

Compte-Rendu

Deep Learning TP 7-8-9-10

Yining BAO

Hanshuo WANG

Yongtao WEI

TP 7 - Transfer Learning par extraction de features dans un CNN

1. Partie 1 – Architecture VGG16

1. Sachant que les couches fully connected comptent la majorité des paramètres du modèle, estimer grossièrement le nombre de paramètres de VGG16 (en utilisant les tailles données sur la Figure 1).

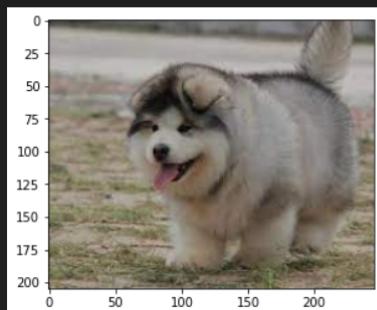
La couche fully connected occupe la plupart des paramètres d'apprentissage. Dans l'exemple de VGG16, il y a trois couches fully connected, leurs tailles sont $1*1*4096$, $1*1*4096$ et $1*1*1000$. Les paramètres à former entre les deux premières couches sont $(4096+1)*4096$, le paramètre à former entre les deux dernières couches est $(4096+1)*1000$. Par conséquent, une estimation approximative est que le paramètre à former est $(4096+1)*4096+(4096+1)*1000$.

2. Quelle est la taille de sortie de la dernière couche de VGG16 ? À quoi correspond-elle ?

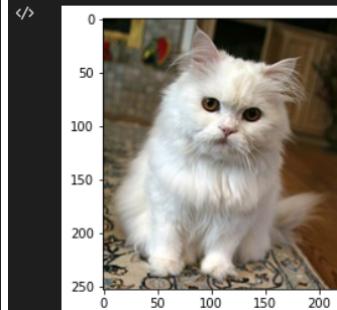
La taille de sortie de la dernière couche de VGG16 est de $1 * 1000$. Après la fonction softmax, chaque nombre représente une probabilité. Ces probabilités correspondent respectivement à 1000 catégories, indiquant la possibilité que l'image d'entrée appartienne à la catégorie.

3. Bonus : Appliquer le réseau sur plusieurs images de votre choix et commenter les résultats de classification.

```
[('malamute, malemute, Alaskan malamute', 84.99201202392578),
 ('Eskimo dog, husky', 4.173534870147705),
 ('Siberian husky', 3.490572690963745),
 ('Great Pyrenees', 3.1113996505737305),
 ('Norwegian elkhound, elkhound', 0.7462154626846313)]
```



```
...  [('Persian cat', 91.72280883789062),
 ('Angora, Angora rabbit', 5.417550563812256),
 ('Pekinese, Pekingese, Peke', 0.6849016547203064),
 ('lynx, catamount', 0.48189815878868103),
 ('Japanese spaniel', 0.2164827436208725)]
```



On trouve le chien Alaskan, le résultat on le probabilité de 84,99%, c'est parfaitement correspond au image.

On trouve le chat Persan, le résultat on le probabilité de 91,72%, c'est parfaitement correspond au image.

2. Partie 2 – Transfer Learning avec VGG16 sur 15 Scene

2.1. Principe de la démarche

5. Pourquoi ne pas directement apprendre VGG16 sur 15 Scene ?

Théoriquement, nous pouvons entraîner le modèle vgg16 à travers un grand nombre d'ensembles de données de scène, en utilisant des paramètres pour initialiser de manière aléatoire. Cependant, cette méthode d'entraînement direct est considérée comme très inefficace, non seulement parce qu'il est difficile d'obtenir un ensemble d'entraînement aussi suffisant, mais aussi parce que nous pouvons utiliser les paramètres de pré-entraînement de VGG16.

Qu'il s'agisse de classification d'objets ou de classification de scènes, les réseaux de neurones doivent extraire des caractéristiques d'images, ce qui est exactement le travail effectué par les couches précédentes de VGG16. Par conséquent, les informations sur les caractéristiques de l'image sont obtenues et nous pouvons modifier le réseau de classification pour obtenir un apprentissage de la migration qui répond à nos attentes.

6. En quoi le pré-apprentissage sur ImageNet peut aider à la classification de 15 Scene ?

Les couches convolutives de VGG16 entraînées sur le jeu de données ImageNet sont capables d'extraire les caractéristiques des objets, et la reconnaissance des scènes consiste pour la machine à reconnaître des combinaisons d'objets. Nous pouvons donc utiliser directement les paramètres de ces couches convolutives entraînées et modifier la structure des couches entièrement connectées afin d'obtenir des résultats satisfaisants avec un jeu de données des scènes plus petit et en moins de temps.

7. Quelles sont les limites de cette approche par feature extraction ?

L'apprentissage par transfert de l'extraction de caractéristiques est limité par la similitude des tâches. Si la différence entre la nouvelle tâche de classification et la tâche d'origine est trop importante, alors cette extraction de caractéristiques aura du mal à jouer un rôle, et même provoquera des erreurs d'apprentissage. C'est le problème du transfert négatif.

2.2. Extraction des features de VGG16

8. Quelle est l'influence de la couche à laquelle les features sont extraites ?

Les caractéristiques sont extraites dans la couche relu7 qui relie la première couche entièrement connectée. Nous pouvons utiliser les caractéristiques comme caractéristiques de l'image et les donner à notre propre classificateur, mais la taille de l'entrée de notre propre classificateur et la taille de la couche (couche relu7) à partir de laquelle nous extrayons les caractéristiques doivent être

les mêmes. Dans le même temps, la couche `relu7` peut modifier la taille de sortie de la couche précédente pour mieux s'adapter au classificateur.

9. Les images de 15 Scene sont en noir et blanc, alors que VGG16 attend des images RGB. Comment contourner ce problème ?

Nous créons une fonction appelée `duplicateChannel`, cette fonction permet de recopier 3 fois une image qui ne serait que sur 1 channel (donc image niveau de gris) pour la "transformer" en image RGB.

Nous utilisons ensuite `transforms.Lambda(duplicateChannel)` dans le prétraitement de l'ensemble de données pour achever la transformation de l'image de l'échelle de gris en RGB.

2.3. Apprentissage de classifieurs SVM

10. Plutôt que d'apprendre un classifieur indépendant, est-il possible de n'utiliser que le réseau de neurones ? Si oui, expliquer comment.

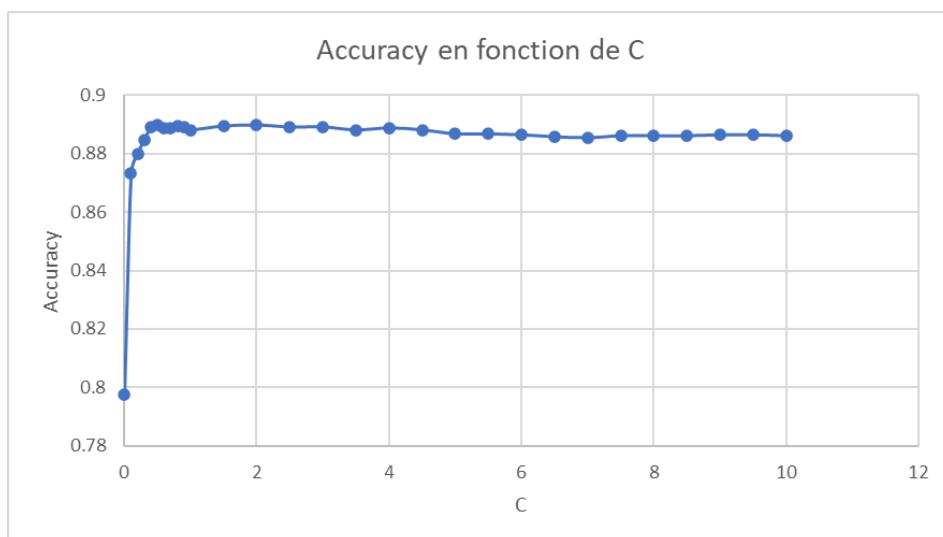
Oui. Le numéro de la dernière couche du réseau de neurones peut être utilisé comme nombre de classifications.

2.4. Aller plus loin

11. Pour chaque amélioration testée, expliquer ses justifications et commenter les résultats obtenus.

Au défaut, il entraîne 1min29s avec `accuracy=0.888107`

1. Si on règle le paramètre `C` de 0 à 10, l'accuracy s'évalue comme le graphique ci-dessous.



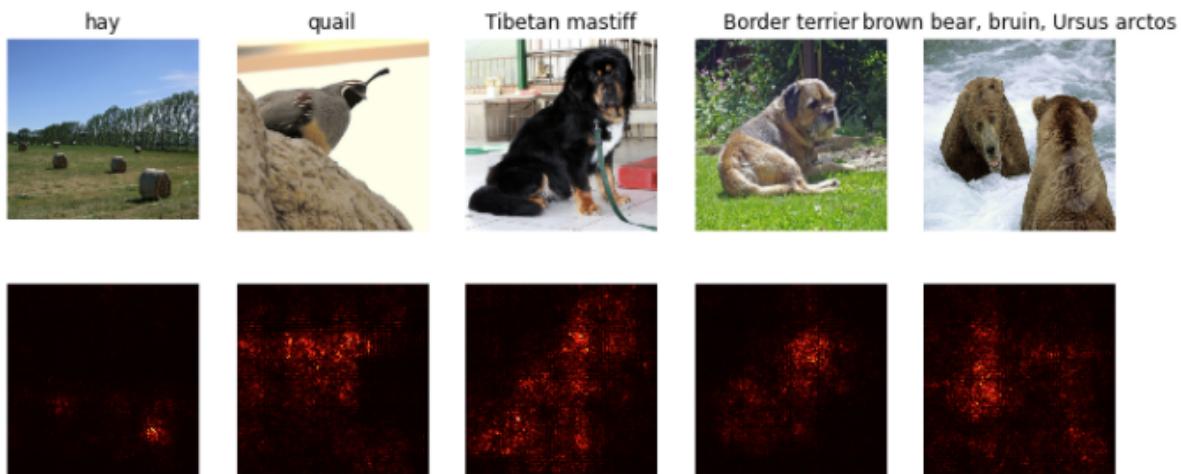
On peut visualiser que pour $C \geq 1$, l'accuracy ne varie presque pas. Cela implique que les données sont plus ou moins séparables. Donc on peut garder $C=1$.

2. Si on change le modèle par AlexNet au lieu de VGG16 et garder les autres paramètres inchangés. On obtient l'accuracy=0.868342 qui est moins bonne que VGG16 avec accuracy=0.888107

TP 8 - Visualisation des réseaux de neurones

1. Partie 1 – Carte de saillance

1. Montrer et interpréter les résultats obtenus.

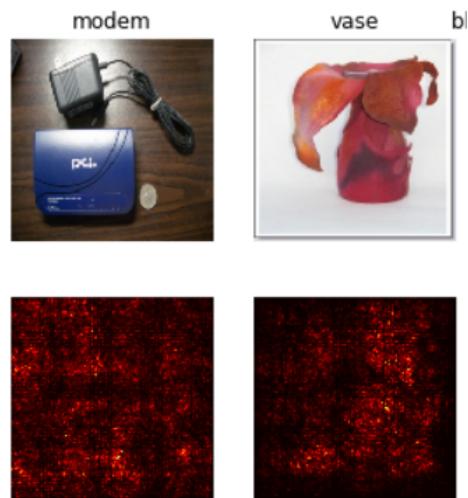


Le résultat de l'opération est montré sur la figure. L'image de la deuxième rangée n'affichera la couleur que là où il y a un objet correspondant. Nous sélectionnons le gradient maximal de chaque pixel pour former une carte de saillance, afin qu'il puisse nous indiquer l'ampleur du score de classification correct lorsque le pixel change légèrement.

2. Discutez les limites de cette technique pour visualiser l'importance des différents pixels.

Du point de vue de la mise en œuvre de cette technologie, comme cette technique prend la valeur absolue du gradient w_i et prend pour chaque pixel la valeur maximale sur les 3 channels RGB pour produire une matrice 2D, seul le gradient d'un des trois canaux RGB a été conservé, ce qui signifie que des informations ont été perdues.

Du point de vue des résultats de sortie, nous constatons que les limites de la carte de saillance se reflètent également dans les divergences de compréhension humaine des "preuves" qu'elle reflète pour la prise de décision. Par exemple, dans l'image ci-dessous, nous ne comprenons pas pourquoi la partie qui n'appartient pas au vase sont toujours considérées comme des pixels importants par la machine.



3. Cette technique pourrait-elle servir à autre chose qu'à interpréter le réseau ?

Il peut être utilisé pour d'autres choses. Cette technologie peut détecter les pixels d'objet appartenant à une certaine classe et représenter approximativement la position du pixel important pour la classification, elle peut donc également être utilisée pour la segmentation d'images, telle que la segmentation sémantique.

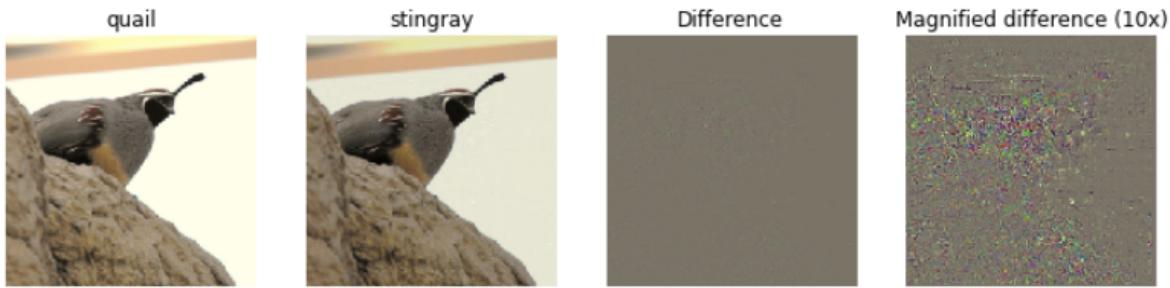
4. Bonus : Tester avec un autre réseau, par exemple VGG16.

Voici l'image générée lorsque VGG16 est utilisé. On peut observer qu'il est plus clair.



2. Partie 2 – Exemples adversaires

5. Montrer et interpréter les résultats obtenus.



Les résultats montrent que lorsque les entrées sont modifiées d'une manière difficile à détecter pour les humains, le réseau neuronal les classe mal, alors que les entrées originales non modifiées sont correctement classées. Si on affiche les différences entre les deux images et on agrandit les différences, on peut voir comment la remontée de gradient fonctionne.

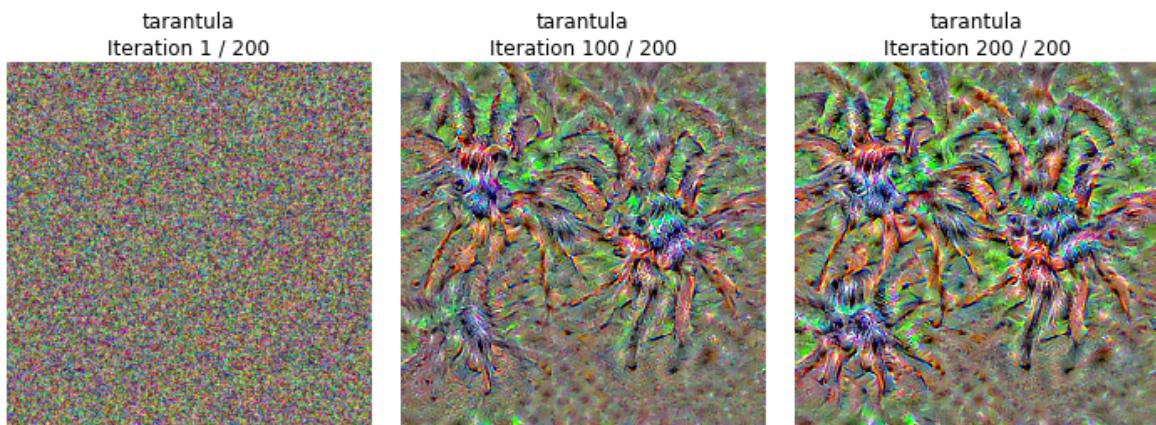
6. Quelles conséquences cela peut-il avoir pour l'utilisation de réseaux de convolution en pratique ?

La présence d'exemples adversaires indique que le modèle a tendance à s'appuyer sur des caractéristiques peu fiables pour maximiser ses performances, ce qui pourrait entraîner des erreurs de classification aux conséquences potentiellement catastrophiques en cas de perturbation.

Par conséquent, nous devons choisir des images avec moins de bruit pour obtenir un résultat de classification correct.

3. Partie 3 – Visualisation de classes

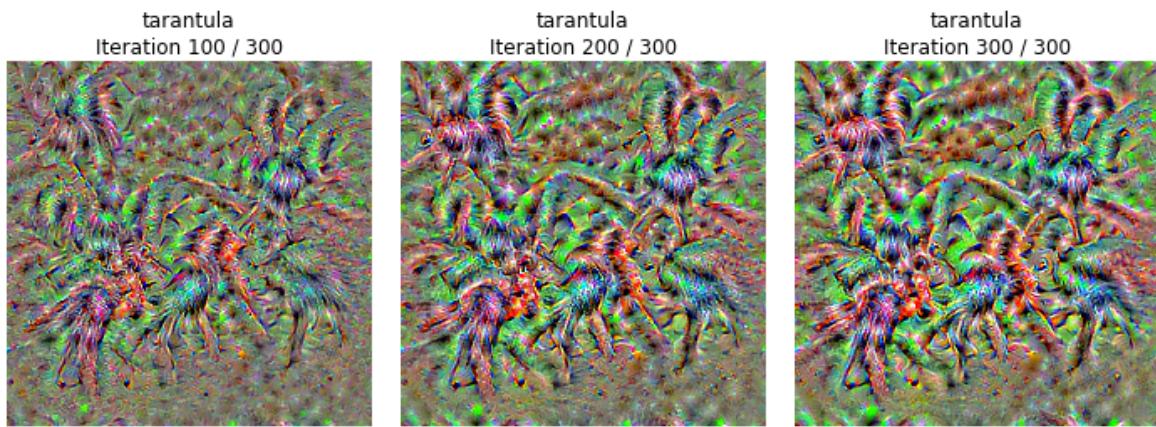
8. Montrer et interpréter les résultats.



Dans l'image ci-dessus, nous pouvons voir que l'image passe progressivement du pur bruit à une tarentule avec learning rate 5 et nombre de itérateur 200.

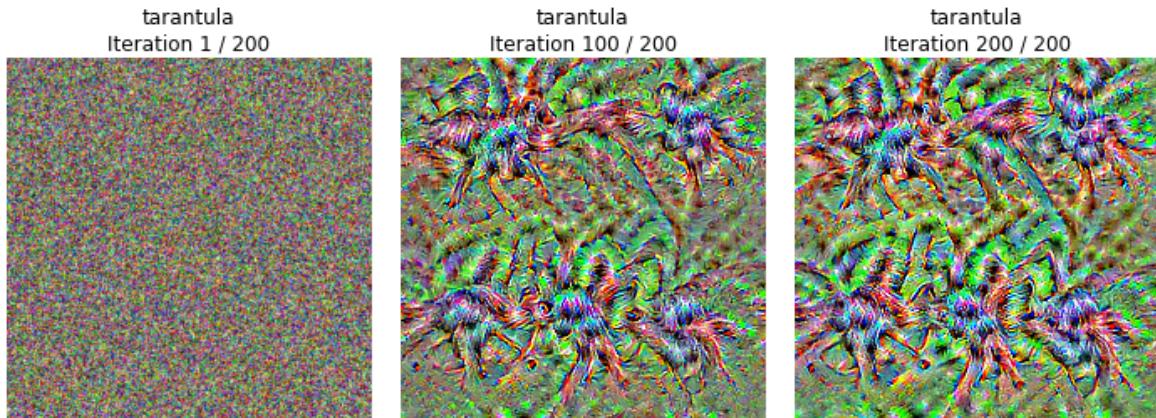
9. Essayer de varier le nombre d'itérations et le learning rate, le poids de la régularisation.

- 1) Augmentez le nombre d'itérations à 300 et les deux autres paramètres restent inchangés.



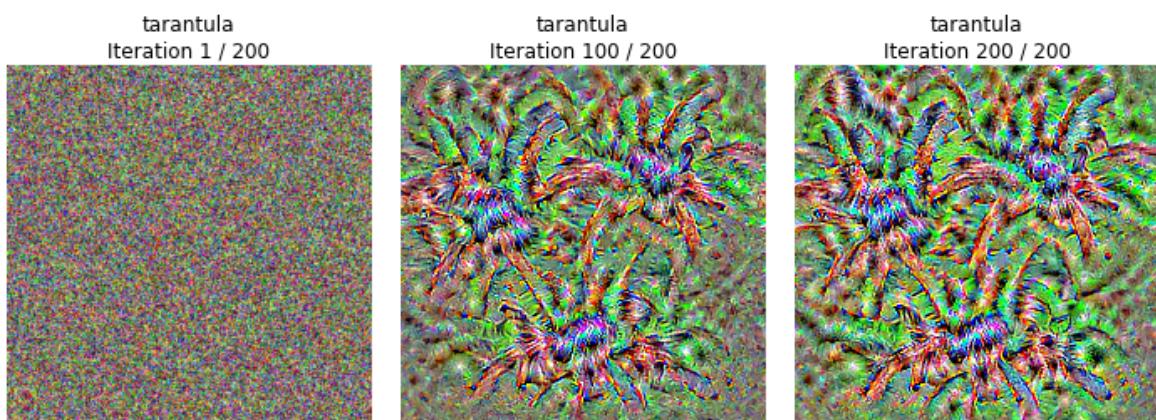
Pour le changement du nombre d'itérations, nous observons que le changement de l'image atteint la convergence lorsque iter = 200. Après iter = 200, le résultat est presque le même.

- 2) Augmentez le taux d'apprentissage à 10 et les deux autres paramètres restent inchangés.



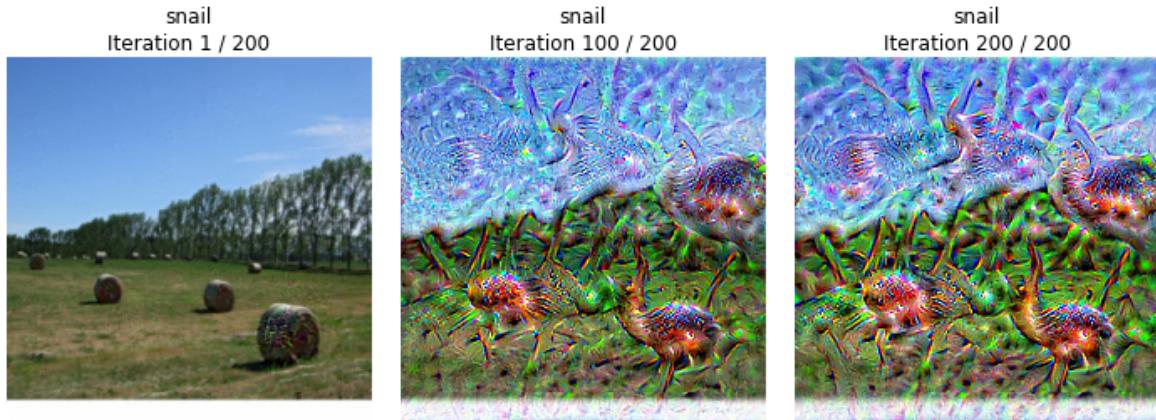
On observe que quand le learning rate augmente, l'apprentissage atteint le point de convergence plus vite.

- 3) Augmentez le poid de regularization à 1e-2 et les deux autres paramètres restent inchangés.



On observe que quand on augmente le poids de la régularisation, le résultat du visualisation de classe est meilleur, qui peut bien distinguer le fond et l'objet.

10. Essayer d'utiliser une image d'ImageNet comme image source au lieu d'une image aléatoire (paramètre `init_img`). Vous pouvez utiliser pour classe cible la classe réelle. Commenter l'intérêt.

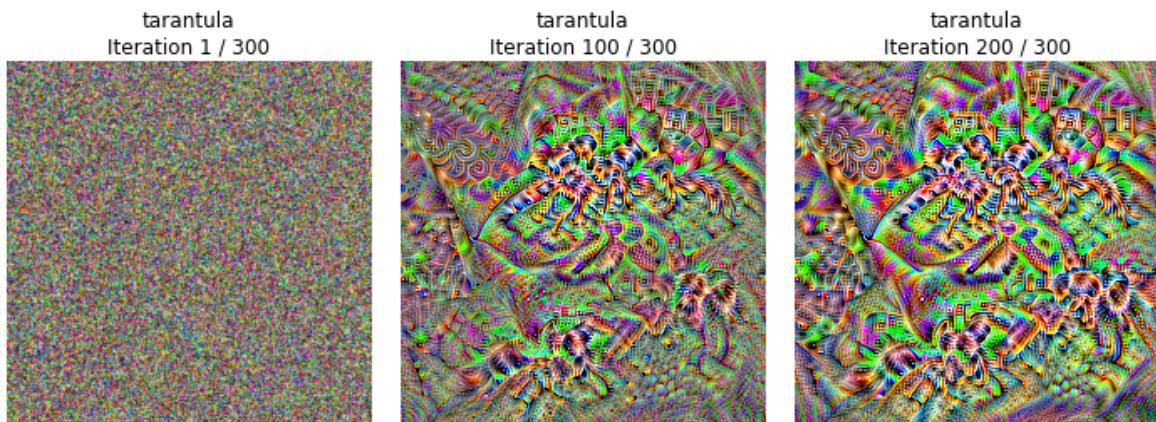


En utilisant une image d'ImageNet plutôt que aléatoire, la vitesse de convergence est plus rapide.

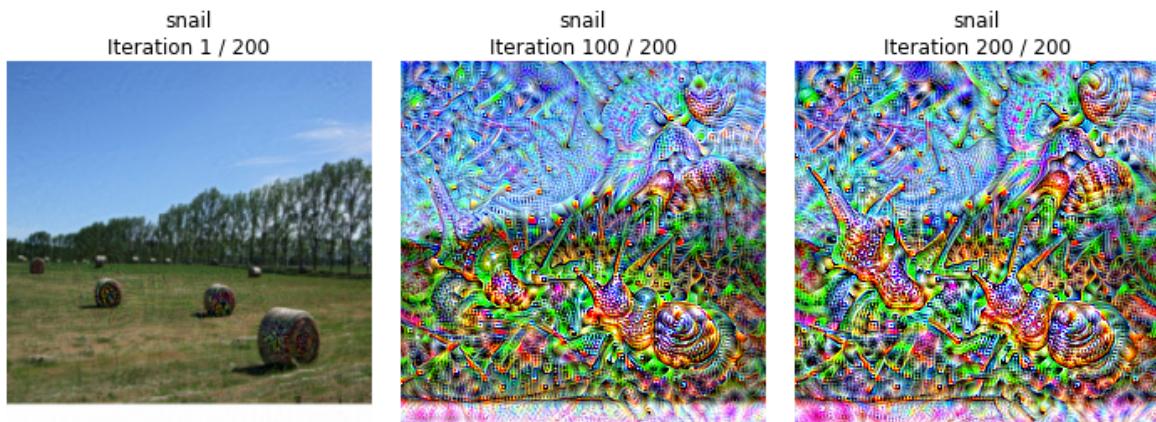
11. Bonus : Essayer avec un autre réseau, par exemple VGG16.

Nous avons observé qu'en utilisant VGG16, l'image observée est plus claire et les caractéristiques sont plus importantes.

Pour les images bruitées :



Pour de vraies images :



TP 9-10 - Generative Adversarial Networks

1. Partie 1 – Generative Adversarial Networks

1.1. Principe général

1. Interprétez les équations (6) et (7). Que se passe-t-il si on utilisait seulement une des deux ?

L'équation (6) est utilisée pour optimiser le générateur G. La signification de la formule est que nous espérons trouver les paramètres du générateur G pour que la formule ait la valeur maximale, qui est proche de zéro. Le but du générateur est de générer une image réaliste afin que le discriminateur puisse la confondre avec l'image réelle. Lorsque la prédiction $D(G(z))$ est proche de 1, c'est-à-dire lorsque le discriminateur croit à tort que l'image générée par le générateur est vraie, $\log(D(G(z)))$ est égal à 0, qui est la valeur maximale.

La formule (7) est utilisée pour optimiser le discriminateur D. On espère trouver les paramètres du discriminateur D pour que la formule ait une valeur maximale. Le premier terme de la formule est que le discriminateur distingue des images réelles. Nous espérons que le discriminateur peut identifier des images réelles, donc $D(x^*)$ vaut 1, $\log(D(x^*))$ vaut 0. Le deuxième terme de la formule est que le discriminateur distingue l'image générée G. Nous espérons que le discriminateur pourra distinguer les fausses images, donc $D(G(z))$ devrait être égal à 0, à ce moment $\log(1-D(G(z)))$ est égal à sa valeur maximale de 0.

Si nous utilisons l'un de ces réseaux pour nous entraîner, nous perdons le sens de la compétition. Ce n'est qu'à travers de multiples itérations de compétition que nous pouvons obtenir G et D avec une grande précision.

2. Idéalement, en quoi le générateur G transforme la distribution P(z) ?

$P(z)$ devient une distribution inconnaisable $P(x \sim)$ après avoir traversé le réseau neuronal complexe du générateur. Afin de rendre l'image générée $x \sim$ plus réaliste, le but du GAN est de faire en sorte que $P(x \sim)$ ressemble de plus en plus à $P(X)$, qui est la distribution d'images réelles.

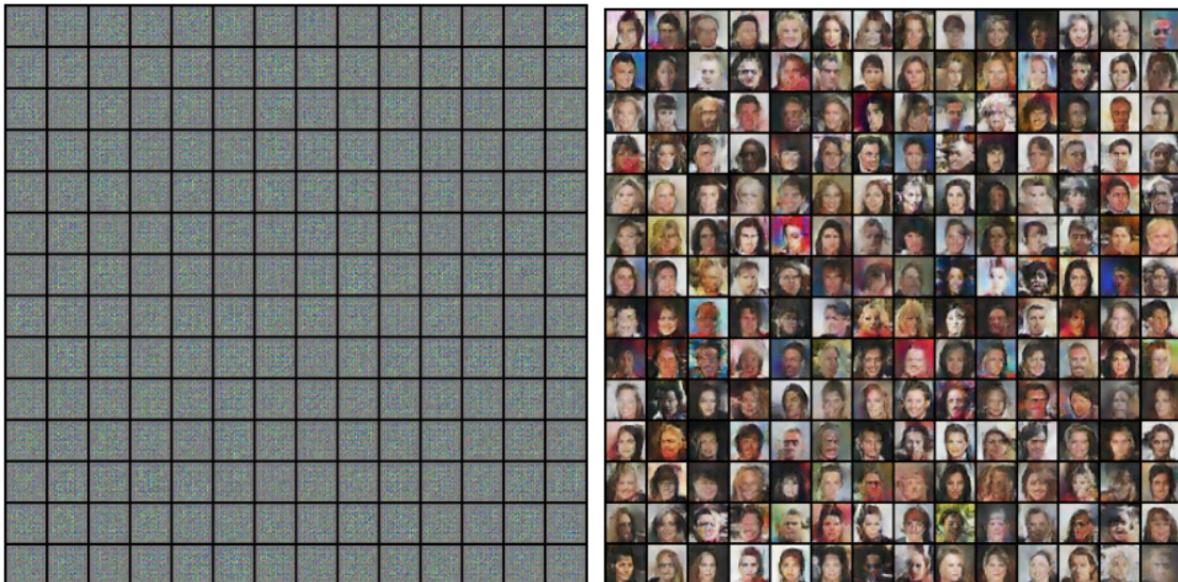
3. Notez que l'équation (6) n'est pas directement dérivée de l'équation 5. Cela est justifié par les auteurs pour éviter la saturation des gradients. Quelle aurait dû être la "vraie" équation ici ?

$$\min_G \mathbb{E}_{z \sim P(z)} [\log(1 - D(G(z)))]$$

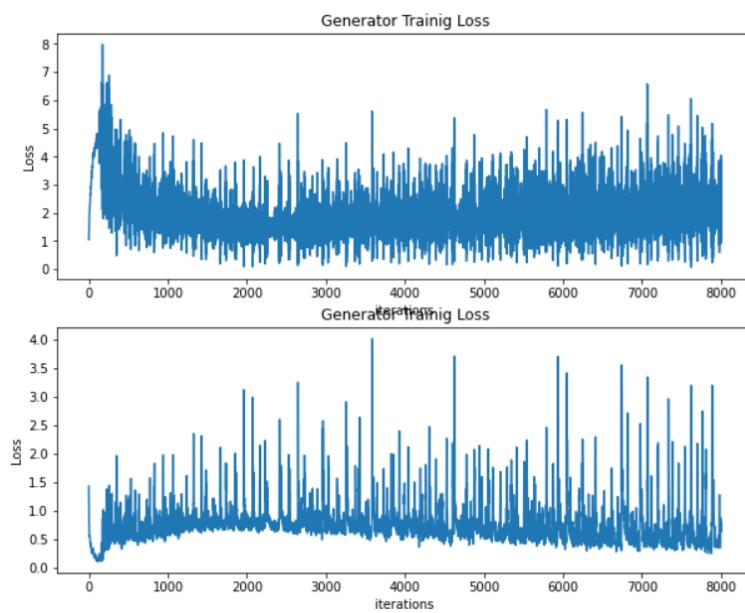
1.2. Architectures des réseaux

4. Commentez l'apprentissage du GAN avec les hyperparamètres proposés par défaut (évolution des générations, de la loss, stabilité, diversité des images, etc.)

Lorsque le nombre d'époques augmente, nous constatons que les images générées passent d'un bruit uniformément distribué à gauche à des visages humains à droite.

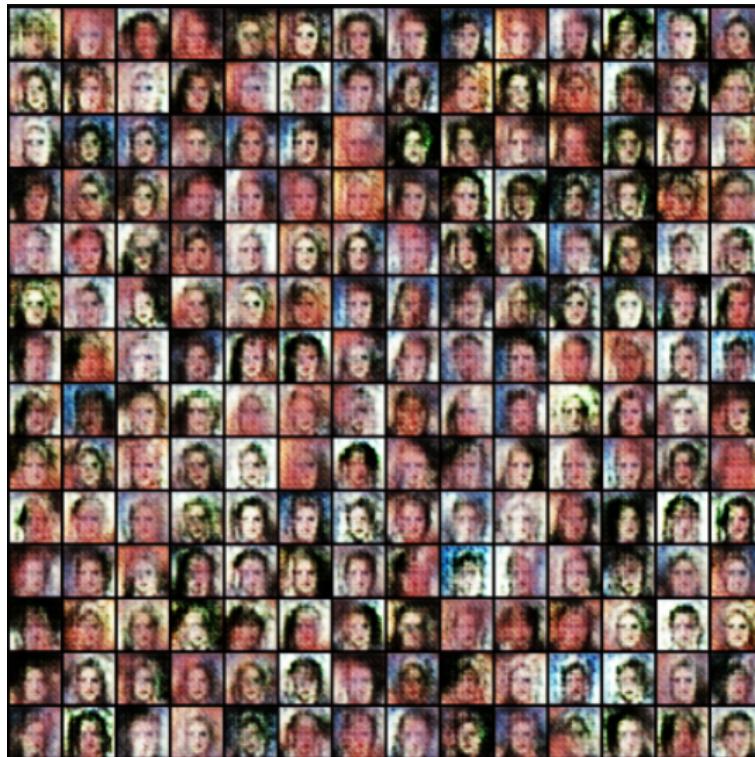


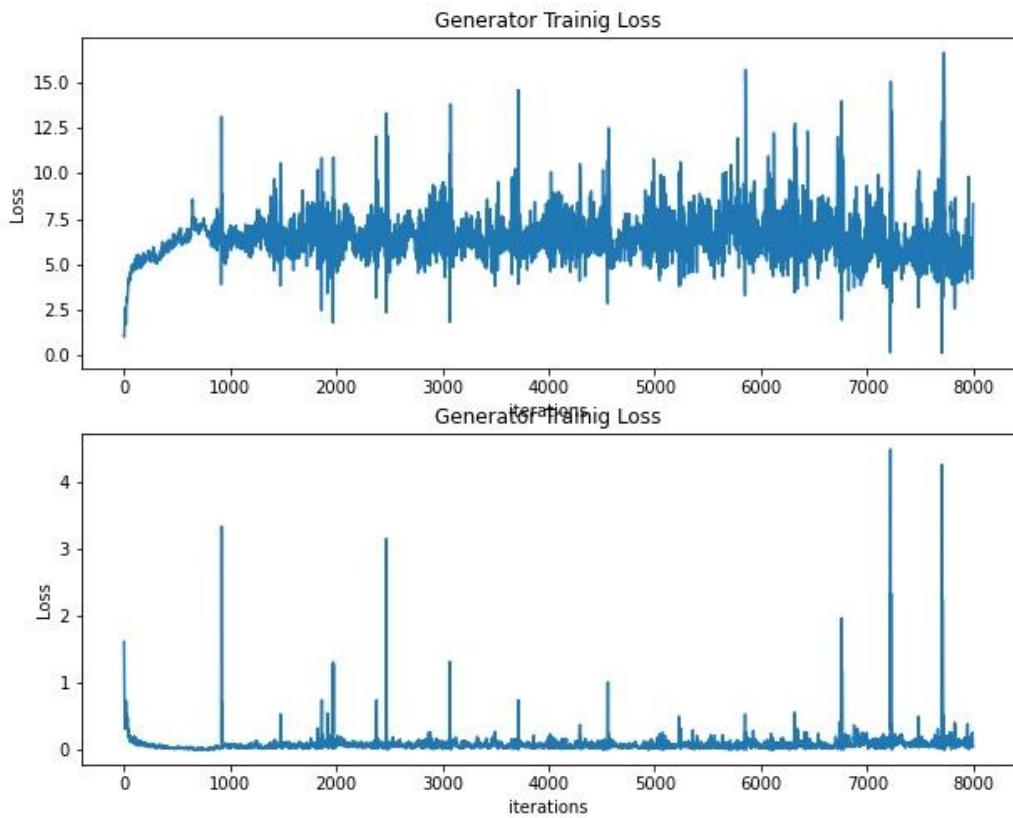
Mais en regardant de près les images, nous constatons que les visages générés ne sont pas très réalistes. De plus, la perte du générateur et du discriminateur continue de fluctuer mais ne converge pas, nous devons donc ajuster les hyper-paramètres par défaut de manière appropriée.



5. Commentez les diverses expériences complémentaires que vous avez réalisées (consignes ci-dessus), comment influe-t-elle sur la qualité de l'apprentissage (vitesse, stabilité, évolution de la loss, qualité visuelle, diversité des générations, etc.)

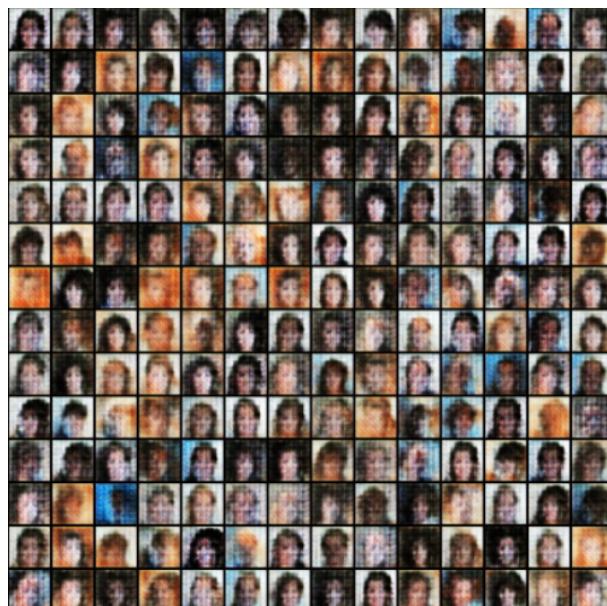
1. On change le momentum pour G et D par 0.9

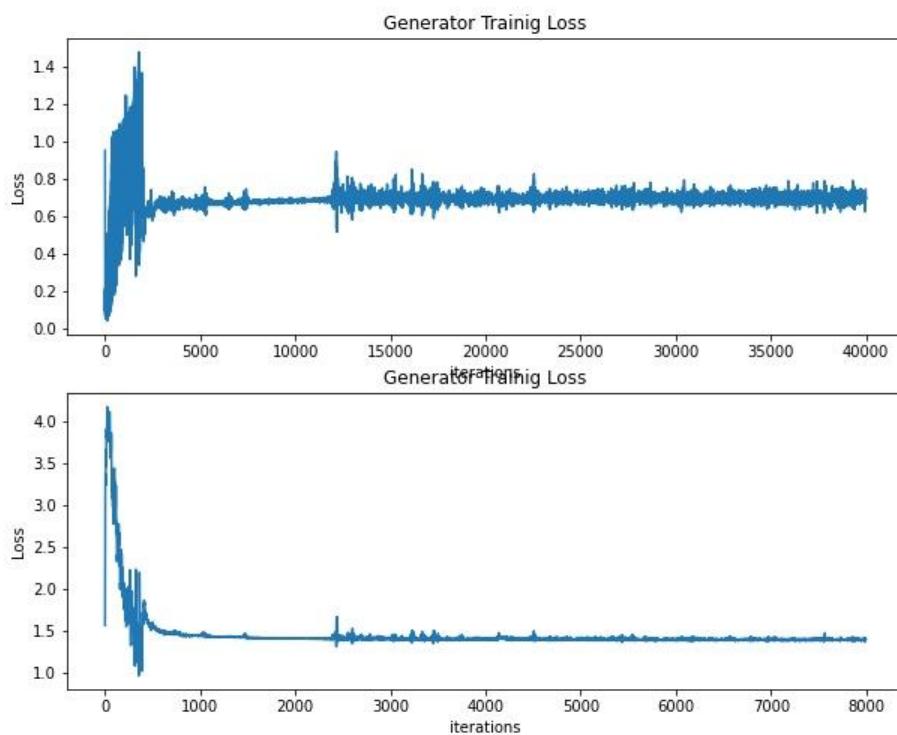




Si on change 0.9 pour le momentum de générateur et discriminateur, alors on obtient le résultat plus mauvais que 0.5 pour le momentum . Loss de discriminateur est souvent presque nul mais la loss du générateur est toujours élevée (la plupart est supérieur à 6). Les images générées par générateur sont mauvaises aussi, peuvent être discriminées facilement par le discriminateur

2. On change nb_update_D=1 et nb_update_G=5

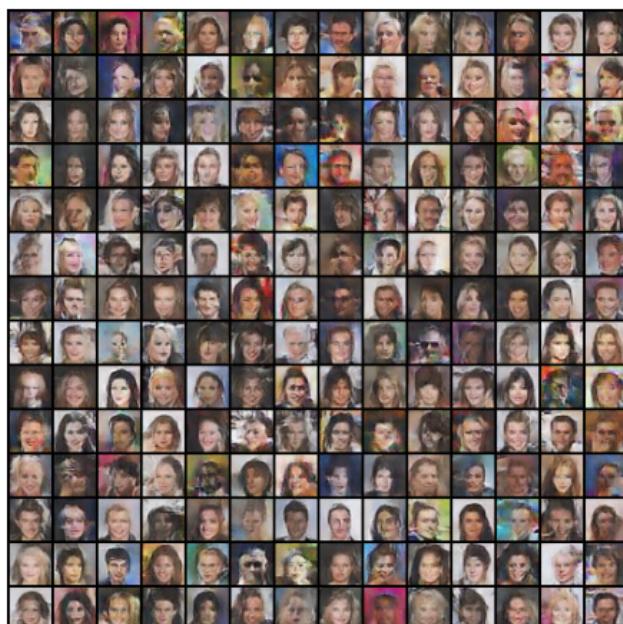


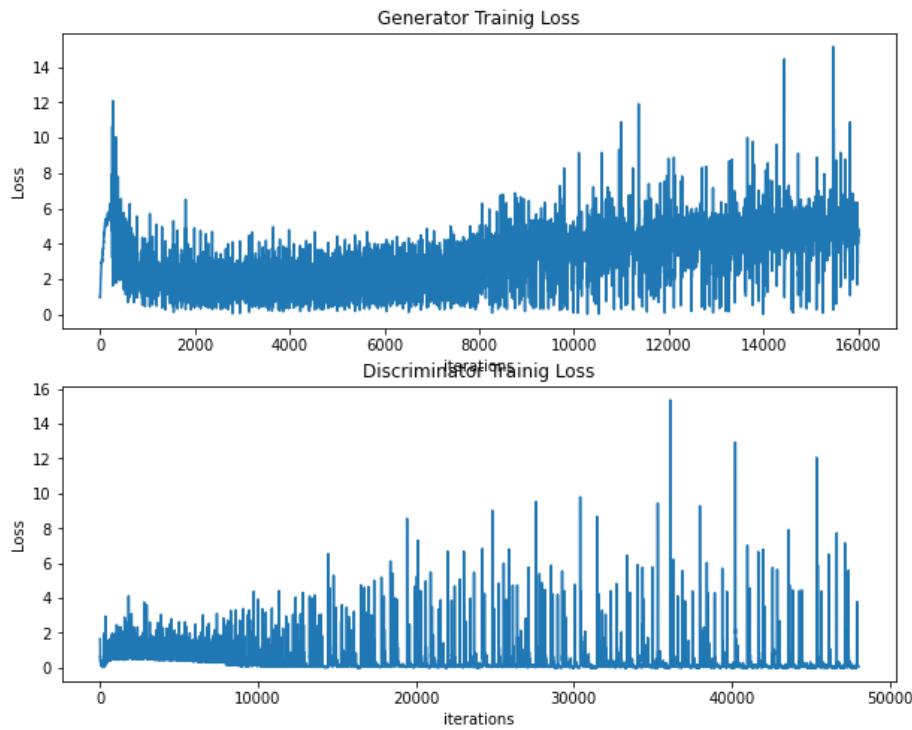


On obtient de mauvaises images aussi . Les deux loss ne sont pas stables au tout début, la loss du générateur est très petite puisque il est mis à jour 5 fois de chaque époche donc il est mauvais de tromper le discriminateur car il est plus avancé que générateur .

Et la loss du discriminateur ne varie presque pas dans les plus grandes itérations donc il apprends mieux à différencier les images avant que le générateur et il ne corrige pas les erreurs.

3. puis nb_update_D =5 et nb_update_G=1





Les images générées sont de bonne qualité, on peut distinguer les visages.

2. Partie 2 - Conditional Generative Adversarial Networks

2.1. Principe général

6. Formellement dans le cas cGAN, quel est le problème qu'on cherche à optimiser ?

(réécrire l'équation (6) et (7) avec un discriminateur et générateur conditionnels)

$$\min_G \max_D (\mathbb{E}_{x,y \sim P_{\text{data}}(x,y)} [\log D(x,y)] + \mathbb{E}_{y \sim p_y, z \sim p_z} [z] [\log (1 - D(G(z,y), y))])$$

7. A quelle(s) variable(s) le générateur de la figure 6 pourrait-il être conditionné, sachant que ce modèle permet de modifier des images existantes ?

Le générateur de figure 6 va conditionner avec les variables de genre(femme ou homme) et l'age(jeune ou vieux).

8. A quelle(s) variable(s) le générateur de la vidéo ci-après pourrait-il être conditionné ?

<https://twitter.com/HumanVsMachine/status/1068909241405251584>

Le générateur de la vidéo peut être conditionné avec les variables de nuages, les arbres et les pelouses etc.

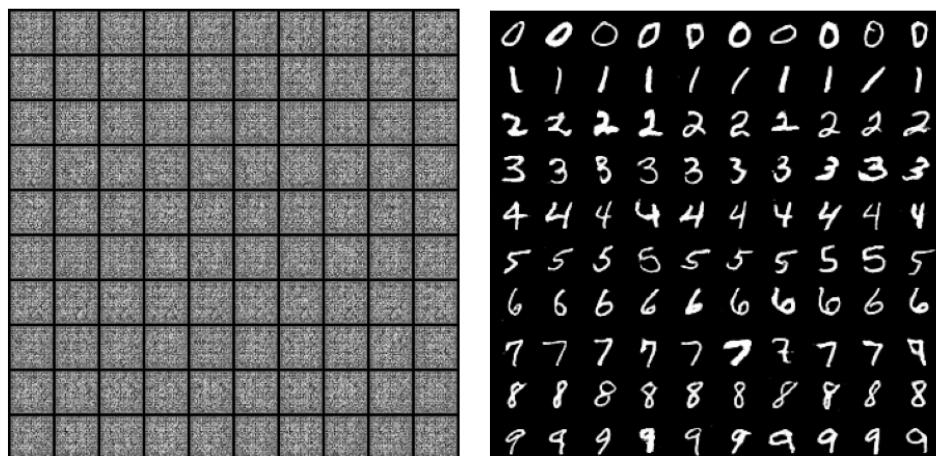
9. A quelle(s) variable(s) le générateur du début de la vidéo ci-après pourrait-il être conditionné ?

Le générateur du début de la vidéo peut être les personnages, les arbres, les transports, les promenades, les panneaux de signalisation etc.

2.2. Architectures cDCGAN pour MNIST

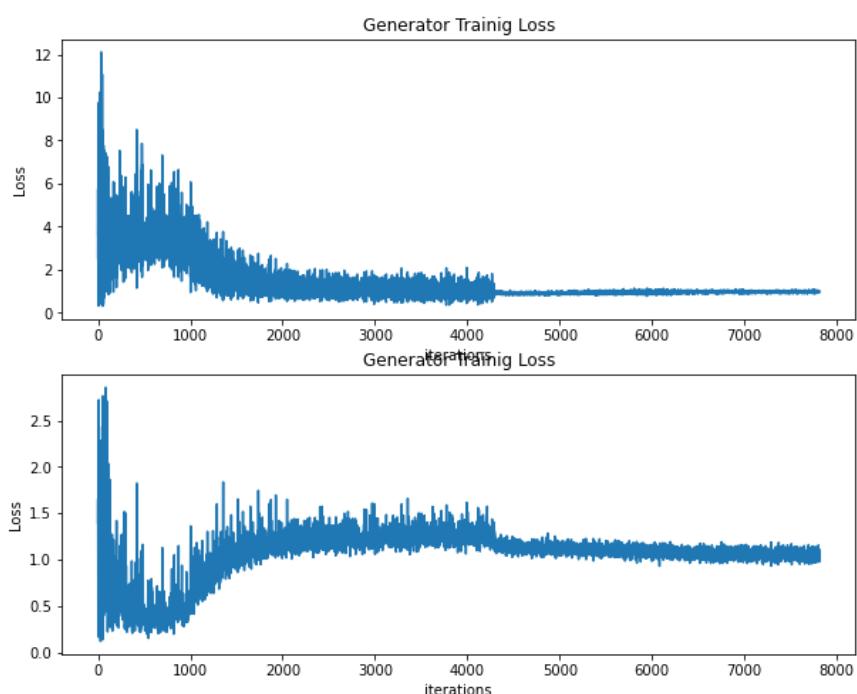
10. Commentez vos expériences avec le DCGAN conditionnel

On observe qu'au fur et à mesure que le nombre d'itérations augmente, les nombres générés par le générateur se rapprochent de plus en plus des nombres manuscrits.



Initial

Fin



11. Pourrait-on enlever le vecteur y des entrées du discriminateur (c'est-à-dire avoir $cD(x)$ et non $cD(x, y)$) ?

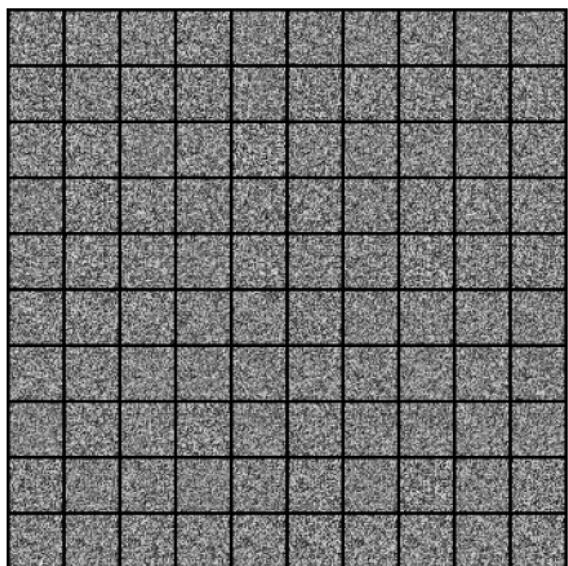
Non, on ne peut pas enlever le vecteur y . En effet, le discriminateur n'a pas seulement à discriminer si l'image générée est réaliste ou non, il doit également déterminer si l'image générée satisfait à la condition y .

2.3. Architectures cGAN pour MNIST

12. Commentez vos expériences avec le GAN conditionnel, reportez la meilleure génération de chiffres conditionnés

Encore une fois, nous pouvons voir dans les résultats que plus le nombre d'itérations augmente, plus les chiffres générés deviennent réalistes.

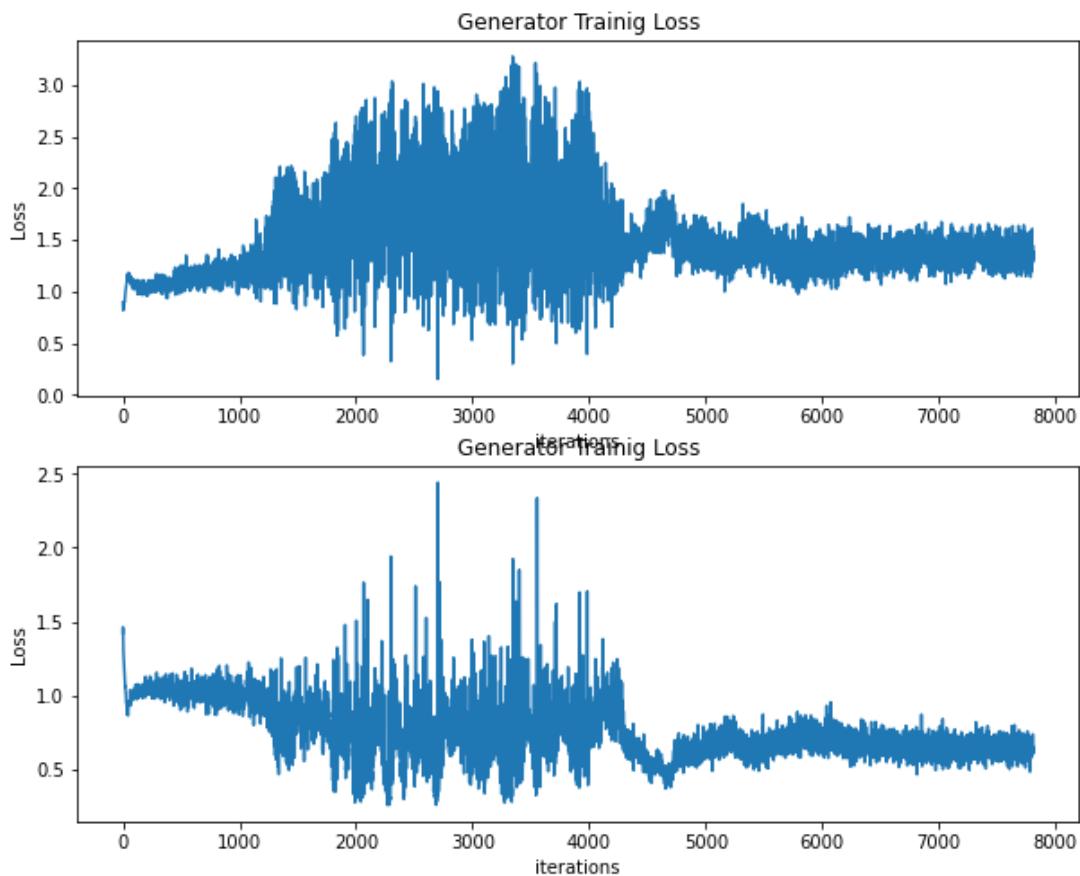
Si nous comparons les résultats de cGAN avec ceux de cDCGAN, nous pouvons voir que les résultats de cGAN ont plus de bruit, mais en même temps ils ont un temps de formation plus court.



Initial



Fin



13. Il est relativement plus difficile de générer des chiffres conditionnés avec un cGAN qu'avec un cDCGAN, pourquoi ?

Par rapport au cGAN, le cDCGAN utilise des couches convolutionnelles profondes dans sa structure, ce qui le rend plus adapté au traitement des images et permet d'obtenir des images plus nettes.