

# **Compte rendu de deep learning**

## **TP 10 - 11**

**Elodie Difonzo**  
**Roqyun Ko**

## Partie 1 – Generative Adversarial Networks

$$(5) \min_G \max_D \mathbb{E}_{x^* \in Data} [\log(D(x^*))] + \mathbb{E}_{z \sim P(z)} [\log(1 - D(G(z)))]$$

$$(6) \max_G \mathbb{E}_{z \sim P(z)} [\log(D(G(z)))]$$

$$(7) \max_D \mathbb{E}_{x^* \in Data} [\log(D(x^*))] + \mathbb{E}_{z \sim P(z)} [\log(1 - D(G(z)))]$$

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

L'équation (6) représente l'espérance de la probabilité pour laquelle l'image générée est "vraie" (l'image générée ressemble à des images réelles).

L'équation (7) représente la somme de :

- l'espérance de la probabilité pour laquelle l'image réelle est vraie
- l'espérance de la probabilité pour laquelle l'image générée est fausse.

Leur grandeur est en échelle logarithmique. C'est-à-dire :

- $\log[D(x^*)] = 0$  car idéalement,  $D(x^*) = 1$
- $\log[1 - D(G(z))]$  = 0 car idéalement,  $1 - D(G(z)) = 1$

Donc, au mieux la fonction D est entraînée, au mieux la fonction D est minimisée.

Lorsqu'on entraîne le générateur (équation 6), les labels des images générées sont réels.

Lorsqu'on entraîne le discriminateur (équation 7), les labels des images générées sont faux.

Si on n'utilisait que l'équation (6):

- Le discriminateur n'est entraîné sur que les images fausses dont les labels sont "vraies".
- $G(z)$  serait mal entraîné car  $D(x)$  ne reconnaîtrait pas la validité des images générées.

Si on n'utilisait que l'équation (7):

- $G(z)$  serait mal entraîné car le réseau  $G(z)$  ne pourrait pas lancer correctement la backpropagation pour s'optimiser pour générer image vraies.

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

La distribution  $P(z)$  s'approchera de la distribution des images réelles.

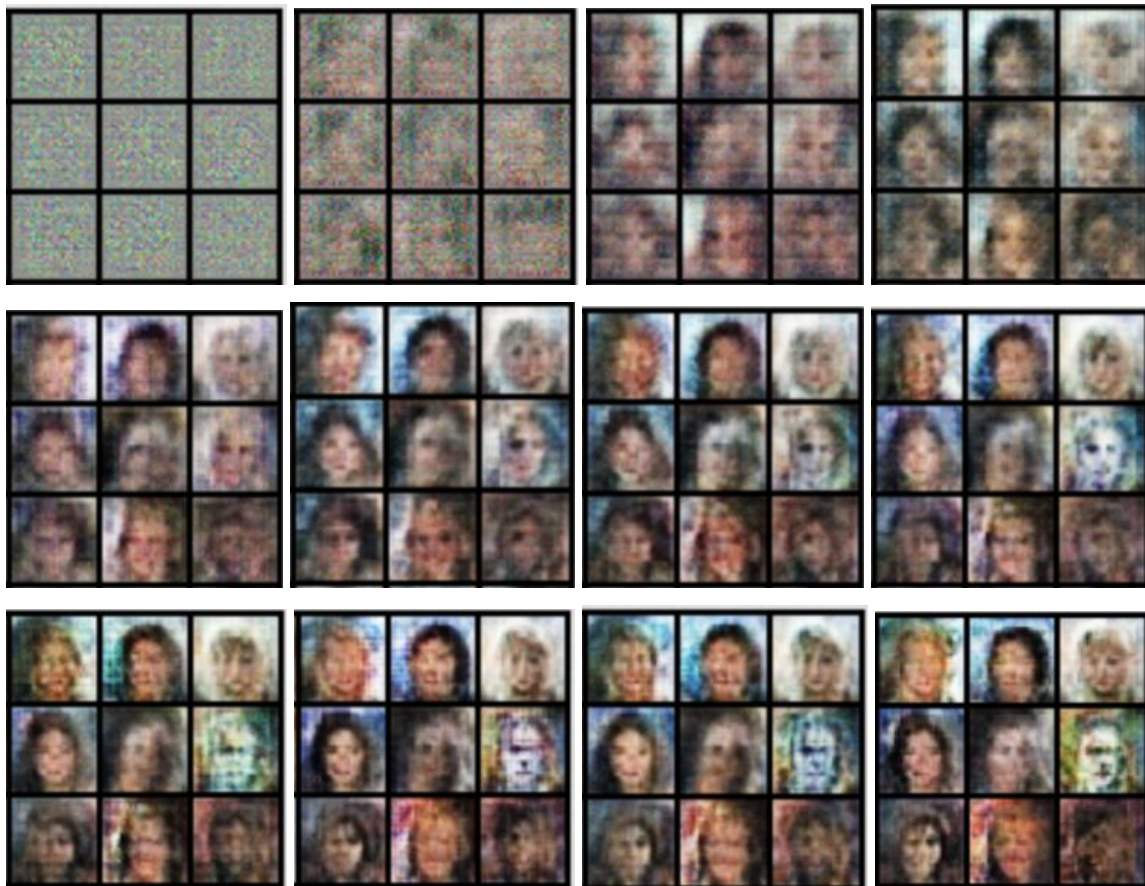
- 3) Notez que l'équation (6) n'est pas directement dérivée de l'équation 5. Cela est justifié par les auteurs pour éviter la saturation des gradients. Quel aurait du être la "vraie" équation ici ?

$$\log[1 - D(G(z))]$$

Cela représente la probabilité que l'image générée soit fausse.

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

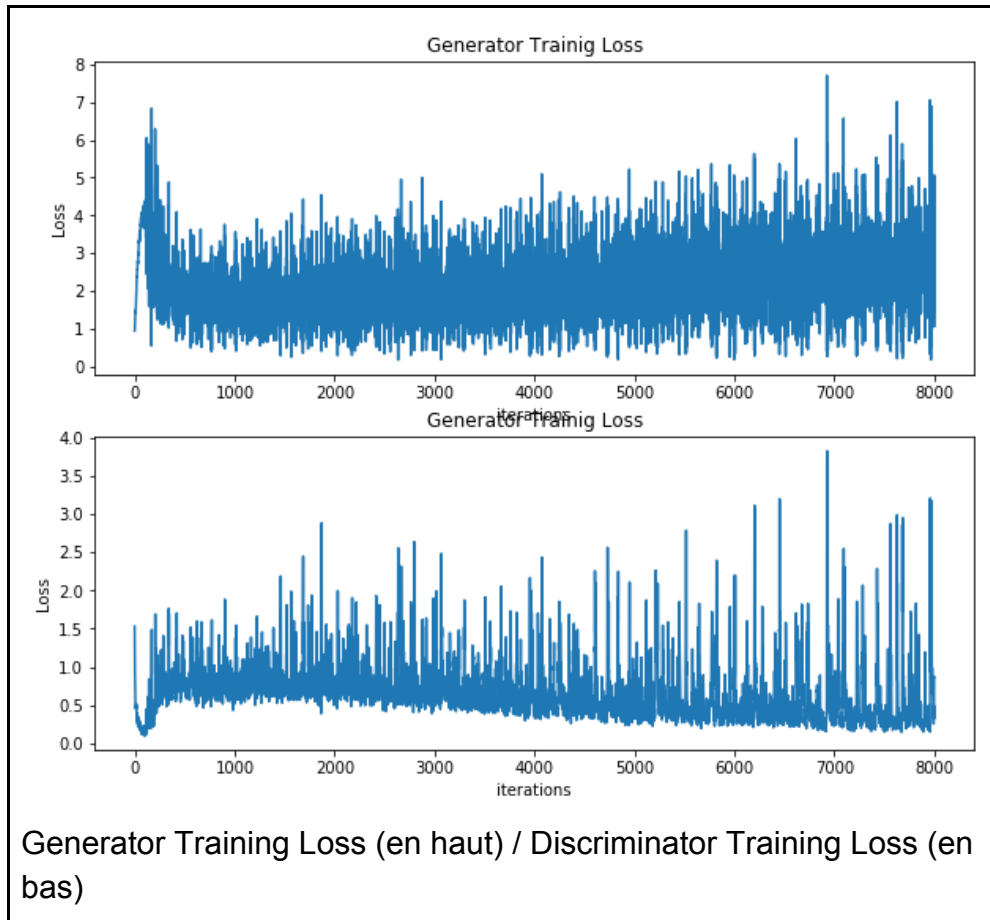
0 → 1 100ème itération :



7 700 → 7 900 ème itération :



Les images originelles (ensemble de données *Celeba*):



La moyenne de “generator training loss” baisse pendant les 1000 premières itérations puis elle monte progressivement.

En revanche, la moyenne de “discriminator training loss” monte pendant les 100 premières itérations puis elle diminue progressivement.

Les deux évolutions de “training loss” sont très instables.

Avec les hyper-paramètres par défaut ci-dessous, on génère les images des visages plus ou moins semblables mais il y a des images “écrasées”

- `nz = 100`
- `lrD = 0.0002` # Learning rate for the discriminator
- `lrG = 0.0002` # Learning rate for the generator
- `beta1G = 0.5` # Momentum beta1 for the discriminator
- `beta1D = 0.5` # Momentum beta1 for the generator
- `batch_size = 128` # Images per batch





- steps = 8000 # Number of global steps in the training loop (5.1 epochs)


5) Commentez les diverses expériences complémentaires que vous avez réalisé (consignes ci-dessus), comment influent-elles sur la qualité de l'apprentissage (vitesse, stabilité, évolution de la loss, qualité visuelles, diversité des générations, etc.)

### Cas 1) l'apprentissage pendant plus longtemps

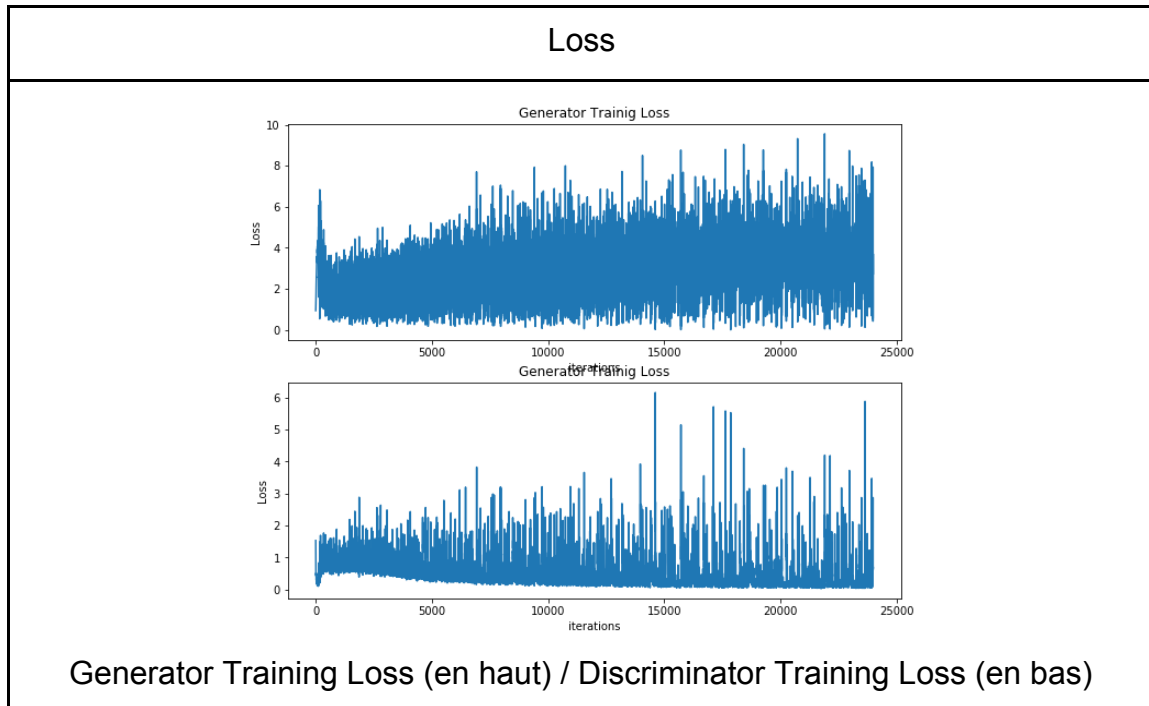
- nz = 100
- lrD = 0.0002 # Learning rate for the discriminator
- lrG = 0.0002 # Learning rate for the generator
- beta1G = 0.5 # Momentum beta1 for the discriminator
- beta1D = 0.5 # Momentum beta1 for the generator
- batch\_size = 128 # Images per batch
- steps = 24000 # Number of global steps in the training loop (15.3 epochs)

13 000 → 24 000 ème itération :

13 000 ème itération	14 000 ème itération	15 900 ème itération	24 000 ème itération
			

7 900 ème itération	24 000 ème itération
	

Par rapport à la 7900-ème itération, la plupart des images ont perdu les détails correctes alors que les contours des visages sont plus semblables à la 24000-ème itération.



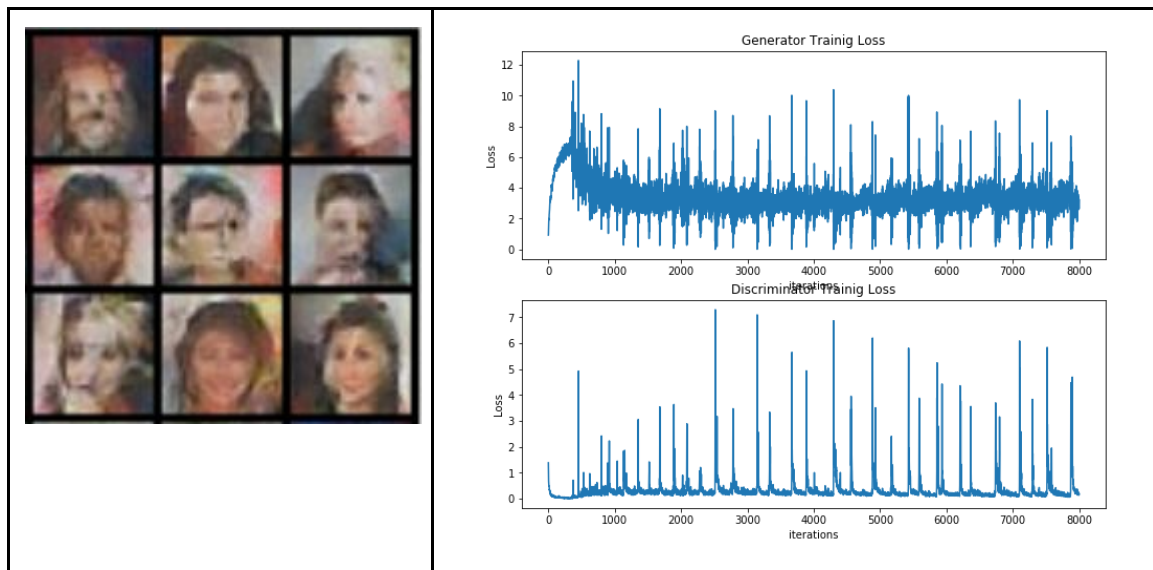
Les “training losses” ne sont pas améliorées non plus.

### Cas 2) la valeur de nz augmentée

- nz = 10000
- lrD = 0.0002 # Learning rate for the discriminator
- lrG = 0.0002 # Learning rate for the generator
- beta1G = 0.5 # Momentum beta1 for the discriminator
- beta1D = 0.5 # Momentum beta1 for the generator
- batch\_size = 128 # Images per batch
- steps = 8000 # Number of global steps in the training loop (5.1 epochs)

8000-ème itération	Loss
--------------------	------





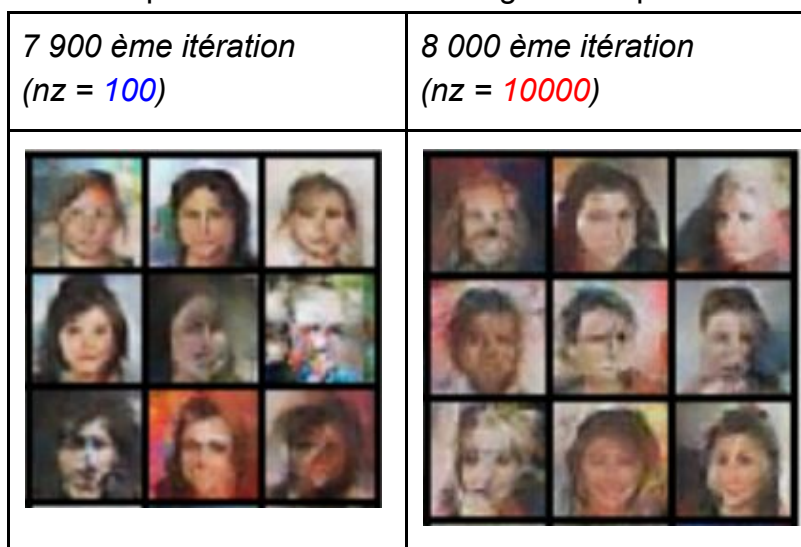
L'évolution des losses est très stable avec une valeur augmentée de ***nz*** mais les qualités d'image ne sont pas améliorées.

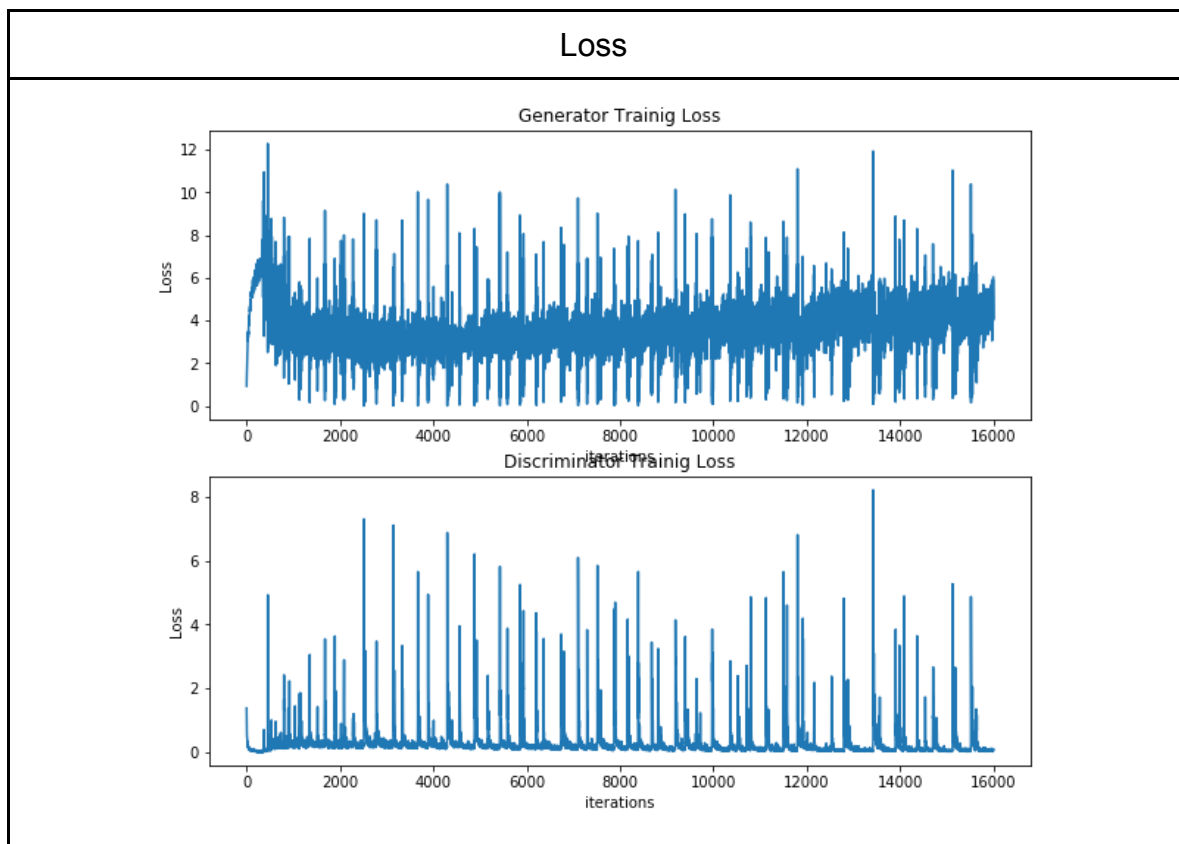
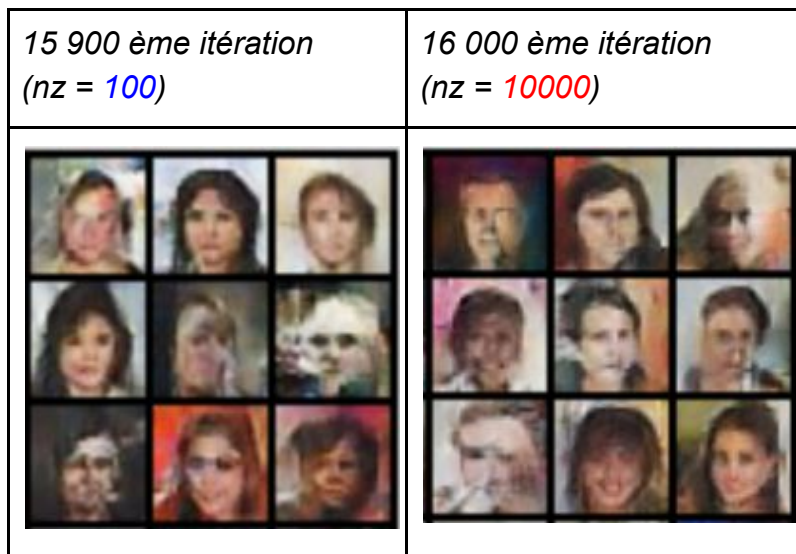
La vitesse d'entraînement devient extrêmement lente et le réseau avec la grande valeur de ***nz*** est plus mémoire intensif.

La diversité de génération est difficile à conclure car les sexes ne sont pas identifiable facilement.

### Cas 3) la valeur de ***nz*** augmentée / l'apprentissage pendant plus longtemps

- ***nz*** = 10000
- lrD = 0.0002 # Learning rate for the discriminator
- lrG = 0.0002 # Learning rate for the generator
- beta1G = 0.5 # Momentum beta1 for the discriminator
- beta1D = 0.5 # Momentum beta1 for the generator
- batch\_size = 128 # Images per batch
- steps = 16000 # Number of global steps in the training loop (10.2 epochs)



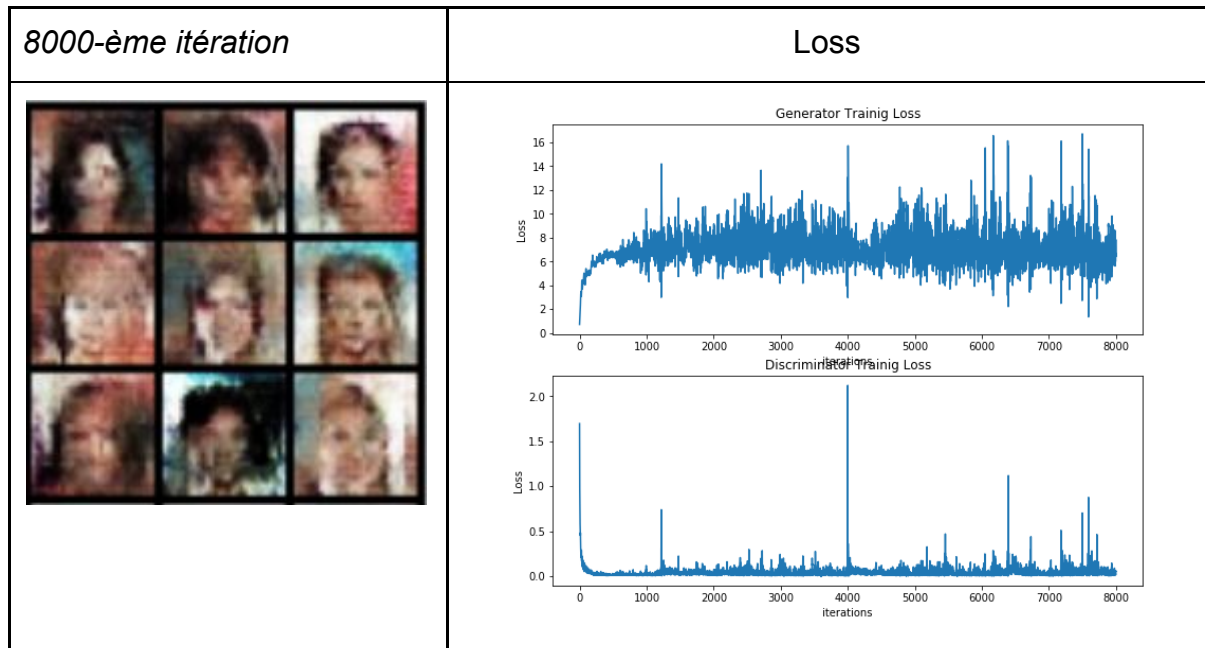


La moyenne de la generator loss augmente légèrement et progressivement. Pour la valeur de  $nz = 10000$ , certaines images montrent les détails plus clairs mais il est difficile de conclure sur la performance du réseau comme certaines images montrent aussi la mauvaise qualité.



#### Cas 4) Momentum augmenté

- $n_z = 10000$
- $lr_D = 0.0002$  # Learning rate for the discriminator
- $lr_G = 0.0002$  # Learning rate for the generator
- $\beta_{1G} = 0.9$  # Momentum  $\beta_1$  for the discriminator
- $\beta_{1D} = 0.9$  # Momentum  $\beta_1$  for the generator
- $batch\_size = 128$  # Images per batch
- $steps = 8000$  # Number of global steps in the training loop (5.1 epochs)



La stabilité de la discriminator training loss est très améliorée par rapport au cas 2 mais la generator loss monte. L'objectif du réseau est d'entraîner le générateur pas le discriminateur.

Dès qu'on a un discriminateur qui fonctionne plus ou moins bien, on veut minimiser la loss du générateur. Ce n'est pas le cas ici. Par conséquent, les images sont plus bruitées que les cas précédents.

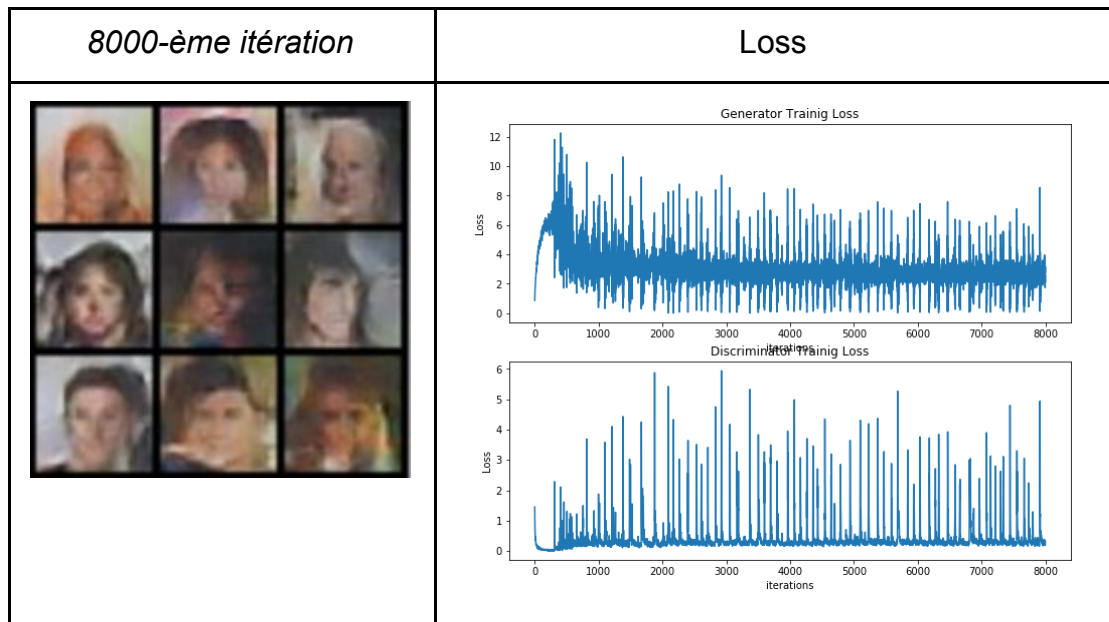
Le changement de la vitesse d'entraînement est peu sensible.

Dans les 2 cas suivants, on étudie l'impacte des momentums inégaux.

#### Cas 5) Momentums $D > G$

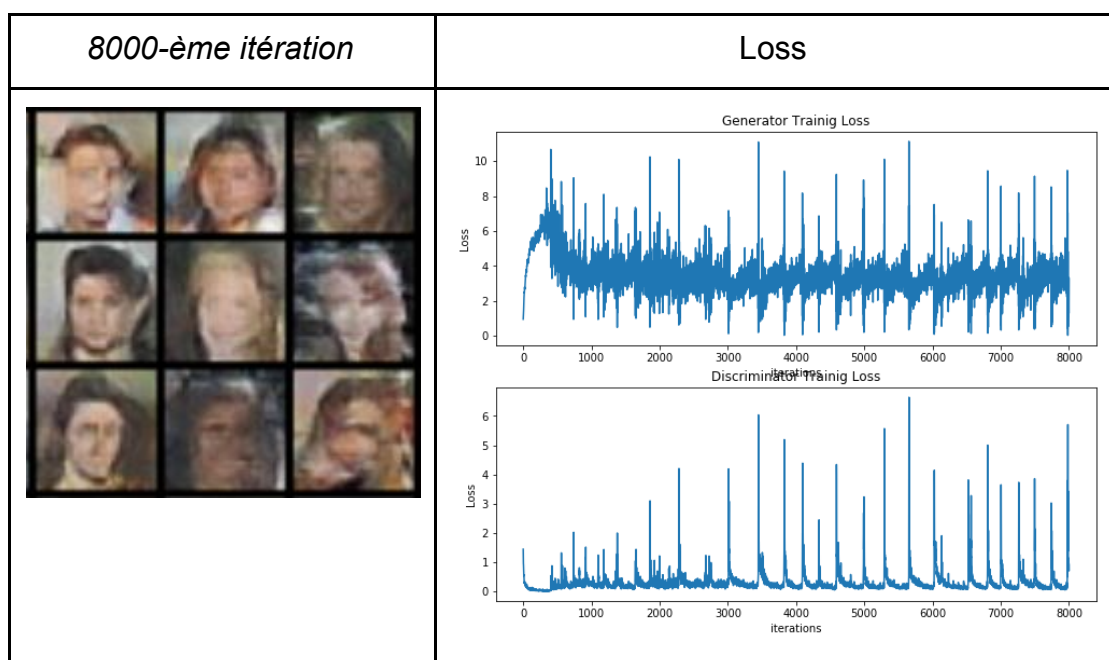
- $n_z = 10000$
- $lr_D = 0.0002$  # Learning rate for the discriminator
- $lr_G = 0.0002$  # Learning rate for the generator
- $\beta_{1G} = 0.6$  # Momentum  $\beta_1$  for the discriminator
- $\beta_{1D} = 0.2$  # Momentum  $\beta_1$  for the generator
- $batch\_size = 128$  # Images per batch

- steps = 8000 # Number of global steps in the training loop (5.1 epochs)



#### Cas 6) Momentums G > Momentums D

- nz = 10000
- lrD = 0.0002 # Learning rate for the discriminator
- lrG = 0.0002 # Learning rate for the generator
- beta1G = 0.2 # Momentum beta1 for the discriminator
- beta1D = 0.6 # Momentum beta1 for the generator
- batch\_size = 128 # Images per batch
- steps = 8000 # Number of global steps in the training loop (5.1 epochs)




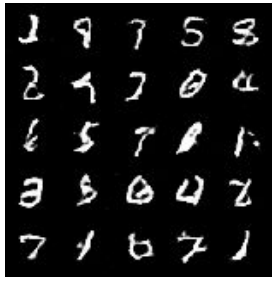



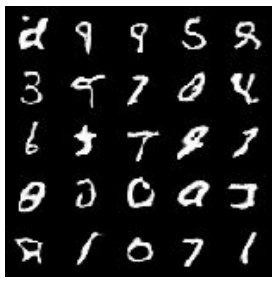






L'évolution de la discriminator training loss est plus stable avec le momentum de D augmentée mais il n'y a qu'une différence légère entre le **cas 5** et le **cas 6**. Il est difficile de conclure que le réseau est plus performant que le cas 3 où les momentums sont égaux à 0,5. Donc, on trouve que les valeurs de momentums à 0,5 sont les meilleurs compromis.

### Cas 7) MNIST

- lrD = 0.0002 # Learning rate for the discriminator
- lrG = 0.0002 # Learning rate for the generator
- beta1G = 0.5 # Momentum beta1 for the discriminator
- beta1D = 0.5 # Momentum beta1 for the generator
- batch\_size = 128 # Images per batch

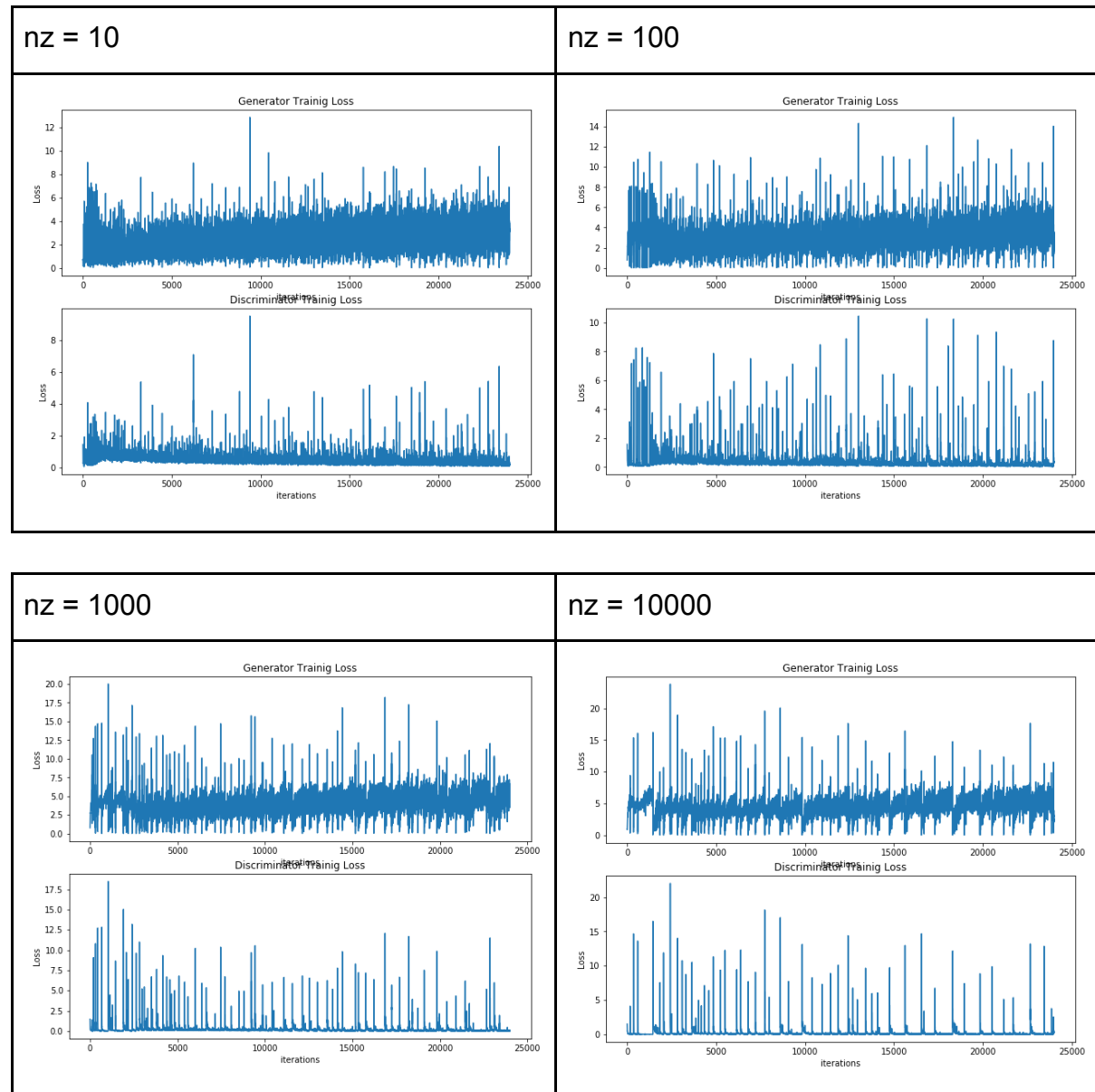
On entraîne le réseau sur l'ensemble de données MNIST pour mieux étudier la diversité de génération et la qualité visuelle des images en fonction des hyper-paramètres.

itération	nz = 10,	nz = 100	nz = 1000	nz = 10000
8000				
16000				
24000				

Pour la plus grande valeur de **nz**, les images de nombres générées sont les moins lisibles. La diversité de génération est peu influencée par cet hyperparamètre.

Le réseau se ralentit avec la grande valeur de ***nz***.

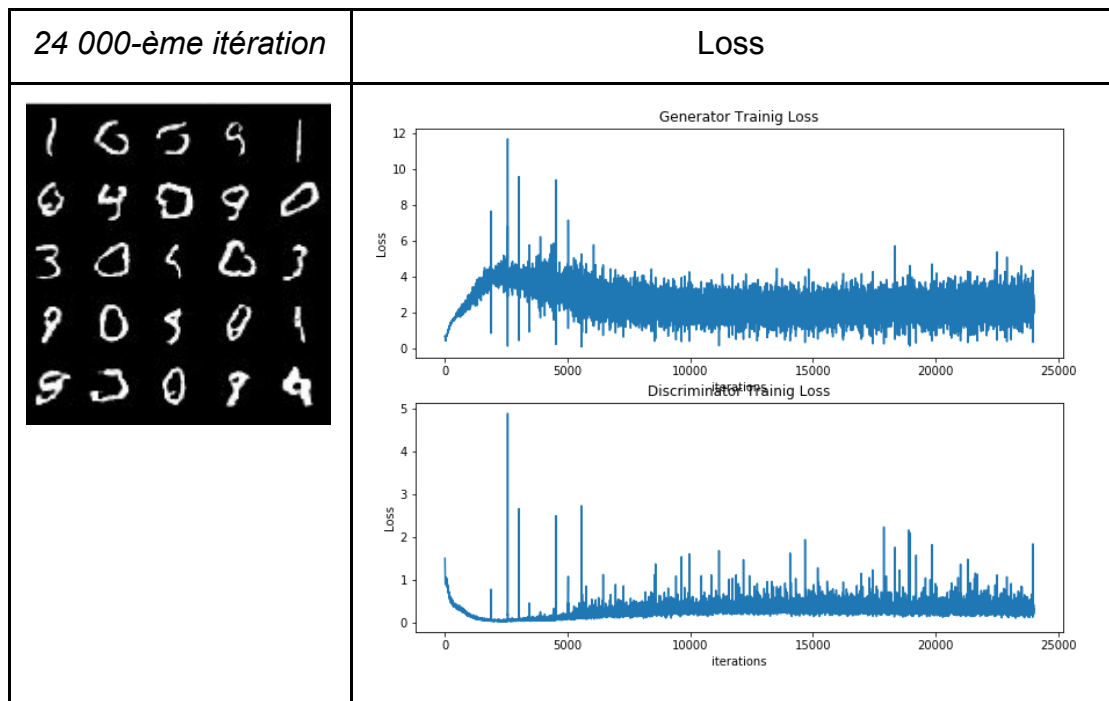
A 24000-ème itération, les images de nombres générées avec la valeur de ***nz*** 100 sont les plus lisibles.



En revanche, la grande valeur de ***nz*** permet d'améliorer la stabilité des courbes de loss comme nous constatons avec l'ensemble de données ***Celeba***

### Cas 8) Taux d'apprentissage diminué

- $n_z = 10$
- $lr_D = 0.00002$  # Learning rate for the discriminator
- $lr_G = 0.00002$  # Learning rate for the generator
- $\beta_1G = 0.5$  # Momentum  $\beta_1$  for the discriminator
- $\beta_1D = 0.5$  # Momentum  $\beta_1$  for the generator
- $batch\_size = 128$  # Images per batch

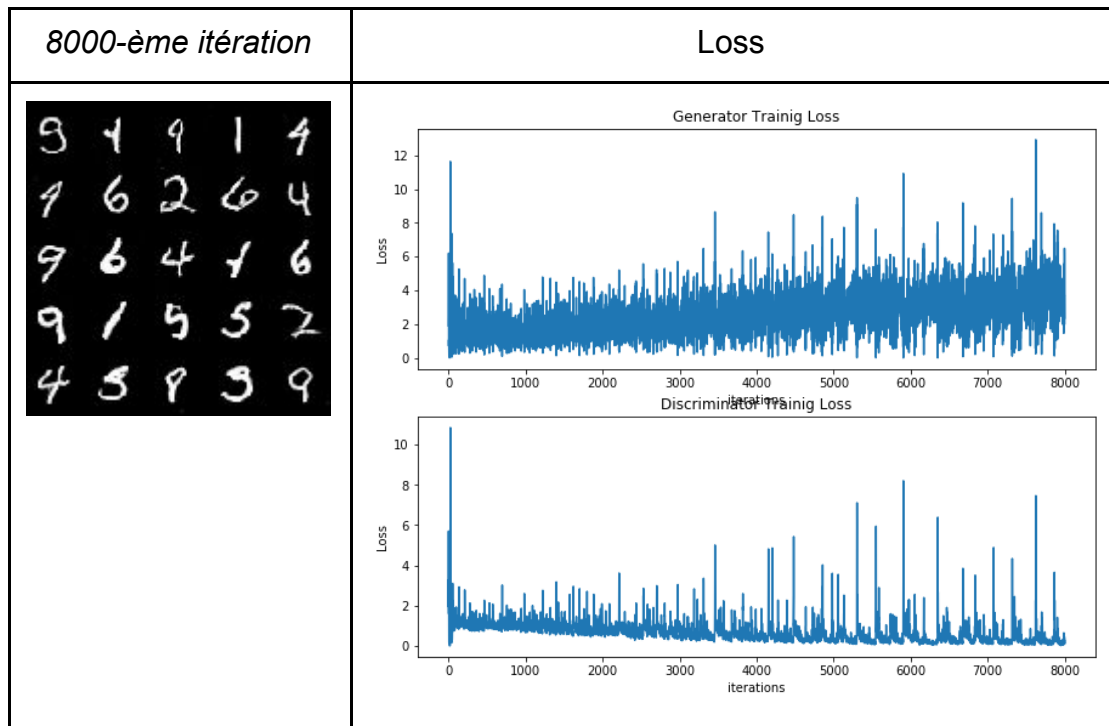


Le taux d'apprentissage diminué rend les images moins diverses. On constate qu'il n'y a que 4 chiffres (0, 1, 3 et 9). Les images sont devenues moins lisibles à la 24000-ème itération alors que les losses sont plus stables par rapport à l'entraînement avec le taux d'apprentissage 10 fois plus grand.




Le changement de la vitesse d'entraînement est peu sensible. Le taux d'apprentissage très bas n'est pas efficace.

### Cas 9) Taux d'apprentissage augmenté

- $nz = 10$
- $lrD = 0.002$  # Learning rate for the discriminator
- $lrG = 0.002$  # Learning rate for the generator
- $\beta_1D = 0.5$  # Momentum  $\beta_1$  for the discriminator
- $\beta_1G = 0.5$  # Momentum  $\beta_1$  for the generator
- $batch\_size = 128$  # Images per batch



À 8000-ème itération, on obtient les images bien lisible et diverses. Donc, même si il n'y pas de changement au niveau de la vitesse d'entraînement, cela permet de gagner du temps.

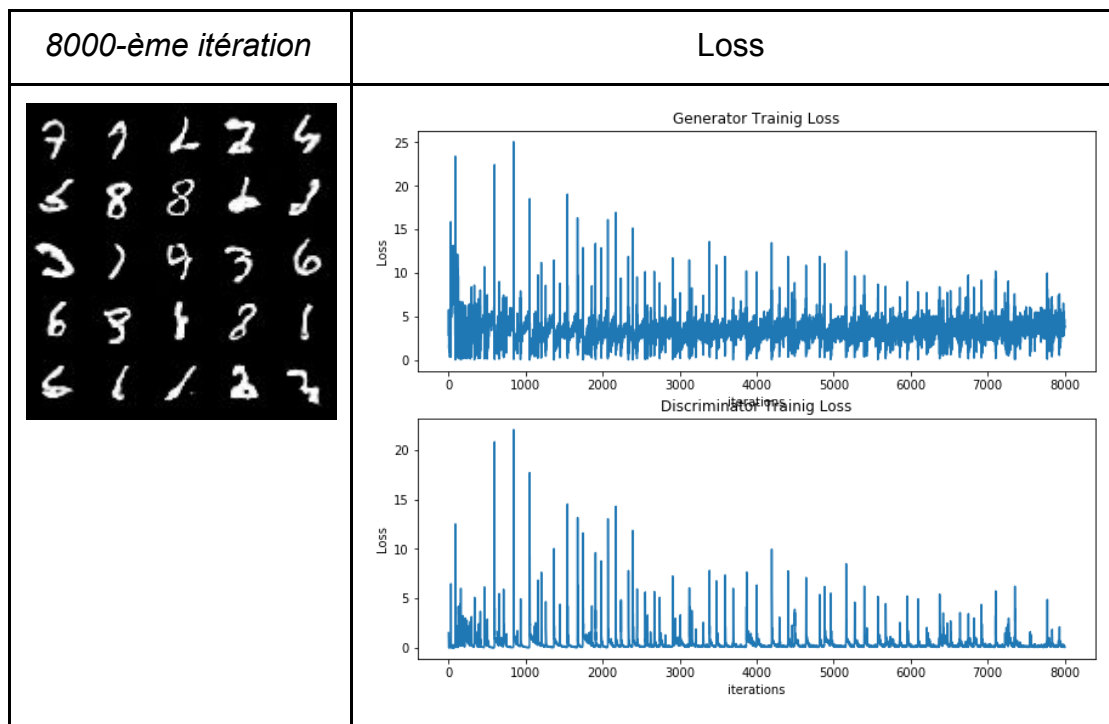
$nz = 10,$ $lrD = lrG = 0,00002$ 4000-ème itération	$nz = 10,$ $lrD = lrG = 0,0002$ 4000-ème itération	$nz = 10,$ $lrD = lrG = 0,002$ 4000-ème itération
		



On observe que le réseau avec le plus grand taux d'apprentissage donne la meilleure qualité visuelle des images à la même itération.

### Cas 10) Taux d'apprentissage D > Taux d'apprentissage G

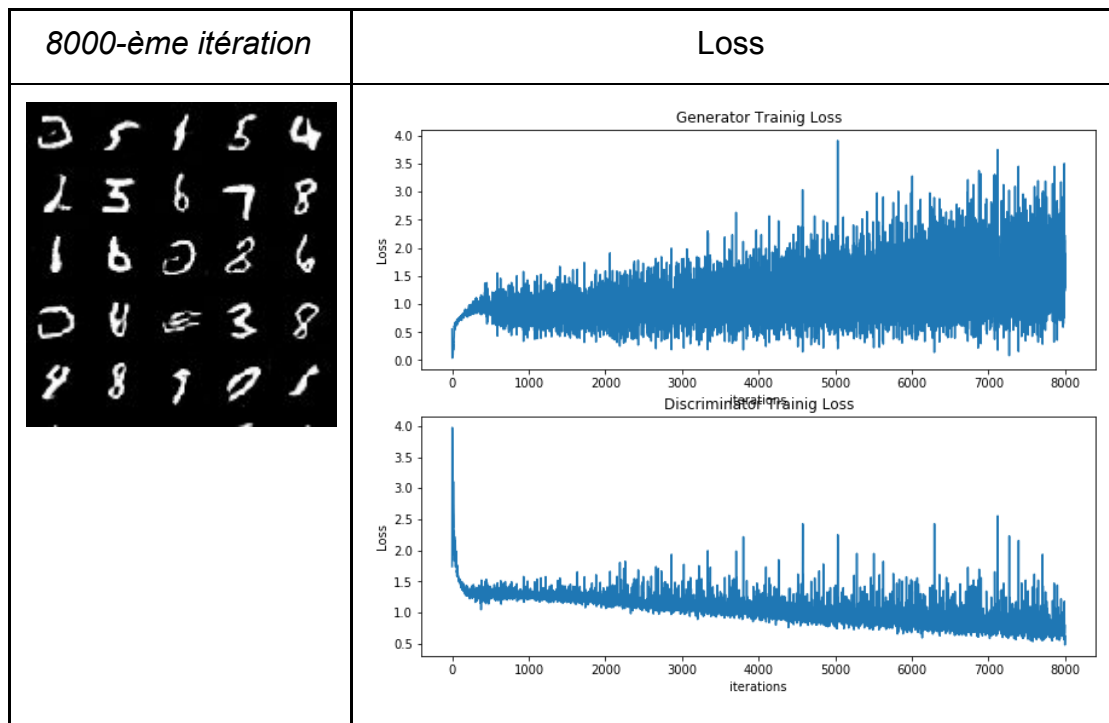
- nz = 10
- lrD = 0.002 # Learning rate for the discriminator
- lrG = 0.0002 # Learning rate for the generator
- beta1G = 0.5 # Momentum beta1 for the discriminator
- beta1D = 0.5 # Momentum beta1 for the generator
- batch\_size = 128 # Images per batch



La qualité d'image est clairement pire que dans les cas précédents. Ce résultat est attendu. L'objectif du réseau GAN n'est pas de différencier les images vraies et fausses mais de générer les images vraies. Donc, augmenter le taux d'apprentissage du discriminateur n'est pas un bon choix.

### Cas 11) Taux d'apprentissage G > Taux d'apprentissage D

- $n_z = 10$
- $lr_D = 0.0002$  # Learning rate for the discriminator
- $lr_G = 0.002$  # Learning rate for the generator
- $\beta_1D = 0.5$  # Momentum  $\beta_1$  for the discriminator
- $\beta_1G = 0.5$  # Momentum  $\beta_1$  for the generator
- $batch\_size = 128$  # Images per batch



La generator loss devient de plus en plus instable. La qualité visuelle n'est pas mieux que le **cas 11**.

### Conclusion)

La loss de générateur du réseau GAN est généralement très élevée et évolue de manière instable en général. Néanmoins, l'instabilité ne semble pas avoir beaucoup d'influence sur la qualité visuelle. Dans le **cas 7**, on observe que la grande valeur de  $n_z$  permet de stabiliser la loss mais la qualité devient que  $n_z = 100$ .

Entraîner le réseau pendant une longue durée n'est pas très efficace. L'amélioration est peu sensible avec les bons hyperparamètres.

Les **cas 5, 6, 10 et 11** montrent qu'il est important d'avoir un équilibre entre les hyperparamètres même s'il est plus important de mieux entraîner le générateur pour duper le discriminateur. Avoir les hyperparamètres correctes donne la bonne qualité visuelle en moins de 10000 itérations. Pour améliorer le résultat de l'entraînement

sur l'ensemble de données **Celeba**, on propose d'augmenter le filtre au lieu de modifier les hyperparamètres comme le taux d'apprentissage ou le momentum.

## Partie 2 – Conditional Generative Adversarial Networks

### 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)

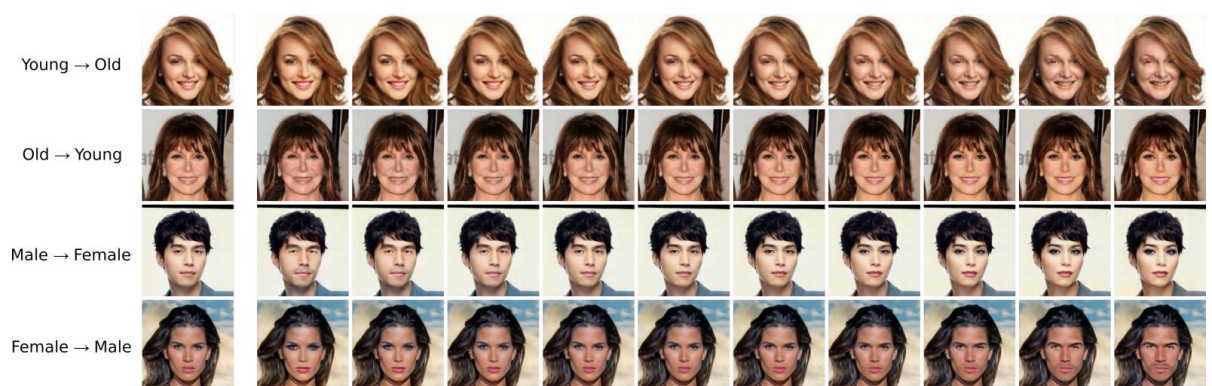
On cherche à optimiser le modèle avec les conditions. C'est-à-dire, on veut maintenant générer des images réalistes avec les attributs qu'on désire.

$$(5) \min_G \max_D \mathbb{E}_{x^*, y \in P_{Data}(x, y)} [\log(D(x, y))] + \mathbb{E}_{y \sim P_y, z \sim P_z(z)} [\log(1 - D(G(z, y), y))]$$

$$(6) \max_G \mathbb{E}_{y \sim P_y, z \sim P_z(z)} [\log(1 - D(G(z, y), y))]$$

$$(7) \max_D \mathbb{E}_{x^*, y \in P_{Data}(x, y)} [\log(D(x, y))] + \mathbb{E}_{y \sim P_y, z \sim P_z(z)} [\log(1 - D(G(z, y), y))]$$

### 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 ?



Il y a 4 variables qu'on pourrait conditionner :

- Âge
  - Jeune
  - Âgé
- Sexe
  - Male
  - Femelle

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



En fonction de la saison, on pourrait conditionner 4 variables : le printemps, l'été, l'automne, l'hiver.

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



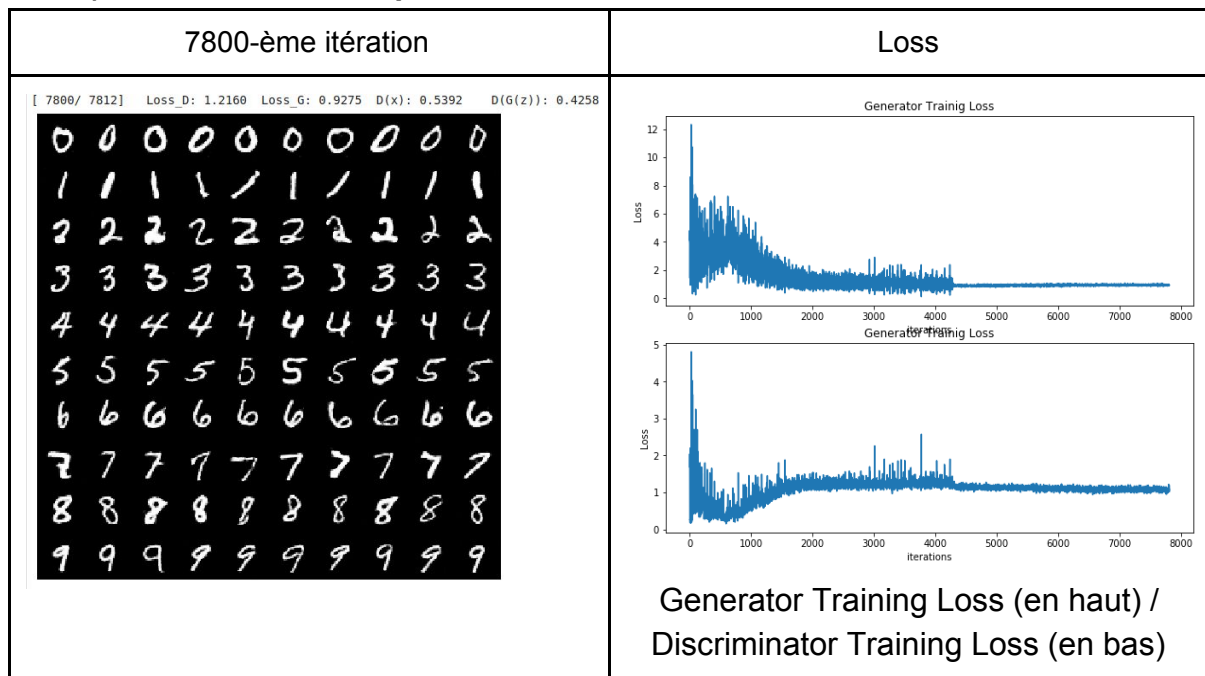
On constate qu'il y a 9 objets segmentés dans la figure à droite :

- Piétons
- Trottoir
- Voitures
- Motos
- Routes
- Arbres
- Feu tricolore
- Panneaux
- Bâtiment

Donc, on pourrait conditionner 9 variables. On peut conditionner plus de variables pour avoir une image plus détaillée. Par exemple, pour les voitures, on pourrait conditionner au moins 4 sous-variables :

- Convertible
- Bus
- Berline
- SUV
- etc

## 10) Commentez vos expériences avec le DCGAN conditionnel



En comparaison avec le GAN, le DCGAN conditionnel prend beaucoup plus de temps pour la génération d'images mais la qualité des images est aussi beaucoup mieux. Les chiffres sont bien lisibles et clairs. On constate que l'évolution des courbes des losses est très stable par rapport au GAN.

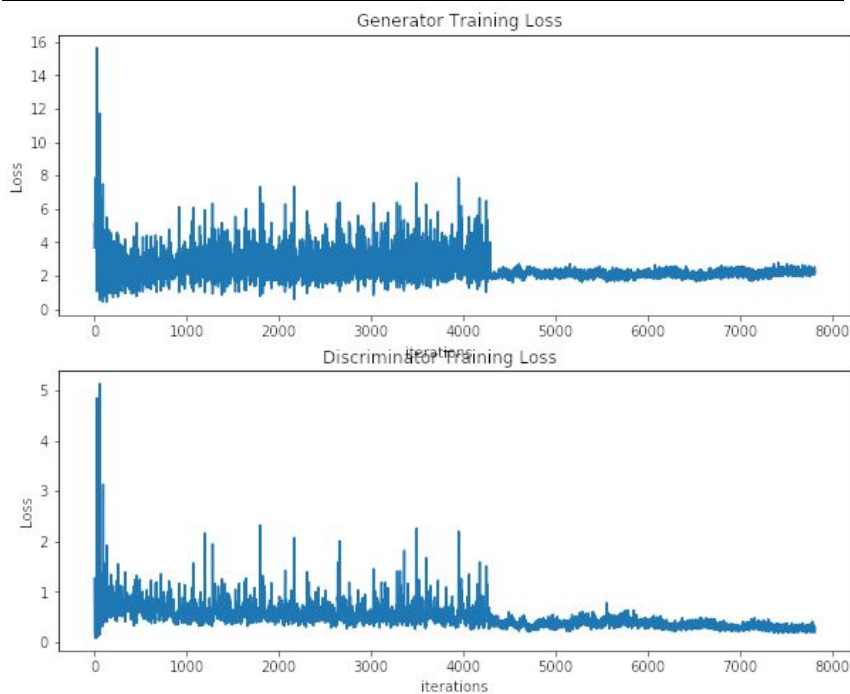
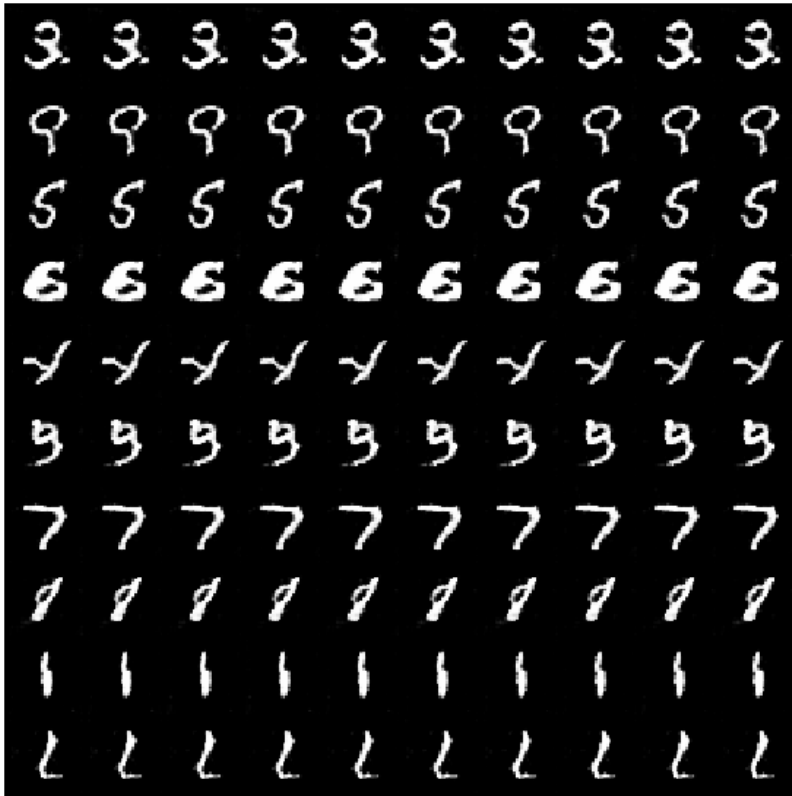
## 11) Pourrait-on enlever le vecteur $y$ des entrées du discriminateur (c'est-à-dire avoir $cD(x)$ et non $cD(x, y)$ ?

L'architecture de  $cD(x)$  ( l'architecture réduite de  $cD(x,y)$  )

```
Discriminator(
  (emb_xy): Sequential(
    (0): Conv2d(1, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2)
    (8): Conv2d(256, 1, kernel_size=(4, 4), stride=(2, 2))
    (9): Sigmoid()
  )
)
```



7800-ème itération



Techniquement, c'est possible, mais le réseau ne fonctionne pas bien comme cD(x) ne connaît plus la condition. Il n'extrait que les features pour savoir si l'image générée semble vraie, pas pour savoir s'il s'agit d'un chiffre.

Les images générées font des ressemblances d'un caractère mais la plupart d'entre elles n'a pas de sens.

Donc, on peut conclure que cD(x) ne marche pas.

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

L'image générée avec cGAN est en général plus bruitée et moins lisible. Le réseau est peu fiable. Il ne délivre pas le résultat de manière fixe. Par exemple, au bout d'un moment, le réseau peut générer des images qui ne contiennent que les bruits.

Son entraînement est beaucoup plus rapide par rapport à l'entraînement du cDCGAN.

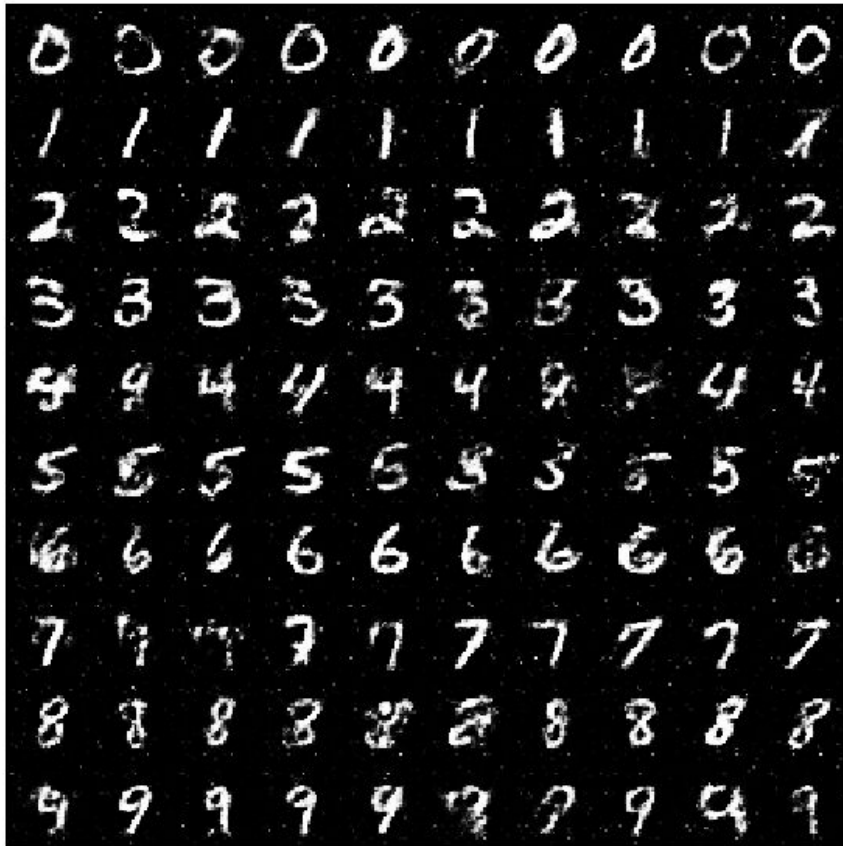
Nous avons pu obtenir les images plus ou moins lisible en moins de 1000 itérations.

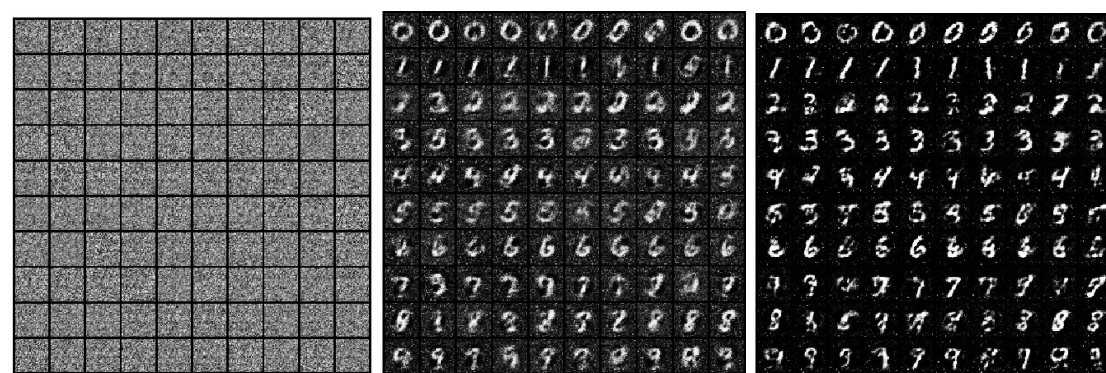
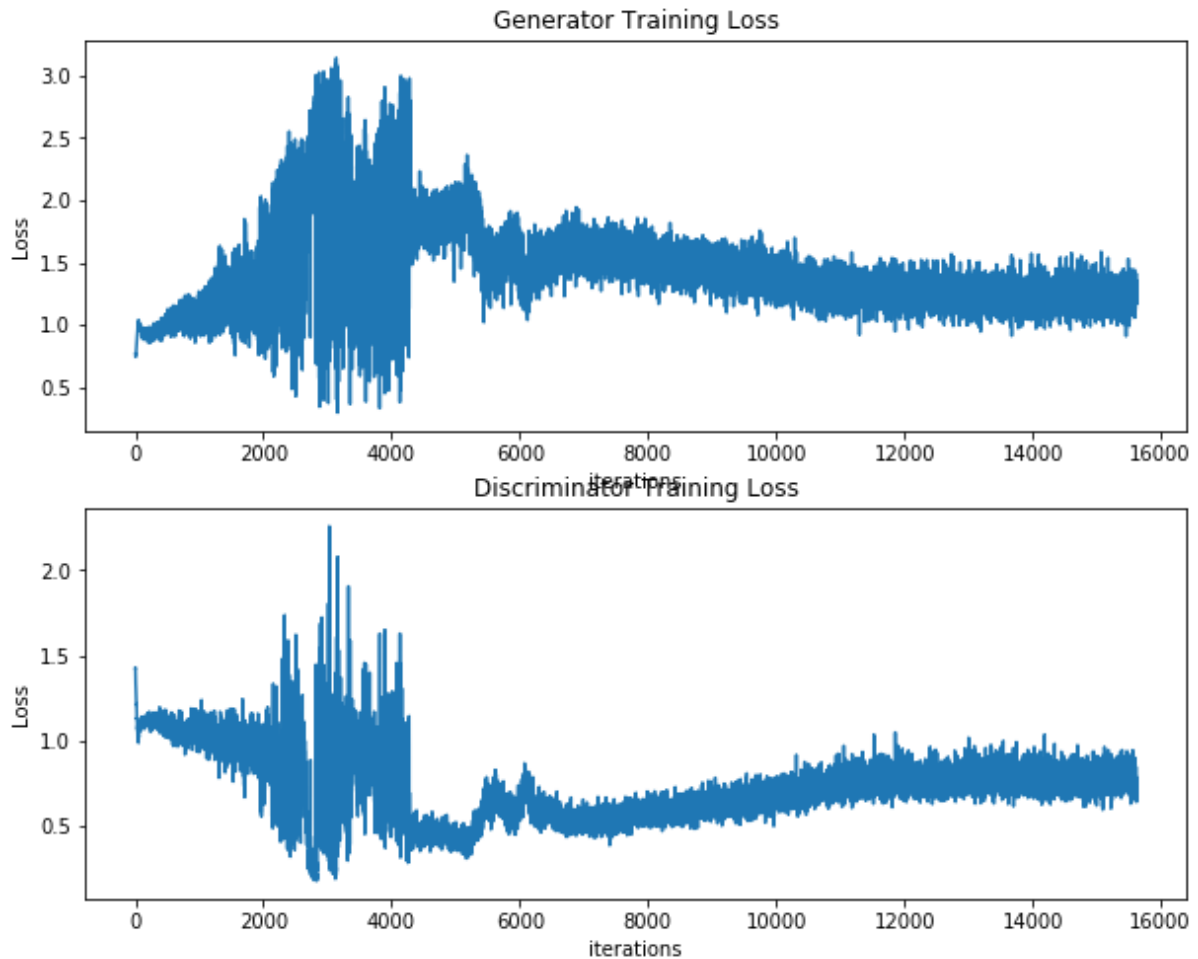
[ 400/15625] Loss D: 1.0914 Loss G: 0.9445 D(x): 0.5827 D(G(z)): 0.4133



# La meilleure génération de chiffres conditionnées

[15600/15625]    Loss\_D: 0.7363    Loss\_G: 1.1659    D(x): 0.7046    D(G(z)): 0.3020



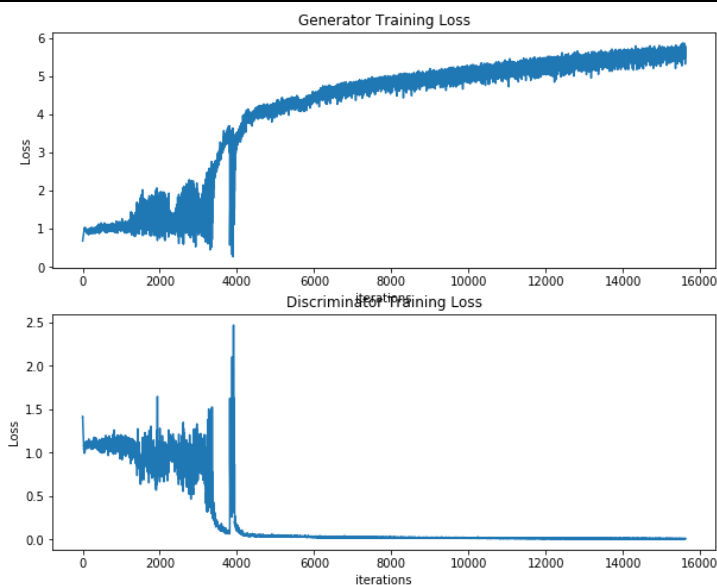
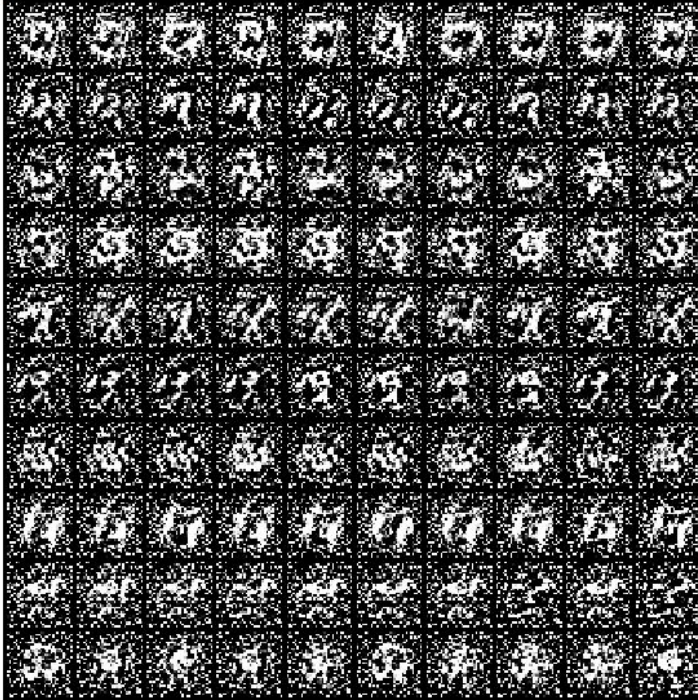


Les évolutions des images (0 -> 200 -> 1000-ème itération)

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

8000-ème itération

[ 8000/15625] Loss\_D: 0.0246 Loss\_G: 4.7234 D(x): 0.9862 D(G(z)): 0.0106



Le cGAN utilise les couches linéaires. Une couche linéaire applique une convolution sur une image entière pour sortir un pixel. Il est donc plus difficile d'extraire les features locaux. Par conséquent, le réseau n'est pas très robuste à des imprévus et il faut plusieurs couches intermédiaires et plus de poids à entraîner.