une autre différence est la méthode utilisée pour la normalisation. Dans les techniques de normalisation classique, cette dernière est appliquée inconditionnellement et de la même façon sur chaque pixel de l'image. SPADE est la méthode de normalisation utilisée dans cet outil. Elle ne va pas normaliser de manière uniforme l'image. En effet, quand on a une normalisation uniforme, de nombreuses zones avec des pixels équivalents vont être toutes normalisées à 0( ou proche de 0). La méthode SPADE permet de localiser ces espaces, et d'appliquer une dénormalisation. Cela permettra d'avoir une sortie plus réaliste lorsque l'on a une entropie faible (peu de label). Pour plus de détails et d'information, vous pouvez consulter directement le papier de nvidia ou ce notebook qui explique le fonctionnement de SPADE en détails. Une fois la méthode choisie, il ne nous reste plus qu'à créer le dataset et entraîner notre modèle.

## 3 Création des dataset et entraînement des modèles

Pour notre dataset, nous avons besoin d'image de paysage d'anime, avec une segmentation de l'image correspondante. On fait face à notre premier problème, il y a peu de dataset de paysage d'anime, et aucun avec les segmentations correspondantes. Nous avons donc cherché un modèle de segmentation de paysage.

La création du dataset fut une tâche aussi complexe que le projet en lui-même. Nous avons dû tester différents modèles de segmentation entraîné sur des paysages.

### 3.1 Recherche d'un modèle de segmentation d'image

La segmentation d'un paysage est loin d'être une tâche trivial (reflet du ciel dans l'eau, intensité de la lumière, zone d'ombre...)

Nous avons testé un premier modèle de segmentation PYLC. On constate sur la figure 6 que le modèle PYLC n'est pas du tout adapté à nos besoins. Après quelques recherches supplémentaires, nous avons trouvé un modèle avec des résultats convenables.

La segmentation du second modèle CSAILVISION fut beaucoup plus prometteuse.

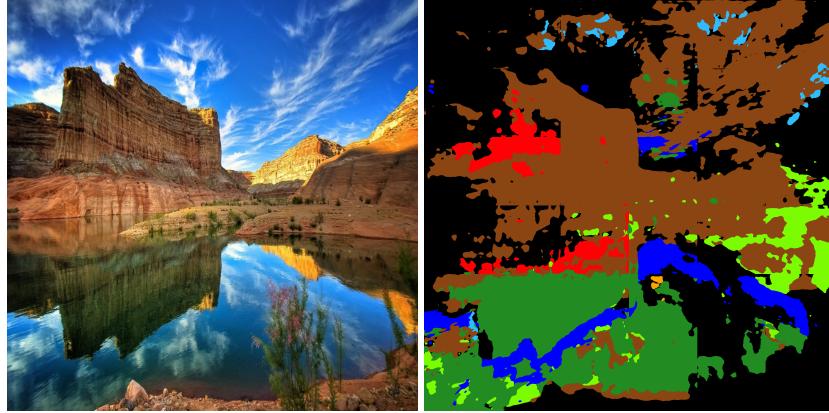


Figure 5: Segmentation d'un paysage avec le modèle PYLC



Figure 6: Segmentation d'un paysage avec le modèle CSAILVISION

### 3.2 Création de notre dataset

Maintenant que nous avons trouvé une méthode de segmentation d'image qui marche très bien sur des paysages réels, on va pouvoir passer au paysage d'animé. Nous avons récupéré un dataset sur Kaggle comportant des images de paysages animés. Nous avons appliqué le modèle de segmentation CVISION sur ces images afin de créer nos fragmentations pour notre jeu de données, et enfin obtenir notre dataset. Cependant de nouveaux problèmes surgissent

### 3.3 Problèmes rencontrés

Tout d'abord, la segmentation sur les paysages d'anime ne fonctionne pas aussi bien que sur des paysages réels.

De plus, la taille de notre dataset est limitée (nous n'avons que 400 images au lieu de milliers).

Le problème de segmentation est facilement explicable. Les paysages d'animés sont très différents graphiquement des paysages classiques. Il est difficile pour le modèle CVISION entraîné sur des paysages classiques (sans style animé) de segmenter l'image correctement. Par exemple, lors de son entraînement le modèle de segmentation n'a jamais vu de cerisier et est donc incapable de les identifier (figure 7). Le modèle a fait correspondre le cerisier et le ciel.

Cela pourra poser problème plus tard lors de la génération d'une image. Notre modèle pourrait potentiellement créer un ciel contenant un cerisier. Pour résoudre ce problème, il faudrait donc reprendre le modèle Cvion et faire du finetuning sur ce dernier. Cependant cela nous ramène au problème initial, pour entraîner le modèle à segmenter des paysages d'animés, il nous faut un dataset avec des paysages d'animé segmenter.

Concernant le problème de la taille du dataset, il peut être résolu en créant un dataset à la main

mais ceci prend beaucoup de temps, et ce n'est pas l'objet de ce projet donc nous avons décidé de garder uniquement ce dataset.

Malgré le fait que notre dataset n'est pas optimal, nous allons quand même essayer d'en tirer les meilleurs résultats possibles, et nous allons également le comparer à un autre modèle que nous avons entraîné nous même sur un dataset plus consistant étant constitué de paysage réel qui sont beaucoup plus simple à trouver.

Nous pouvons désormais entraîner notre modèle avec ces dataset.



Figure 7: Segmentation d'un paysage d'anime avec le modèle CSAILVISION

### 3.4 Entrainement du modèle

Pour générer nos images d'animé, il faut qu'on entraîne notre propre modèle. Nous avons donc utilisé la fonction train.py de GAUGAN.

Optimalement, pour un bon entraînement et des bonnes images générées, il faut beaucoup d'images (des milliers) avec des fragmentations adaptées. De plus, il est conseillé d'entraîner le modèle pendant 2 à 3 semaines avec 8 cartes graphiques de 32go.

Dans le cas des animés, nous avons peu d'images et des fragmentations médiocres. De plus, nous n'avons pas les ressources informatiques nécessaires. Les images générées ne vont donc pas être très réalistes.

Concernant la génération d'images standard, nous avons un nombre d'images d'entraînement correct (3655) et les résultats de segmentation sont plutôt bons dans l'ensemble. Nous avons entraîné notre modèle sur 50 epoch sans batch sur une seule carte graphique RTX 3060. L'exécution a pris 24h, ce qui est très loin des temps de calcul recommandés pour de bon résultats.

On va voir dans la partie suivante les résultats de génération de nos deux modèles.

## 4 Présentation des résultats

Dans cette partie, nous allons vous présenter les résultats de notre méthode. On va voir les résultats des différents modèles (animés et paysage classique). Ensuite on parlera de ce qui doit être amélioré pour des résultats de meilleure qualité.

### 4.1 Résultats nvidia

Sur la figure 8, on peut voir le genre de résultats de la méthode sur un modèle énormément entraîné. On peut voir que même avec beaucoup de données et d'epoch durant l'entraînement du modèle, on obtient une image de moyenne qualité.

### 4.2 Nos résultats

Sur la figure 9, on a affiché les images générées par les différents modèles. On a pris des données de test, on a donc accès à la véritable image. On voit tout d'abord la vraie image puis l'image générée par le modèle sur les paysages classiques et enfin celle générée par le modèle entraîné sur des paysages animés.

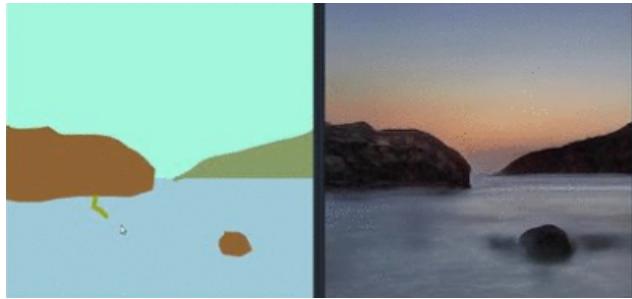


Figure 8: Résultat obtenu par nvidia

On voit que lorsque l'image est compliquée (première photo), le modèle a du mal à générer quelque chose de réaliste. On retrouve également ce problème sur la figure 10 où il y a beaucoup de détails.

Globalement, les résultats du modèle entraîné sur les paysages classiques sont plutôt bons. Ils sont réalistes dans l'ensemble mais un peu pixelisés.

Pour les images animées, les images sont prometteuses mais restent très moyennes. On reconnaît le style et les couleurs d'animé mais on voit que le modèle n'est pas assez entraîné.

Globalement, les résultats sont plutôt prometteurs. Mais plusieurs choses doivent être changées pour de meilleurs résultats.

Tout d'abord, nous avons très peu d'images. Pour le dataset d'animé nous avons seulement 389 images au lieu de millier. Il faudrait donc, dans un premier temps, réussir à collecter ces données.

Dans un second temps, on a un problème de fragmentation au niveau des images d'animés. En effet, les paysages animés possèdent un style graphique bien particulier avec des réflexions de lumière bien particulière et des couleurs particulières. Il faudrait donc effectuer à la main un certain nombre de fragmentation afin d'entraîner un modèle qui va faire de la bonne fragmentation d'images d'animés. C'est une lourde tâche à effectuer et qui coûte énormément d'argent.

Pour finir, dans notre cas, nous avons un problème de ressource informatique. Nous avons exécuter la méthode sur nos machines avec une seule carte graphique dans des temps raisonnables (1 jour max). Il faudrait donc louer beaucoup de machines et pouvoir faire tourner l'entraînement pendant 2/3 semaines.

En rectifiant ces 3 points, on pourrait avoir des résultats très satisfaisants en vue des résultats actuels.

## 5 Conclusion

En conclusion, ce projet a eu pour but de générer des images à partir de dessins. On a tout d'abord dû utiliser un modèle de segmentation pour créer notre dataset d'entraînement afin d'entraîner notre propre modèle, puis nous avons entraîné deux modèles différents (un sur des paysages réels, et un autre sur des paysages d'anime). Enfin nous avons utilisé ces modèles pour générer des images que le modèle n'a jamais vues durant son entraînement.

Nous avons essayé de ré-entraîné un modèle déjà pré-entraîné pour comparer avec nos résultats, mais sans succès.

Il y a énormément de pistes d'amélioration à explorer, dans un premier temps il faudrait étudier la convergence de notre modèle (est ce qu'ils continuent à apprendre, ou sommes nous dans une situation d'overfitting). Dans le cas où notre modèle continue à apprendre, il nous suffirait de réaliser plus d'epoch. Les résultats obtenus en utilisant un modèle entraîné avec moins d'époque étant moins bon, il est très peu probable que nous soyons dans une situation d'overfitting.

Une autre piste d'amélioration serait de changer notre dataset. Dans les résultats obtenus nous observons que notre modèle est très bon pour générer du ciel (présent sur la majorité des photos d'entraînement) mais moins bon pour générer une montagne (montagne très peu présente dans le dataset). Il pourrait donc être intéressant d'étudier la répartition des label lors de la segmentation

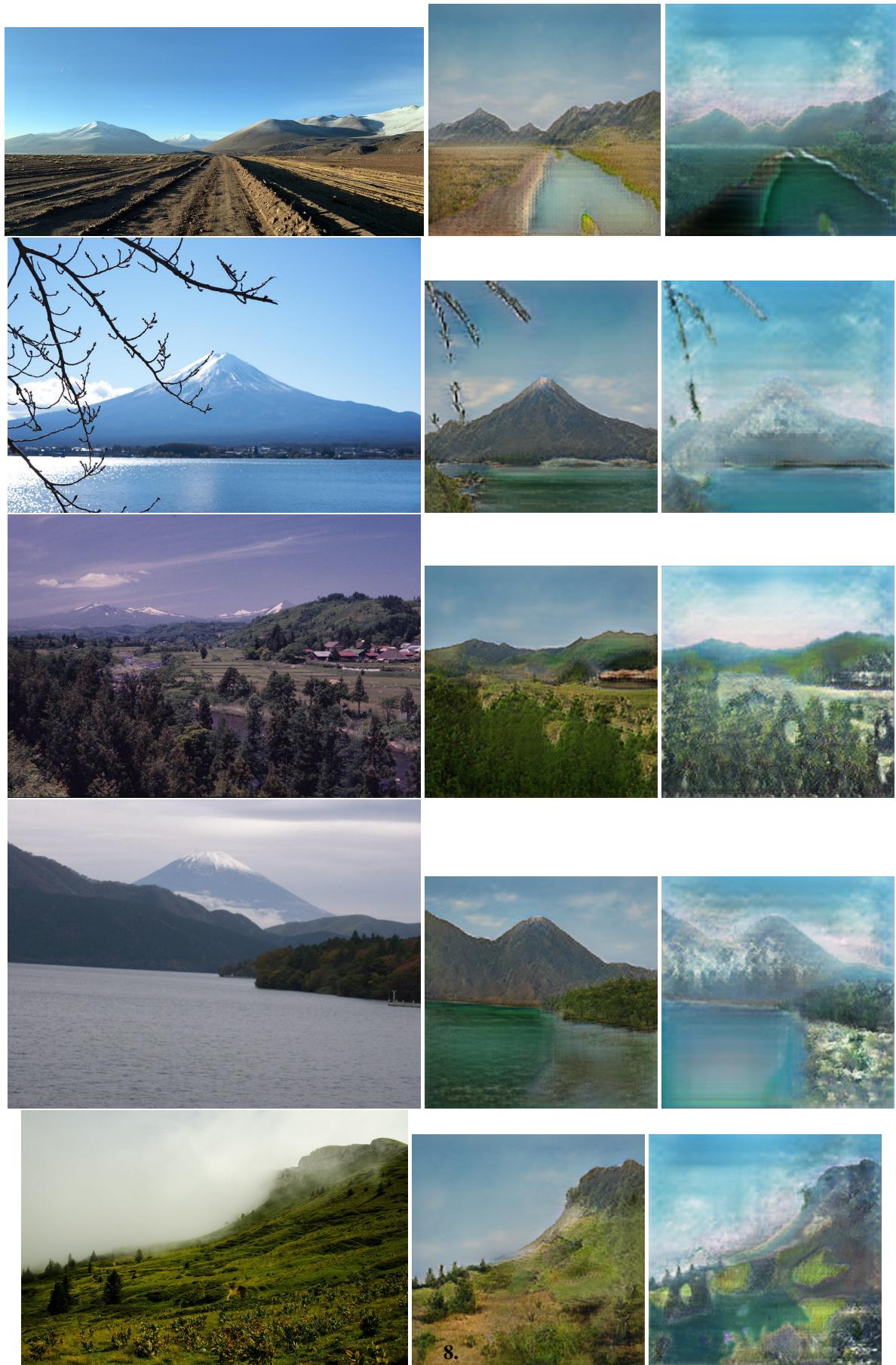


Figure 9: Génération d'image avec GauGAN

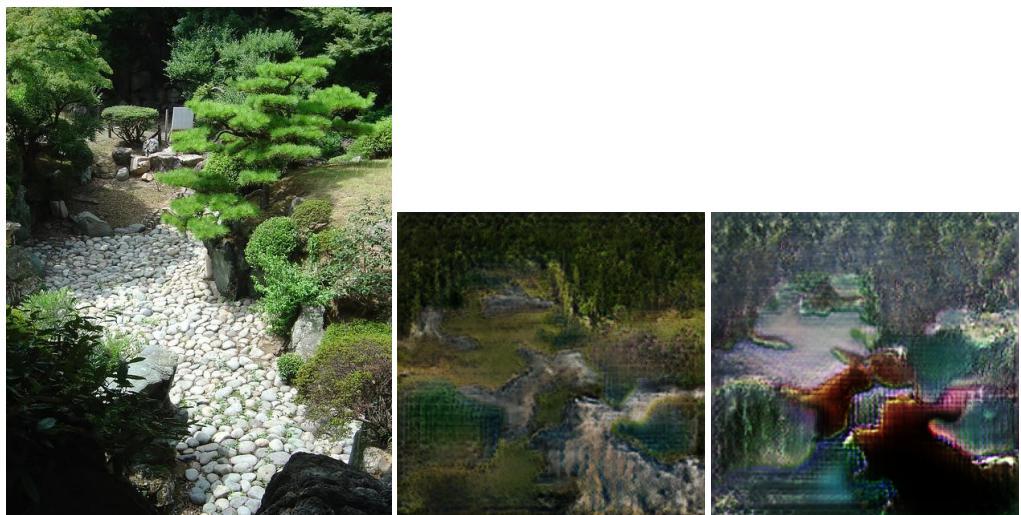
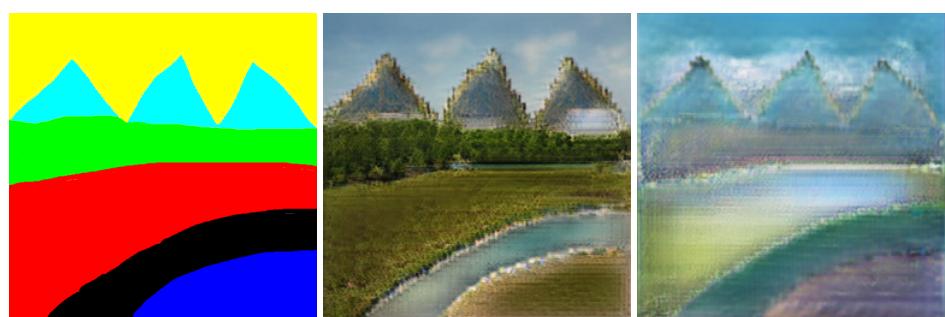


Figure 10: Génération d'image avec beaucoup de détails



des images ( nous avons la répartition pour chaque image individuel, il nous suffira donc juste de faire une moyenne).

Deux opérations spécifiques à la génération d’images d’anime peuvent être réalisées. Premièrement, il faudrait augmenter la taille du dataset qui est de seulement (400 images). Deuxièmement, il faudrait entraîner un modèle de segmentation sur des paysages d’anime pour avoir des fragmentations fidèles.

## References

- [1] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip Torr. Controllable text-to-image generation. *Advances in Neural Information Processing Systems*, 32, 2019.
- [2] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.