RCP211 : Traduction d'images

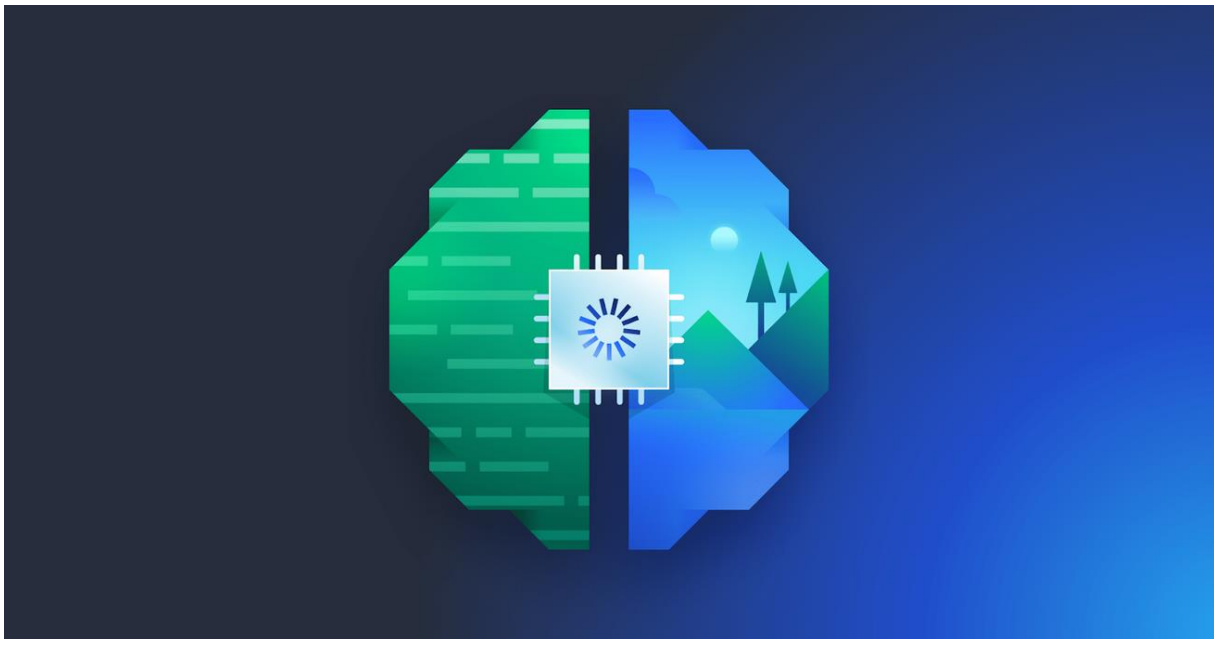


Table des matières

I- Présentation du projet

L'objectif de ce projet est de créer une image satellite qui pourrait être réaliste à partir d'une image grossière. Pour mettre en place cette traduction d'image j'ai à ma disposition deux jeux de données :

- Inria Aerial Image Labeling : qui est un jeu de donnée mono-classe. Dans celui-ci uniquement le tracé des structures sont représentées distinctement l'image grossière de départ, les autres éléments ne sont pas distingués.
- LandCover.ai : qui est un jeu de donnée multi-classe. Ce dataset fait une distinction entre : les bâtiments, les forêts, les zones d'eau et les routes, pour les autres éléments ne sont pas distingués.

Nous pouvons contacter que ces datasets ne contiennent pas beaucoup d'éléments, quelques dizaines d'images, ce qui peut sembler peu. Mais à la suite des traitements qui seront présentés dans le rapport, nous allons pouvoir en augmenter la quantité et la qualité.

II- La problématique

La problématique de ce projet est de trouver une solution permettant de faire une synthèse d'images satellites à partir de cartes. Nous savons qu'aujourd'hui les Generatif Adversarial Network (GAN) sont états de l'art en ce qui concerne la génération d'image. Et qu'il existe différentes solutions pour réaliser des conditionnal GAN (cGAN) comme Pix2Pix pour réaliser de la traduction d'image. Nous allons donc tester des solutions pour générer ces images et aussi voir l'importance d'u choix du jeu de données.

III- Cas mono-class

A- Le dataset

Comme écrit précédemment le dataset mono-classe est Aerial Image Labeling.Ce dataset possède deux types d'images :

- Des images minimalistes qui représentent deux classes :
  - Les bâtiments
  - Les non-batiments
- Des images aérienne orthorectifiées qui sont les équivalences des images minimalistes

Ces images sont des représentations de différentes villes telle que Viennes, Chicagi, San Francisco …

La composition du dataset est la suivante :

- 180 paires d'entrainement
- 180 images aérienne orthorectifiées qui n'ont pas leur équivalence en minimaliste

Ce manque au niveau des données de test est expliqué par l'objectif du dataset qui n'a pas été construit pour la traduction d'image minimaliste en image en prise aérienne.

La dimension des images est de 5000*5000.

B- Choix du modèle

Comme présenter dans la partie II- La problématique, nous sommes dans un problème de génération d'image à partir d'une condition cGAN. J'ai alors choisi de traiter ce problème avec le cGAN Pix2PIx. Pour mettre en place ce modèle je me suis basé dans un premier temps du tutoriel de tensorfflow pix2pix : Traduction d'image à image avec un GAN conditionnel car il permet de mettre en place un modèle pix2pix de manière efficace. Ce qui va enclencher un certain nombre de prétraitement nécessaire pour que le modèle fonctionne.

C- Le prétraitement

Maintenant, que nous avons à notre disposition un datatset, nous allons pouvoir commencer la préparation des données afin que notre modèle puisse les exploiter.

Pour commencer, nous avons trois soucis :

- Nous n'avons pas de paires d'équivalence pour les images de test
- Les images qui sont à notre disposition sont beaucoup trop grande
- Nous n'avons pas beaucoup d'image dans notre dataset

Pour le problème des données de test, j'ai fait le choix de les supprimer car je n'ai pas de solution pour retrouver les images minimalistes correspondante, alors ces images sont inutiles pour notre besoin. Mais avoir faire ce choix ne fait que renforcer un autre problème qui est le manque de données.

Le problème du nombre des données et de leur dimension c'est réglé assez naturellement. EN effet, en mettant en place le tutoriel de tensorflow, il est demandé d'avoir des images de taille 256*256. En suivant cette condition, j'ai mis en place un algorithme permettant cette découpe. Il en résulte alors :

- 36 367 paires pour l'entrainement
- 9065 paires pour les tests

Donc en plus d'avoir un jeu de données qui devient intéressant au moins en quantité, nous avons eu aussi la possibilité de se créer un jeu de données de test ce qui est essentiel pour se conforter dans le bon fonctionnement du modèle. Lors de la création de ces datasets, j'en ai aussi profité pour mettre sur un support comment (un fichier png) les paire image minimaliste / image aérienne correspondante.

L'utilisation de pix2pix engendre un certain nombre d'autre prétraitement comme :

- Redimensionner les images en 286*286
- Recadrer au hasard les image en 256*256
- Faire des rotations aléatoires horizontal
- Normaliser les pixels des images dans un intervalle [-1, 1]

Tous ces prétraitements sont décrits dans l'article de tensorflow.

D- Le modèle

Comme tous les GAN pix2pix est constituer d'un discriminateur et d'un générateur :

- Le générateur est U-net, ce modèle performe dans computer vision et la segmentation d'image, ce qui est intéressant dans notre cas car la première partie concerne la segmentation d'image afin de détecter les zones de bâtiments. Comment est constitué U-net :
    - Un encodeur (downsampler) qui va récupérer les informations de l'image minimaliste
    - Un décodeur (upsampler) qui va générer l'image en prise aérienne

  La fonction de perte du générateur est constituée de deux éléments, une combinaison d'une entropie croisée et de la perte L1 qui sera appliquer entre l'image généré et l'image cible. Cette perte L1 permet d'essayer d'avoir une image qui se rapproche le plus de l'image d'origine. Nous utilisons aussi optimiseur qui est Adam. Le choix de cette structure pour le générateur est fait en se basant sur l'article de pix2pix.
- Le discriminateur est un PatchGan qui analyse des zones de taille 70*70 de l'image d'entrée. Le discriminateur a 2 entrées :
    - La première : l'image d'entrée et l'image cible qu'il doit classifier comme vrai
    - La deuxième : l'image d'entrée et l'image générée qu'il doit classifier comme faux

  La fonction de perte du discriminateur est la somme de l'entropie croisée sigmoïde par rapport à un tableau de 1 et l'image réelle (ici on prend un tableau rempli de 1 car me discriminateur devrait détecter cache patch de l'image comme réel). On ajoute à cette perte l'entropie croisée sigmoïde entre l'image générée et un tableau de zéro. C'est la somme de ces deux composant qui constitue la fonction de perte du discriminateur. L'optimiseur du discriminateur est Adam.

E- Entrainement

Dans le tutoriel, ils n'ont pas mis en place un entrainement sous forme d'époque mais plutôt sur un nombre d'étape, une génération d'image plus le calcul de sa perte compte pour une étape. Ils sont partis sur une base de 40000 étapes. J'avais prévu de mettre en place un entrainement par époque par la suite pour cette partie mais nous verrons dans la partie suivante pourquoi je n'ai pas réalisé cette étape.
Ce qui est important à savoir c'est que pour chaque étape, nous mettons à jours les points di discriminateur et du générateur.

40000 étapes est quelque chose d'assez léger car lors de la rédaction de ce rapport je me suis rendu compte que ça ne correspondait finalement presque qu'a une seule époque au vu de la taille de notre dataset d'entrainement.

F- Les premiers résultats

De cet entrainement, nous avons pu sortir deux graphs de perte, un pour le discriminateur et un autre pour le générateur. Concernant le générateur, on voir que la perte diminue fortement des premières étapes avant de se stabiliser (passant de 83 à 33) mais à un niveau assez élevé ce qui prouve qu'il y a encore moyen d'améliorer la génération. Nous pouvons constater le même phénomène pour la perte du discriminateur passant de 1.71 à 0.41 et semble se stabiliser vers ces valeurs. Vu la différence entre les deux pertes on est même en droit de se demander si le discriminateur n'est pas en train de dominer le générateur dans cette situation. Car même dans le tuto il y a un grand écart entre la perte du discriminateur et du générateur mais elle n'est quand même pas si importante. Je ne peux pas vous montrer de graph pour cette partie car j'étais en train de prendre en main TensorBoard lors de cette étape et les graphs que j'ai obtenu ne sont pas très lisible.

Au niveau visuel on peut voir que la forme des bâtiments est bien respecter et qu'une idée de la forme de la toiture est apparu. Mais il y a un élément qui me dérange c'est que pour les parties en noir sur l'image minimaliste le générateur a du mal à construire quelque chose de cohérent. A ce moment-là, la raison qui m'a semblé naturelle est la forte disproportion dans le

dataset entre les parties tagguées et celle non taggués. Mais avec le recul cela pouvait être aussi dû au fait de l'entrainement qui était bien trop court.

Néanmoins, avec cette hypothèse je me suis ouvert une nouvelle porte pour continuer mes recherches.

G- Modification

Je devais alors trouver une solution pour rendre le dataset plus proportionnel. Lors de mes multiples parcours du dataset je me suis rendu compte que beaucoup d'image minimaliste étaient entièrement noire. J'ai donc ajouté un prétraitement qui supprime ces images.

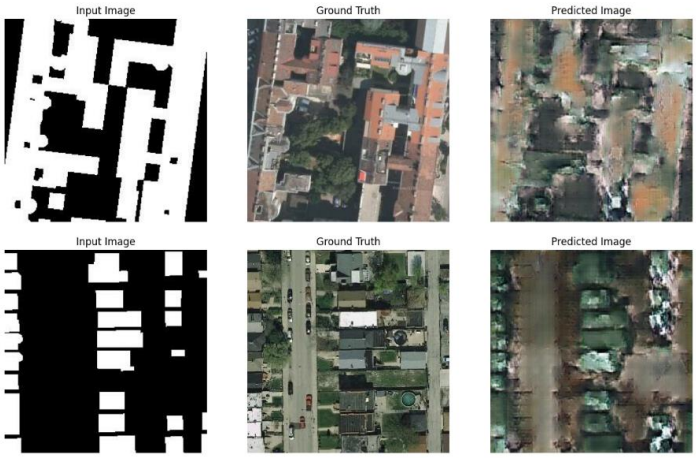
Ce qui a fait diminuer drastiquement la taille du dataset. Pour donner suite à cela j'ai relancé l'entrainement de mon modèle. Diminuer la taille du dataset en gardant le même nombre d'étape a permis de monter de manière artificielle par la même occasion le nombre d'époque ce qui pourrait être bénéfique pour l'entrainement.

H- Résultats et critiques

En analysant l'évolution de la perte à la suite de ce changement (graph non exploitable aussi) je vois que la perte du discriminateur n'a pas trop évolué mais celle-ci était déjà basse donc ce n'est pas dérangeant. En revanche, celle du générateur a continué de diminuer, en passant à 26 ce qui prouve que l'entrainement va dans le bon sens même si celle-ci avait tendance à se stabiliser dans les dernières étapes. Mais cette évolution semble confirmer mon hypothèse du moins dans les chiffres.

En regardant ses deux éléments qui ont été générés :

- La forme des bâtiments est toujours respecté et un semblant de texture pour le toit est toujours présent
- Certaines « textures » générer pour les zones noires sont cohérente comme on peut le constater sur la deuxième image générer par exemple



- En revanche, ce n'est pas cela reste assez aléatoire comme on peut le voir sur la première image

A ce moment-là, pour moi le problème venait du fait que nous manquons d'éléments pour définir les éléments dans la partie en noire. L'hypothèse que j'ai alors posée est d'utiliser une dataset multi-classe afin de résoudre ce problème.

En rédigeant ce rapport, je pense que le problème pouvait aussi venir comme je l'ai écrit dans la partie F- Les premiers résultats d'un entrainement trop succinct mais je ne pourrais pas plus détailler ce point dans ce rapport car je ne l'ai pas traité.

IV- Cas multi-class

Avec mes tentatives précédentes j'ai donc conclu que la meilleure solution était d'avoir un jeu de données multi-classes pour avoir plus d'informations concernant la composition de l'image à générer.

A- Le dataset

Pour ce faire j'ai décidé d'utiliser le dataset proposer dans le sujet du projet qui est landcover. Ce dataset est donc multi-classes et sont réparties de cette façon :

- Les bâtiments
- Les forêts
- L'eau
- Les routes
- Autres éléments

Donc on voit clairement au vu de cette répartition que nous avons beaucoup plus d'éléments de détails que pour le premier dataset.

Ce dataset est contituer de 41 photos : 33 de taille 9000*9500 et 8 de tailles 4200*4700. Donc on se retrouve dans le même souci que lors du premier dataset. De plus les images sont au format GeoTiffs ce qui ne les rends pas exploitable directement par le modèle. Il est bon à savoir que ce dataset n'a pas été construit à la base pour une tache de classification d'image mais plutôt d'auto-mapping comme on peut le voir sur ce papier.

Pour plus d'information sur le dataset les images ont été prise en Pologne.

B- Choix du modèle

Dans un premier temps je suis partie exactement sur le même modèle que pour le premier dataset ce qui permet d'avoir un élément de comparaison pour voir les différences de résultat entre les deux dataset en changeant un minimum de paramètre.

C- Le prétraitement

Avec ce dataset nous avons le même problème qu'avec le premier dataset qui concerne le peu d'images que nous avons à notre disposition. Mais l'avantage de celui-ci c'est qu'un script est intégré pour split les images en des image de taille 512*512 et les répartir dans un dataset d'entraiment, de test et de validation (ce dernier que nous ne disposions pas lors du précédent dataset). Pour un souci de cohérence et pour comparer avec les résultats du premier dataset, j'ai décidé dans un premier temps de rassembler le dataset d'entrainement avec celui d'évaluation ce qui nous fera finalement que deux dataset.

Un autre problème de ce dataset concerne les images minimalistes celle-ci sont en nuance de gris (Geodiff) ce qui les rends difficilement lisible par l'Homme. La solution a été de les rendre plus lisible en ajoutant du « contraste » grâce à un biblothèque Python. Comme on peut le voir sur l'image la carte est maintenant lisible facilement.

Après avoir fait ces étapes, j'ai dû mettre les images minimalistes et les images aérienne sur un même support afin de les lier par paire et aussi de les convertir au format PNG pour que celles-ci soient exploitable par le modèle.

Les autres prétraitements sont les même que j'ai dû faire pour le dataset mono-classe à l'exeception que les images d'entrée ne sont pas de taille 256*256 mais de taille 512*512.

D- Les résultats

Comme je l'ai dit précédemment dans un premier temps j'ai fait le choix de ne pas toucher au fonctionnement du modèle afin de pouvoir effectuer une comparaison avec le premier dataset. Donc ni la construction du modèle ni l'entrainement du modèle n'a changé comparer au premier dataset.

Quand nous regardons les résultats obtenus pour les différentes pertes, ils sont bien différents de ceux obtenu précédemment. En effet, concernant la valeur de la perte du discriminateur, celle-ci est légèrement plus importante que celle obtenu après l'entrainement du dataset mono-classe, mais cela s'explique par le fait qu'il y a beaucoup plus d'éléments que le générateur doit prendre en compte, donc un surement plus compliqué pour lui de générer des images qui semblent réelles en si peu d'étapes. En revanche, la perte du générateur est bien plus basse aux alentours de 22 ce qui montre une large progression. D'ailleurs dans ce cas-là on ne peut pas affirmer qu'un modèle domine l'autre, car aucun des deux modèles à une erreur extrêmement basse.

Au niveau des résultats visuel, on obtient des images un peu plus floues que ceux générer par le data set mono-classe comme on peut le voir sur les différents exemples. En revanche, on constate aussi aisément que les images sont beaucoup plus détaillées et cohérente. Et que même certains paterne non tagguées sur le dataset sont ressortie comme un semblant de sable qui fait la jointure entre la terre et l'eau.

On peut conclure que ce dataset nous permettra d'obtenir des résultats bien plus cohérents et donc on va pouvoir continuer à approfondir son utilisation.

E- Epoques

Au vu du potentiel qui a été détecté lors du dernier test, j'ai enfin décidé de mettre en place un système d'époque (10) afin de pousser un peu plus l'entrainement de cette solution.

Concernant les différentes fonctions de perte, elles ont subi toutes les deux une légère diminution ce qui semble logique au de la durée de l'entraînement qui augmente. Mais je me serais attendu à une plus grosse diminution ce qui signifie peut-être que le modèle a atteint ses limites avec la configuration actuelle. En revanche, on peut clairement constater que la meilleure époque n'est pas la dixième mais un précédente ce qui peut sembler logique étant donné que la perte varie d'une époque à l'autre et ne va pas automatiquement en diminuant.

Pour la partie visuelle, on constate une nette augmentation de la qualité des images. Comme on peut le voir sur l'image ci-dessous, les bâtiments sont plus structurés, de plus le modèle à compris qu'autour des maisons ce n'est pas directement de l'herbe par exemple et il essaye de texturer le toit par exemple ce qui n'était pas forcément le cas pour nos images de la partie précédente.

Pour donner suite à ses résultats, la décision qui a été prise est de modifier la composition du modèle et de l'entrainement afin d'essayer d'améliorer les résultats.

F- Evolution du modèle

Dans la recherche de toujours améliorer le modèle, à cette étape 3 nouveaux éléments vont être ajouté :

- Du early stop avec un patience de 10 sur 30 époques pour éviter tout sur entrainement et sélectionner la meilleure configuration possible pour générer les images. Donc si la perte globale du générateur ne diminue pas sur 10 époques sur les données d'évaluation, le modèle s'arrête et restore les paramètre qui ont obtenu la loss la plus basse. Ce qui a donc induit de diminuer la taille du jeu d'entrainement pour utiliser le jeu de validation initialement fourni par les créateurs du dataset, ce qui a eu pour conséquence de diminuer le jeu d'entrainement de 1600 paires d'images.
- Des couches résiduelles, elles seront ajoutées dans le générateur et l'encodeur. Elles répondent à différents objectifs :

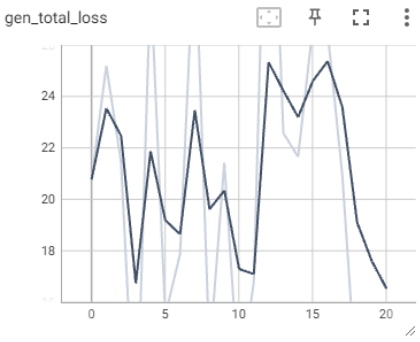
    - Pour le générateur :
        - Dans l'encodeur : Capturer les caractéristique complexe pour extraire des informations qui sont plus fines.
        - Dans le décodeur : Générer des structures plus détaillées
    - Dans le discriminateur l'objectif est de pouvoir examiner des structures plus complexes
- Un mécanisme d'attention qui est ajouté dans les couches résiduelle. Le mécanisme d'attention mis en place est le « Squeeze and Excitation », le principe est le suivant :
    - Squeeze : On effectue une opération de global average polling sur les données d'entrée, ce qui réduit les caractéristiques
    - Excitation : On passe le résultat de l'étape précédente dans des couches complétements connecté avec des fonction d'activations RELU et la dernière doit être une fonction d'activation sigmoïde qui servira à pondérer les données d'entrée.

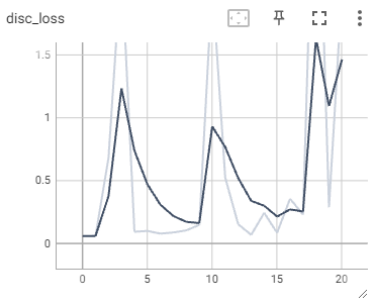
L'objectif avec cette démarche est de savoir qu'elle est la caractéristique la plus importante à cette étape pour générer ou détecter une image factice et ainsi avoir moins d'informations superflu.

Comme on peut le voir sur les images d'exemple, les images sont d'une qualité correcte même si pas en Haute Définition, mais ce n'est pas l'objectif du modèle. En revanche, nous pouvons voir quelques différences par rapport au images générée auparavant :

- Des zones d'ombres cohérentes sont représentées sur les images
- Les toits des bâtiments peuvent être d'une couleur différente, par exemple les petits bâtiments ont un toit gris alors que les grand ont un toit marron
- Il y a des tentatives pour raccorder les bâtiments aux routes comme on peut le voir pour le grand bâtiment représenté sur la première image.

Pour cette étape, une correction a été appliquée pour les anomalies causant un problème d'affichage sur les graphes de perte. Donc comme on peut le voir sur le graph « Gen_total_loss », l'entrainement c'est arrêté de manière prématurée et donc a restauré les meilleurs paramètres qu'il a peut trouver pour le modèle. Ce qui dans les faits nous donne alors une perte de 16.8 pour le générateur et de 0.52 pour le discriminateur. Donc on peut voir visuellement que les paramètres qui ont été restaurés sont ce de la onzième époque. Le problème de cette restauration est que nous sommes presque sur un pique pour la perte du discriminateur (mais ce n'est pas le pique le plus important que nous ayons obtenu). Mais cela n'est pas grave car elle est à peine plus importante que celle qui ont été obtenues jusqu'à présent, à contrario, la perte du générateur elle est beaucoup plus basse que toutes celles calculées jusqu'à présent ce qui prouve donc l'efficacité de cette solution.





À la suite de ces résultats qui commencent à devenir satisfaisant, je me suis lancé dans un changement complet de la fonction de perte du modèle afin d'essayer d'améliorer encore plus les résultats.

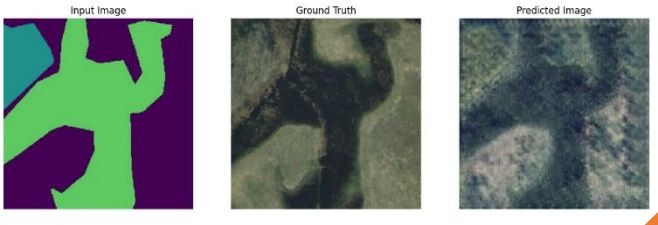
G- WGAN pix2pix

Pendant mes recherches pour améliorer mes résultats, je suis tombé sur un article traitant des Wasserstein GANs que l'on pourrait combiner avec pix2pix. Cette idée me semblant intéressant, j'ai commencé à implémenter ce modèle en modifiant l'existant tout en ne copiant pas 100% l'article afin de faire un mix des deux modèles. Cela a engendré un certain nombre de changement :

- Remplacement dans la fonction de calcul de perte du générateur et du discriminateur de la fonction des d'entropie croisée sigmoïde par la perde de Wasserstein.
- Changement de l'optimiseur en mettant RMSProp avec un learning rate de 0.00005 comme conseillé dans l'article à la place d'Adam.
- Ajout de fonction de clipping lors du calcul du gradient du discriminateur afin de forcer la contrainte de Lipschitz plutôt que la pénalité des gradients comme il est conseillé dans l'article.

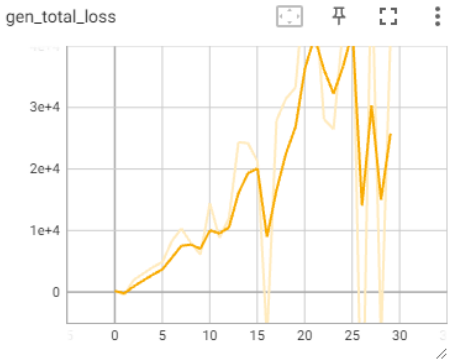
Lors de l'article, il réalise des entraînements sur 100 époques. Ce qui n'est pas réaliste dans notre cas. En effet, depuis que nous sommes passés à en entraînement composé de 30 époques, la durée de celui-ci a explosé passant à environ 12h. Donc, il n'est pas concevable de tripler le nombre d'époque. Mais cette information apportée par l'article nous indique que notre modèle pourrait être encore plus performant en augmentant encore le nombre d'époque ce qui est assez encourageant.

Maintenant, passons à la critique des résultats visuelle obtenu. Dans un premier temps, on constate facilement que les résultats sont visuellement beaucoup moins satisfaisants que ceux obtenues dans la partie précédente. Un artéfact est apparu sur les contours des images, elles sont beaucoup moins nettes aussi. Néanmoins, on peut

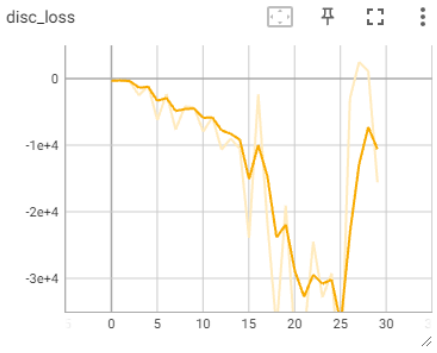


constater que les concepts apportés par les couches résiduelles d'attentions sont toujours présents mais si beaucoup moins précis. Je trouve un aspect positif à ces image, l'impression que les paternes se répètent moins, par exemple sur la deuxième image on voit que dans la zone violette sur l'image initiale, le modèle a généré plusieurs types d'environnement ou qu'il a essayé de texturer la zone d'eau ce qui potentiellement peut rendre l'image plus réaliste. Néanmoins, dans l'état actuelle de mon modèle les images générées sont pas les meilleurs que j'ai pu obtenir. D'ailleurs on constate facilement une grande différence de qualité entre la première image de cette partie et la première image de la partie précédente alors qu'elles sont basées sur la même condition au départ.

Concernant les résultats de la perte pour les deux éléments de ce modèle, la première chose à savoir est que celle-ci ne sont pas du tout comparable avec celle obtenu précédemment car on utilise une fonction de perte différente. En tout cas, on peut constater que l'entrainement ne s'est pas arrêté à cause du early stop ce qui peut signifier que celui-ci aurait pu continuer à s'améliorer. Ensuite, on peut voir que la fonction de perte du générateur grandi fortement,



cela est symptomatique des WGan. En effet, le générateur essaye de s'approcher le plus possible de la vraie image donc il va faire beaucoup de mouvement dans le sens de l'image réel. Donc plus la perte du générateur est haute meilleur elle est. Ici, on voit que la perte du discriminateur est très petite (négative), cela s'explique par le fait qu'il essaye d'éloigner le plus possible les vraies données des fausses. Donc plus elle est négative mieux c'est. Les deux pertes sont dans un ordre de grandeur similaire, donc on peut dire qu'aucun des deux ne domine l'autre.



Maintenant, nous allons mettre en place une solution pour pouvoir comparer les résultats de notre WGan pix2pix avec le dernier pix2pix obtenu de manière numérique. Même si pour évaluer un meilleur modèle par rapport à un autre, un comparatif visuel est important.

H- Comparatif

Donc voici les trois indices numériques (toute basée sur la différence entre les images générées et images réelles) que j'ai mise en place en me basant en grande partie sur l'article WGan pix2pix :

- Une similarité cosinus entre les features map obtenue avec VGG19 : l'intuition qui est présenté dans l'article est que deux images relativement proches doivent produire des features map relativement proche.
- La distance L1
- La distance L2

	Pix2pix	WGan Pix2PIx
VGG19	0.99	0.99
L1	37844.35	45235.23
L2	104.62	125.17

Ces valeurs sont calculées en réalisant une moyenne sur toutes les données de test :

- Pour VGG19 : nous obtenons un très forte similarité cosinus dans les deux cas (Je pense qu'il y a une erreur mais je n'arrive pas à obtenir d'autre résultats), ce qui veut dire que le contenu représenté est fortement similaire entre vrai et généré ce qui est bien car c'est ce que nous cherchons à obtenir.
- Les distance L1 et L2 sont assez importante. Mais on voit que pour les deux solutions la distances L2 est plus faible. Cela peut se traduire que l'on tend à avoir des images réelles et générées similaire. Cependant, il y a une grande distance L1 ce qui peut dire que la couleur et la luminosité des pixels différents grandement d'une image à l'autre.
- Comme nous en avons l'impression lors de l'analyse visuelle, on peut conclure que le modèle pix2pix génère de meilleures images que WGan pix2pix.

D'autre solutions auraient pu être mise en place pour avoir de nouvelles métriques, il aurait été possible par exemple d'utiliser une Fréchet Inception Distance (FID) pour avoir une métrique supplémentaire qui plus est état de l'art.

V- Conclusion

Pour commencer, le premier constat à faire c'est que nous obtenons des images beaucoup plus convaincantes visuellement quand la génération se fait sur un jeu de données multi-classes. Cela semble logique car le générateur dispose de beaucoup plus d'élément pour générer des images.

Comme tous les modèles le nombres d'époques d'entrainement joue un rôle prédominant dans la qualité des images générées. Mais la durée d'une époque peut-être extrêmement longue, pour notre cas environs 40min par époque (GPU).

En générale les images sont quand même assez floutées ce qui rend les résultats peut convainquant. Une solution pourrait être de les passer dans un autre modèle permettant de rendre les images en HD afin de voir si celle-ci semble vraiment réaliste visuellement.

Néanmoins, les images générées respectent correctement la condition d'entrée et sont assez proche des images réelles ce qui était néanmoins l'objectif initiale du projet. Donc la génération d'image satellite à partir de carte est validé en suivant ses critères.

Il aurait aussi pu être intéressant de générer des images avec d'autres modèles comme le GauGAN qui est proposé dans le sujet du projet ou encore un CycleGAN au vu de nos dataset qui ne sont pas non plus très important.

Un autre axe de recherche pourrait être de trouver une solution pour générer des images plus grandes. Actuellement, pour avoir une carte importante, il faut associer plusieurs générations les unes avec les autres ce qui n'est pas forcément des plus pratique.

Pour conclure, ce projet m'a permis de comprendre en profondeur comment fonctionne la génération d'image avec pix2pix et comment fonctionne les WGan. Je ferais le nécessaire pour continuer à approfondir mes connaissances que ce soit avec de la littérature, dans le milieu professionnel ou avec le CNAM.