



UNIVERSITÉ  
**PARIS**  
**DESCARTES**

UNIVERSITÉ PARIS DESCARTES

MLSD 2 2018-2019

---

**Deep-Learning: réseaux  
antagonistes génératifs**

---

Blaize Hanczar  
Université Paris-Saclay  
Laboratoire IBISC

*Auteurs :*  
Joseph Gesnouin  
Yannis Tannier

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Materiel et Méthodes</b>	<b>3</b>
2.1	Dispositifs et outils de développement . . . . .	3
2.1.1	Base de données MNIST . . . . .	3
2.1.2	Réseaux de neurones - CNN . . . . .	3
2.1.3	GAN . . . . .	5
2.1.4	Machines utilisées . . . . .	8
2.2	Présentation des différentes approches considérées . . . . .	8
2.2.1	Baseline . . . . .	8
2.2.2	DC-GAN . . . . .	10
2.2.3	Info-Gan . . . . .	12
2.2.4	WGAN . . . . .	14
2.2.5	Notre méthode d'apprentissage semi-supervisée avec DCGANs	16
<b>3</b>	<b>Étude comparative des résultats obtenus</b>	<b>16</b>
<b>4</b>	<b>Pour aller plus loin: meilleur résultat</b>	<b>17</b>
<b>5</b>	<b>Conclusion</b>	<b>20</b>
	<b>Bibliographie</b>	<b>21</b>
<b>6</b>	<b>Annexe</b>	<b>22</b>

## 1 Introduction

L'apprentissage profond, bien qu'ayant véritablement explosé vers les années 2010, faisant des bonds de géant et écrasant l'état de l'art tenu par d'autres méthodes d'apprentissage pour des domaines tels que l'analyse d'images, du traitement automatisé du langage ou encore pour la reconnaissance faciale n'est en réalité pas une méthode si jeune.

En effet, les réseaux de neurones ont connu plusieurs périodes de posterité. Des années 50 en passant par les années 90 et finalement à notre décennie. Les neurones artificiels et leur système de couches ont également été ignorés voire oubliés pendant certaines périodes et ce pour plusieurs raisons qui ont freiné le développement de ce mode d'apprentissage.

Premièrement les réseaux de neurones demandent énormément de capacité de calcul, c'est ce qui leur a par exemple valu de tomber dans l'oubli au début des années 90, lorsque d'autres méthodes moins gourmandes ont vu le jour. Deuxièmement, ceux-ci nécessitent beaucoup plus de données labélisées pour apprendre comparé à d'autres méthodes capables d'extraire de l'information utile avec moins de data.

De nos jours, les réseaux évoluent et leur taille grandissant, cette considération computationnelle corrélée à la taille de ces réseaux demeure toujours un problème. Cependant, L'utilisation des GPUs pour paralléliser la majorité des calculs permet de minimiser cette contrainte moyennant un investissement certain.

À l'inverse, le nombre de données à labéliser pour obtenir des résultats cohérents reste un réel problème: labéliser des données coute cher, en temps et en argent. C'est ainsi qu'est née une volonté d'apprendre aussi bien avec moins de données labélisées dans l'état de l'art du Deep Learning.

C'est dans ce contexte d'apprendre en utilisant moins de données labélisées que ce rapport s'inscrit. Celui-ci se présentera sous forme d'une étude comparative de différentes méthodes d'apprentissage avec un jeu de données réduit: Ainsi, nous travaillerons sur le jeu de données MNIST en utilisant seulement une centaine de labels sur les 60 000 disponibles du jeu de test, et ainsi voir jusqu'où nous pourrions arriver avec seulement **0.16%** des labels du jeu d'apprentissage, tout en gardant en tête qu'avec la totalité des labels disponibles, un MLP et un CNN arrivent respectivement à environ **98%** et **99.7%** d'accuracy.

## 2 Materiel et Méthodes

### 2.1 Dispositifs et outils de développement

#### 2.1.1 Base de données MNIST

MNIST reste probablement le jeu de données le plus connu lorsqu'il s'agit de faire de l'apprentissage automatique. Celui-ci fait figure de jeu de test pour des nouvelles architectures afin de les comparer au même titre que le seraient Imagenet, Coco Dataset ou encore Yolo. Ce jeu de données se divise en un ensemble d'apprentissage de 60 000 individus et d'un ensemble de test de 10 000 individus. Actuellement à titre informatif, l'état de l'art sur ce jeu de données en prenant la totalité des labels est de **0.18%** d'erreur en utilisant le Random Multimodel Deep Learning (RMDL).

Dans notre cas d'utilisation, en selectionnant seulement 100 labels, l'idée n'est pas d'approcher ce résultat pour le moins excellent, mais plutôt de voir jusqu'où l'on peut faire grimper l'accuracy avec aussi peu de données.

MNIST disposant de 10 classes différentes réparties de manière équiprobable, nous aurions pu décider de manière à optimiser le training de choisir 10 instances de chaque classes, mais nous avons fait le choix de selectionner de manière aléatoire 100 labels en nous reposant sur le principe d'équiprobabilité.

Nous avions considéré le fait de pondérer notre choix des labels en fonction de la difficulté de reconnaissance des chiffres: par exemple, un 0 étant assez facilement distinguable, minimiser le nombre d'images de 0 sélectionnées afin de prendre plus de 3 et de 9 par exemple. Ces deux chiffres étant foncièrement similaires, cela nous aurait probablement permis d'optimiser nos résultats. Cependant considérant ce projet comme un projet de découverte nous n'avons pas souhaité biaiser l'apprentissage dès le début en fonction de nos connaissances sur le jeu de données: lorsqu'il s'agit de traiter des données en production, nous n'avons généralement pas tant d'information et c'est pourquoi nous n'avons pas fait ce choix.

#### 2.1.2 Réseaux de neurones - CNN

Nous avons fait le choix de ne pas considérer les architectures MLP même pour la méthode baseline: celle-ci étant moins aboutie et moins précise que les réseaux de neurones convolutionnaires avec la totalité des labels, il semblait évident que celle-ci serait limitée vis-à-vis des CNN utilisés principalement pour la reconnaissance d'image.

Lors de ce rapport nous aurions pu faire face à certains problèmes relatifs à l'apprentissage profond qui auraient pu fausser nos résultats, même si nos réseaux sont relativement petits, il est toujours formateur de prendre en compte des situations plausibles d'erreur et de savoir comment les éviter en prenant les bonnes habitudes:

- **L'explosion des gradients:** Les gradients peuvent s'accumuler pendant une update. En résultera un gradient de taille conséquente. Ce qui entraînera à terme un grand changement dans les poids du réseau, ayant pour finalité un réseau instable incapable d'apprendre ou dans le meilleur des cas, simplement incapable d'apprendre d'une longue séquence de données.
- **Disparition des gradients** Problème littéralement opposé au précédent, durant la rétropropagation, les petits gradients se multipliant entre eux peuvent disparaître au fil des couches. Empêchant le réseau d'apprendre de dépendances sur plusieurs couches. Une des solutions possibles reste l'utilisation de fonction d'activation comme ReLU qui ne souffre pas des petits gradients.

Pour éviter le surapprentissage ou l'absence d'apprentissage nous avons donc utilisé plusieurs méthodes classiques de deep-learning afin d'optimiser nos modèles.

- **Regularisation L1 & L2:** Nous permettant de contrecarrer cet effet d'explosion des gradients en vérifiant la taille des poids du réseau et en appliquant une pénalité en fonction de la valeur de la taille des poids.
- **Dropout:** Nous permettant de réduire le surapprentissage dans les réseaux. La technique évite des co-adaptations complexes sur les données de l'échantillon d'entraînement. Le réseau neuronal se voit amputé d'une partie de ses neurones pendant la phase d'entraînement, leur valeur étant à ce moment estimée à 0. Ces neurones étant par contre réactivés pour tester le nouveau modèle.
- **Batch Normalisation:** est une technique pour améliorer la performance et la stabilité de notre réseau. Elle permet de nourrir chacune des couches du réseau avec des données centrées réduites et donc d'ajuster et de mettre à l'échelle les activations.
- **Data Augmentation:** Ne disposant que d'une centaine d'images labellisées, nous ne disposons pas d'une infinité de points. Nos modèles auraient donc été sujets au sur-apprentissage assez facilement. En principe, plus le nombre de données est élevé, plus le modèle final sera précis. Comme précisé précédemment, la collection des données a un coût. De ce fait, il est toujours intéressant d'augmenter artificiellement les données existantes afin d'augmenter le jeu de données d'apprentissage final qui sera envoyé au modèle lors de l'apprentissage. Il existe une multitude de méthodes afin d'augmenter artificiellement ses données: dans le cadre des images, il est possible de changer la luminosité, découper

l'image différemment, et si le cas s'y prête, utiliser des rotations...

### 2.1.3 GAN

Les GANs ont marqué une petite révolution dans le monde de l'apprentissage automatique en 2014. Ceux-ci n'ont cessé d'être sujets à diverses améliorations au travers d'articles depuis la publication de l'idée originale. À ce jour il existe des centaines d'articles sur les GANs devenant même une blague récurrente dans le Deep Learning: écrire un papier sur les GANs qui ne serait "pas simplement qu'un autre article sur les GANs".

Toute cette euphorie s'explique très simplement par cette capacité incroyable que les GANs ont. En voulant imiter n'importe quel type de données (images, texte, son, vidéo), les GANs permettent de générer des nouvelles données plus vraies que nature.

En plus de cette caractéristique appréciable et non négligeable de création artificielle de données, ceux-ci ont l'énorme avantage de pouvoir s'entraîner sur des images non-labélisées et donc font partie du domaine d'apprentissage que l'on considère comme non supervisé.

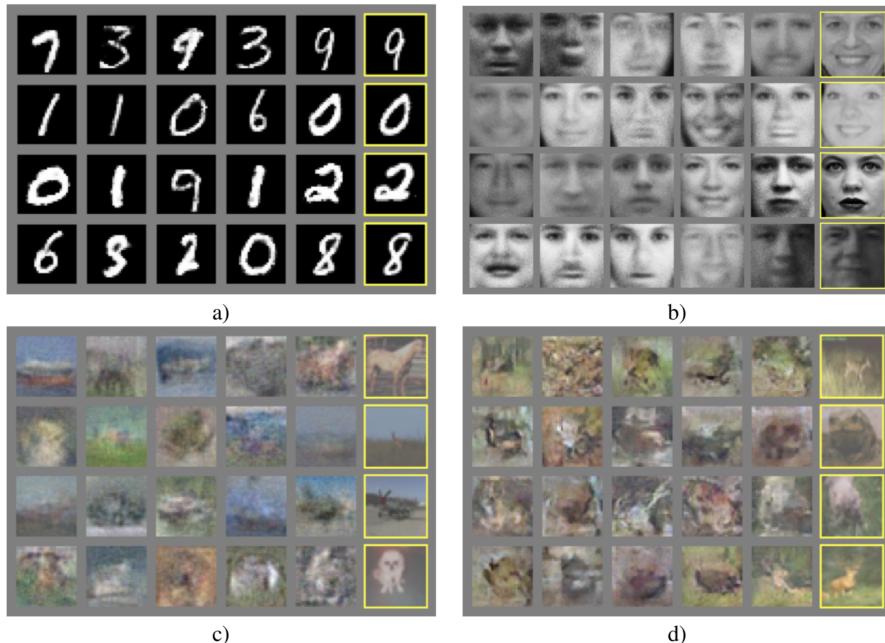


Figure 1: Visualisation d'images provenant du papier initial sur les GANs

Le fonctionnement des GANs est assez simple: il se base sur la théorie des jeux en considérant deux individus:

- On introduit un **générateur**, techniquement il se caractérise par un réseau de neurones générateur en charge de créer des données "fausses". Le tout en essayant d'imiter au mieux les données pour qu'elles semblent réalistes.
- Le deuxième joueur est un second réseau de neurones adversarial, communément appelé le **discriminateur**, en charge d'aider le générateur à déterminer ce qu'il réalise correctement et à l'inverse, lui spécifier les points sur lesquels il pourrait s'améliorer.

À eux deux, ces individus continuent à "jouer" jusqu'à ce que le discriminateur peine à distinguer les vrais des faux. Entrainer le générateur se résume alors à fournir des gradients calculés par un adversaire qui aura préalablement appris à noter le niveau de ressemblance entre une donnée authentique et une donnée fausse. Plus précisément, le discriminateur calcule une probabilité que l'image qu'on lui envoie soit falsifiée et est entraîné à donner des probabilités adéquates en pénalisant la perte logarithmique moyenne quand on lui expose aléatoirement des images authentiques et des fausses images.

De cette manière un apprentissage GAN peut se résumer en deux parties:

- Un apprentissage de l'adversaire à distinguer les authentiques et les faux avec des pertes logarithmiques.
- Un apprentissage du générateur en maximisant l'authenticité des images générées artificiellement selon l'adversaire. Une fois que le générateur s'améliore et génère des images de plus en plus plausibles, on reboucle sur la première étape.

Ces deux étapes sont directement corrélées et devront être répétées des milliers de fois avant d'obtenir des images tellement convaincantes que même l'adversaire ne sera plus capable de différencier les données réelles des données falsifiées.

$$\min_G \max_D \left[ \mathbb{E}_{x \sim p_x} \log D(x) + \mathbb{E}_{z \sim p_z} \log(1 - D(G(z))) \right]$$

$\underbrace{\phantom{\dots}}_{\text{Sortie du discriminateur pour des données réelles}}$ 
 $\underbrace{\phantom{\dots}}_{\text{Sortie du discriminateur pour des données artificielles}}$

Figure 2: Fonction de coût: D représente la probabilité que la donnée soit réelle

Le discriminateur et le générateur essayant succintement de minimiser et de maximiser cette fonction de coût, il est important d'alterner l'apprentissage des deux

réseaux.

À titre d'exemple, des chercheurs de NVIDIA ont réussi en 2017 à générer des images de célébrités créées de toute pièce par ce type de réseau:



Figure 3: Deux célébrités imaginaires générées par un GAN: Progressive Growing of GANs for Improved Quality, Stability, and Variation (NVIDIA)

Autre exemple bluffant: un des derniers papiers présentés en décembre 2018 un peu après la conférence NeurIPS: *A Style-Based Generator Architecture for Generative Adversarial Networks* propose d'aller encore plus loin. On peut alors contrôler encore plus précisément certains traits du visage et à terme possiblement contrôler tout et ainsi générer des images comme on le souhaite:



Figure 4: Les images générées offrent une énorme diversité en terme d'âge, d'éthnicité, d'angle de vue, de luminosité et de fond.

Dans notre cadre d'utilisation des GANs, nous nous focaliserons principalement sur des données de type image, cependant les GANs ne se contentent pas simplement à de la génération de données imagées. Dans la littérature, le nombre de cas d'usages des GANs croît de manière exponentielle: NLP, texte, vidéo... Leur fiabilité étant telle que même le CERN utilise ce type de réseau afin de minimiser l'utilisation de leur grand collisionneur de hadrons tout en travaillant sur des simulations similaires générées par ces réseaux.

### 2.1.4 Machines utilisées

La capacité de calcul nécessitée pour l'apprentissage profond étant une des problématiques à prendre en compte, nous avons eu l'occasion de faire la totalité de nos appren-tissages sur une RTX 2080 TI. De ce fait nous avons eu le temps, et l'avantage d'essayer une multitude de modèles différents, en faisant varier les paramètres à outrance mais également en ne lésinant pas sur le nombre d'epoch utilisés pour chaque modèle. Cette carte graphique dispose d'une micro-architecture Turing développée par Nvidia, de 4352 coeurs Cuda ainsi que de 11Go de ram.

## 2.2 Présentation des différentes approches considérées

### 2.2.1 Baseline

La première méthode considérée aura été de d'utiliser une approche basique en utilisant une architecture CNN classique: une succession de couches de convolution et de max pooling.

**Code source.**

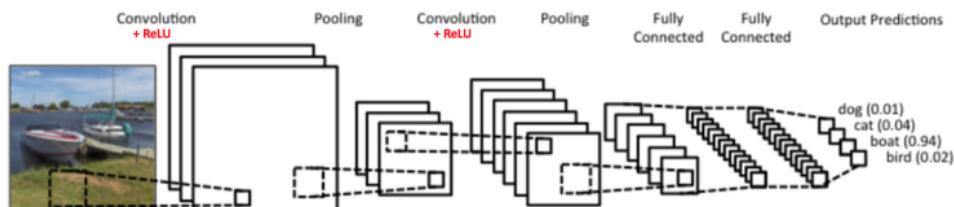


Figure 5: Lenet: un des premiers ConVnet

Afin d'éviter les problèmes présentés précédemment nous avons choisi d'ajouter du dropout et de la batch normalisation pour la régularisation entre les couches de convolutions et de pooling. Avec notre premier réseau convolutionnaire entrainé en utilisant seulement les 100 labels sélectionnés nous obtenons une accuracy de **90.12%**. Cette valeur devenant donc notre valeur baseline que les modèles suivants seront censés surpasser.

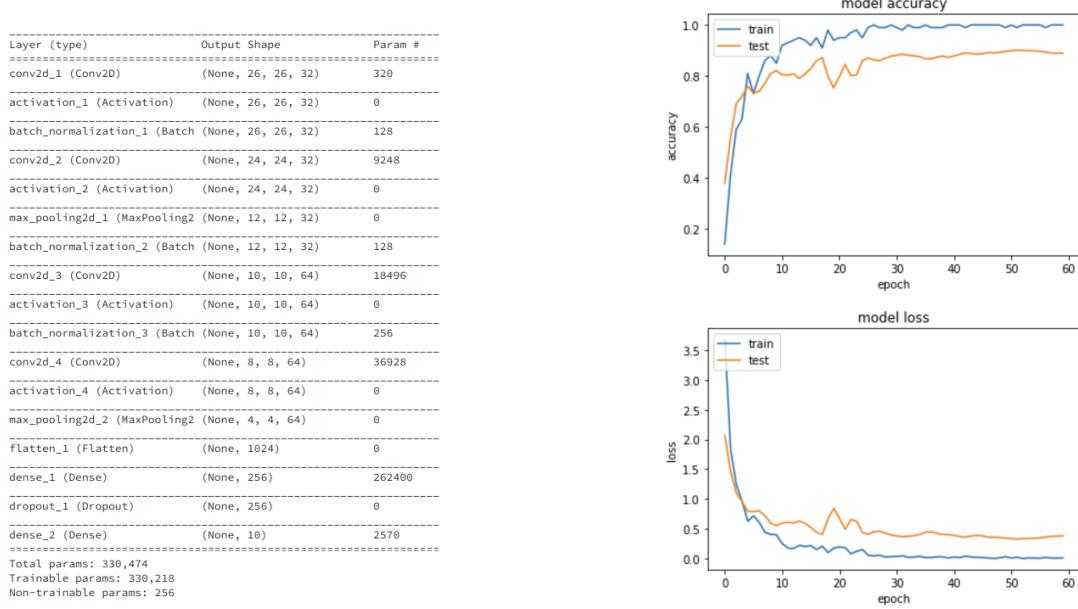


Figure 6: architecture et courbes d'apprentissage du réseau CNN utilisé pour la baseline

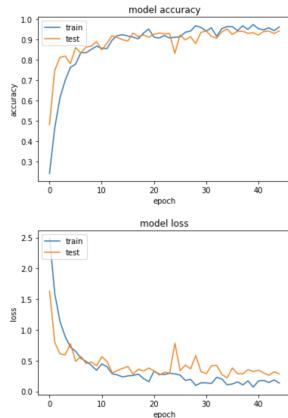


Figure 7: courbe d'apprentissage du réseau CNN utilisé + data augmentation

Afin de voir si nous pouvions améliorer ce score en utilisant la data augmentation, nous avons utilisé la même architecture en générant artificiellement de nouvelles images grâce à des procédés de traitement d'image. La data augmentation nous aura permis faire grimper l'accuracy jusqu'à **95.02%**, le gain obtenu n'est pas négligeable pour un simple procédé de traitement d'images.

Bien qu'assez intéressantes, ces valeurs peuvent potentiellement être améliorées en utilisant la puissance de l'apprentissage non supervisé des GANS. Nous permettant d'apprendre une représentation des données. Grâce au générateur et au discriminateur de ces réseaux, l'amélioration de la représentation des données est croissante, rendant de plus en plus dure voir impossible la distinction entre une image réelle et une image générée.

Pour toutes les méthodes à base de GANs, nous procéderons de la même manière: Après avoir entrainé le reseau GAN selectionné, nous récupererons le discriminateur du reseau utilisé qui aura alors appris une sémantique des images. Puis, on entrainera ce discriminateur pré-entraîné avec nos 100 images en voyant jusqu'où cela peut mener l'accuracy.

Le réseau de neurones du discriminateur est capable d'apprendre une représentation robuste abstraite des images en devenant progressivement meilleur pour discriminer les données initiales des données artificielles. Qu'importe ce que ce réseau apprend des images non labélisées pour apprendre à distinguer de manière binaires les vrais des faux, il est fort probable que ces images vont participer à la sélection de features utiles à la distinction des images labélisées. Il était donc envisageable d'utiliser le discriminateur pour notre classification: celui-ci n'étant pas nécessairement seulement limité à juste délimiter les vraies images des fausses. Nous avons donc également décidé de l'entraîner afin de classifier les vraies données.

### 2.2.2 DC-GAN

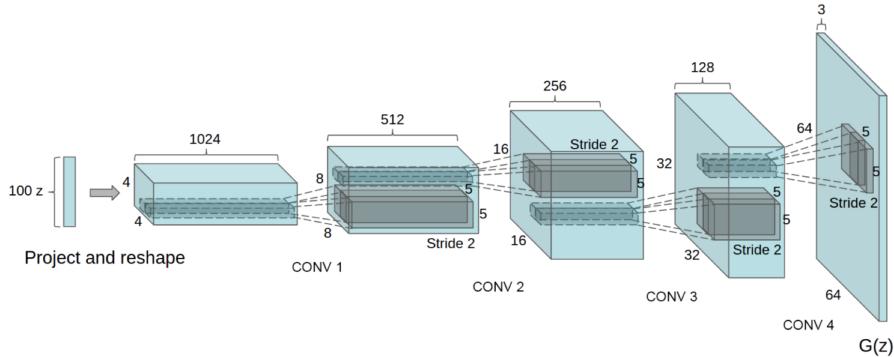


Figure 8: Architecture DC-GAN

DC-GAN est un des réseaux GAN les plus populaires et les plus efficaces. Il est principalement composé de couches de convolutions sans max pooling ni de couches fully connected.

**Code source.**

Le principe d'utilisation de ce réseau étant d'envoyer un latent space aléatoire au générateur et celui-ci générera l'image à partir de ce latent space. Le principe est

assez simple et efficace, cependant l'utilisateur n'a aucun contrôle sur ce qu'il envoie au réseau.

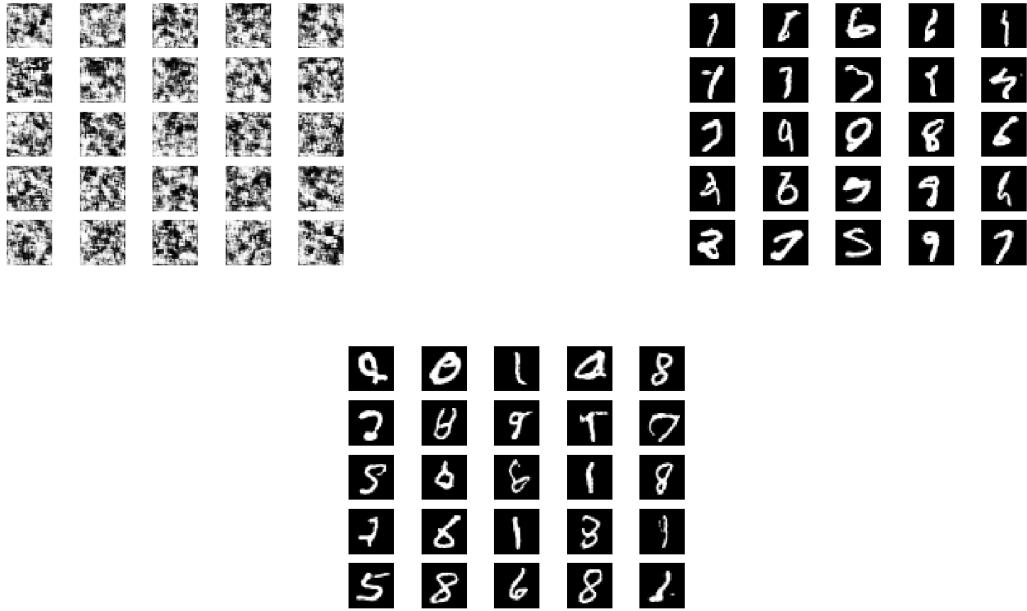


Figure 9: Images générées par notre implementation du DC-GAN: epoch 0, 9000, 19500 [Link](#).

Après avoir laisser jouer le discriminateur et le générateur plus de 20000 fois, nous disposons d'images suffisament plausibles et d'un discriminateur suffisament entrainé pour passer à l'étape suivante de notre training: récupérer le discriminateur et remplacer la dernière couche par une couche Dense de 10 sorties représentant les 10 classes du jeux de données que nous souhaitions modéliser.

Sans utiliser le principe de data augmentation, l'implementation de DC-GAN nous aura permis de faire monter l'accuracy à **94.16%**. Au final l'implementation de cette méthode nous aura permis de grapiller plus de **4%** d'accuracy vis-à-vis de notre méthode baseline avec seulement 100 labels.

De la même manière, nous avons rééssayé cette méthode, tout en réutilisant le principe de data augmentation présenté et utilisé précédemment, ce qui nous permet d'obtenir une accuracy de **95.47%**. De la même manière que sans la data augmentation, nous remarquons une amélioration de l'accuracy grâce à l'utilisation des GANs en ajoutant le principe d'augmentation de l'ordre de **0.45%**

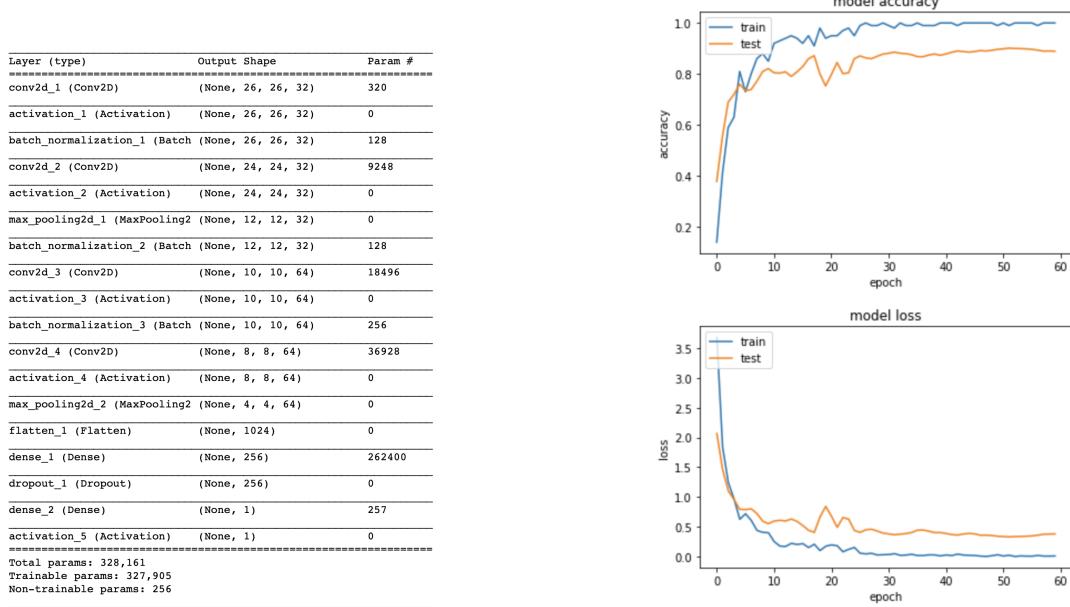


Figure 10: architecture et courbes d'apprentissage du réseau créé à parti du discriminateur provenant du modèle DC-GAN

L'apport des GANs vis-à-vis de la baseline permettant de grappiller des précieux pourcentages d'accuracy, nous avons souhaité essayer différentes méthodes GAN afin de voir si nous pouvions réussir à faire grimper le plus possible notre accuracy.

### 2.2.3 Info-Gan

Info-Gan fonctionne de manière similaire à DC-GAN, en ajoutant une information à priori sur le nombre de classes en entrée, s'inspirant alors de CGAN. Ainsi, on envoie toujours un latent space aléatoire au générateur, mais également un paramètre supplémentaire représentant le label qui sera inféré aléatoirement nous évitant d'utiliser des images labélisées contrairement à CGAN, architecture non présentée dans ce rapport car étant supervisée et donc nécessitant la classe de chaque image. InfoGAN à l'inverse, essaie de créer ces conditions de manière automatique afin d'éviter de spécifier au GAN à quelle classe chaque instance appartient.

Dans la théorie de l'information, il est possible d'exprimer la connaissance que l'on a d'une instance grâce au savoir dont on dispose à propos d'une autre instance fondamentalement différente, en utilisant le principe d'information mutuelle. De ce fait, en maximisant cette information mutuelle, on peut en déduire quelque chose qui

pourrait potentiellement contribuer à améliorer la connaissance d'une instance relativement différente. Dans le cas d'Info-GAN, on cherche à maximiser le savoir dont on dispose sur nos variables conditionnelles.

Nous avons commencé par entraîner nos deux joueurs pendant un certain nombre d'epochs afin que le discriminateur ait une représentation fiable des images, puis, contrairement à précédemment, le discriminateur d'Info-GAN dispose de deux types de sorties: la sortie binaire correspondant à la fiabilité de réalité de l'image, et le label de l'image envoyée, nous avons donc récupéré la dernière sortie du discriminateur d'Info-GAN.

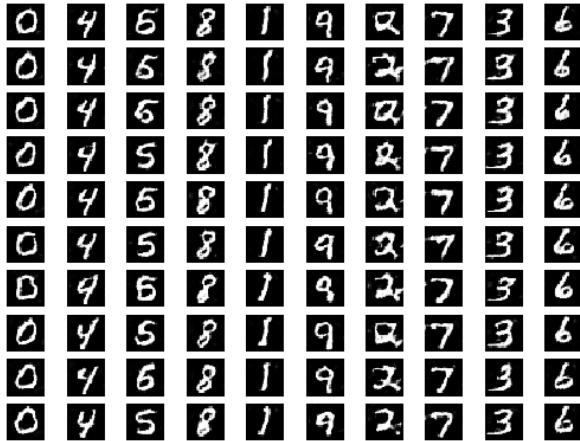


Figure 11: Images générées par notre implémentation d'Info-GAN: chaque colonne représentant une représentation d'une certaine classe **Link**.

Comparé aux images générées par DC-GAN, celles-ci semblent bien plus réelles et sont difficilement criticables concernant leur potentielle véracité. Ainsi le simple ajout de ce second paramètre de manière automatique, créant de l'information supplémentaire pour les instances, rend la génération des images encore plus précise que le modèle présenté précédemment.

Sans utiliser le principe de data augmentation, l'implémentation d' Info-GAN nous aura permis d'obtenir une accuracy à **87.92%**. Sensiblement plus faible que les deux autres résultats obtenus précédemment. De la même manière, nous avons rééssayé cette méthode, tout en réutilisant le principe de data augmentation, obtenant alors une accuracy de **94.81%**.

**Code Source.**

### 2.2.4 WGAN

Dans les deux réseaux GAN présentés précédemment, un problème perssistant est lié à l'entropie: le but du jeu étant de minimiser le plus possible la divergence de Kullback-Leibler entre la distribution generative et les données réelles. Si la divergence KL est à sa position minimum, la probabilité des données générées est alors égale à celle des données réelles.

$$KL(\mathbb{P}_r \parallel \mathbb{P}_g) = \int_{\mathcal{X}} P_r(x) \log \frac{P_r(x)}{P_g(x)} dx$$

Figure 12: formule de la divergence KL

Le modèle aura tendance à générer des images sensiblement similaires les unes aux autres. Cependant dans ce cas particulier, les images générées seront quand même considérées aussi vraies que les autres. La divergence KL étant minime, le modèle n'accordera pas une grande pénalité lors d'une telle situation sans pour autant foncièrement améliorer le modèle.

Plusieurs métriques différentes ont été essayées afin d'éviter ce problème, historiquement parlant, les chercheurs ont premièrement considéré l'utilisation de la divergence de Jensen-Shannon mesurant la distance entre les deux distribution et la distribution moyenne. Dans notre cas nous avons fait le choix d'essayer un dernier réseau GAN utilisant la divergence de Wasserstein au lieu de la divergence KL ou JS.

Ce changement multiple de métrique s'explique par la volonté d'améliorer l'apprentissage des GANs, qui peut s'avérer complexe si les deux distributions sont séparées dans l'espace des unions. Dans le cas où les deux distributions ne sont pas jointes, la divergence n'est qu'une étrange valeur qui ne peut pas aider le générateur à s'améliorer. En utilisant la métrique KL, le seul cas où le GAN peut apprendre de manière satisfaisante est lorsque les deux distributions ont une intersection qui ne peut pas être ignorée.

Les divergences KL et JS ne peuvent pas diriger l'apprentissage dans la bonne direction. C'est une des raisons principales expliquant l'incapacité d'un générateur d'apprendre considéré que la mesure de distance n'est pas porteuse d'information: cela empêche le modèle d'obtenir un gradient continu. À l'inverse, la metrique de Wasserstein correspond à la quantité de travail à réaliser pour transposer la distribution vers une autre. Ce qui en résulte une valeur positive dont la forme est symétrique.

Cette divergence se caractérise par deux propriétés:

- La fonction est continue en chaque points.
- Le gradient de la fonction est défini en chaque points.

Dans les réseaux GAN habituels, on cherche à maximiser le score de classification: si l'image est générée artificiellement, le discriminateur est censé renvoyer 0, à l'inverse si l'image est une vraie, celui-ci est censé renvoyer 1. Dans le cadre de WGAN, le travail du discriminateur est réduit à un problème de regression.

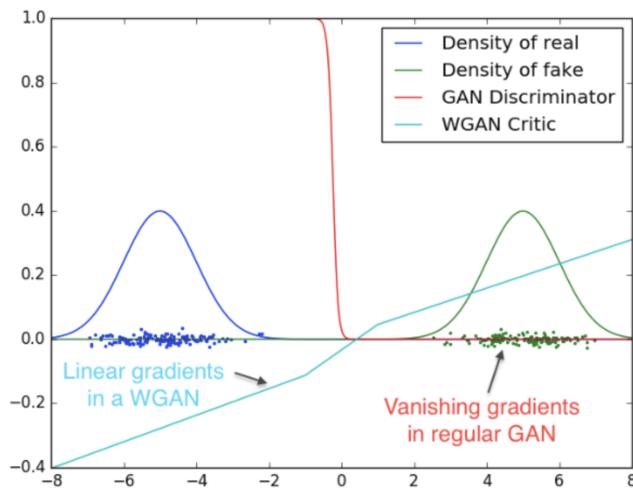


Figure 13: Convergence du gradient selon différentes structures

WGAN permet d'éviter le problème de disparition des gradients. Comme on peut le voir sur la figure ci-dessus, le gradient d'un réseau GAN classique a tendance à tomber à 0 et être saturé. La métrique de Wasserstein propose une loss porteuse d'information et le modèle est capable d'apprendre de manière progressive.

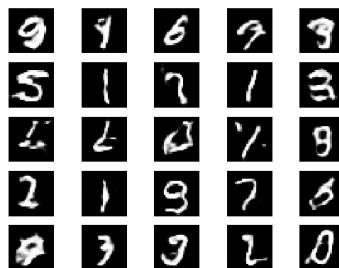


Figure 14: Images générées par notre implémentation d'Info-GAN: [Link](#).

Sans utiliser le principe de data augmentation, l'implementation de WGAN nous aura permis de faire monter l'accuracy à **92.84%**. De la même manière, nous avons réessayé cette méthode, tout en réutilisant le principe de data augmentation, obtenant alors une accuracy de **95.12%**.

**Code source.**

#### 2.2.5 Notre méthode d'apprentissage semi-supervisée avec DCGANs

Nous nous trouvions dans un cadre de problème qui se prêtait bien à l'apprentissage semi-supervisé: nous disposions simultanément d'images labélisées et non labélisées. Comme présenté lors de l'implémentation de DCGAN, il n'est pas nécessaire de disposer d'images labélisées pour l'apprentissage.

Notre méthode consiste à combiner un DCGAN avec un classifieur.

**Code source.**

Le principe consiste à exploiter toutes les images: les données non labélisées auront été utilisées afin de dissocier les vraies images des fausses, tandis que les données labélisées ont été utilisées pour optimiser la performance de la classification. En réalité, une partie de l'apprentissage consiste tout simplement à faire de l'apprentissage supervisé avec nos 100 exemples tandis que la seconde partie reste l'apprentissage adversarial classique entre le générateur et le discriminateur. Ces deux apprentissages se faisant de manière simultanée à chaque itération.

Sans utiliser le principe de data augmentation, notre implementation de DCGAN semi-supervisé nous aura permis d'obtenir une accuracy à **95.29%**.

### 3 Étude comparative des résultats obtenus

Après l'implémentation de notre méthode baseline, de nos trois modèles tirés de trois papiers sur les GANs ainsi notre dernière méthode expérimentale, nous avons souhaité situer nos résultats les uns par rapport aux autres mais également les comparer aux résultats de certaines méthodes classiques de machine learning sur le jeu MNIST, qui elles utilisent la totalité des labels du jeu de données soit 60 000 labels et non les 100 labels auxquels nous nous sommes restreint.

Method	Number of labels	Accuracy without augmentation
Info-GAN	100	87,92%
Baseline	100	90,12%
linear classifier (1-layer NN)	60000	91,60%
WGAN	100	92,84%
DC-GAN	100	94,16%
K-nearest-neighbors, Euclidean (L2)	60000	95,00%
Our DC-GAN Semi supervisé	100	95,29%
2-layer NN, 300 hidden units	60000	95,30%
NLP	60000	98,00%
ConvNets	60000	99,70%
RMDL	60000	99,82%

Figure 15: Tableau comparatif des méthodes utilisées

Nous remarquons qu'avec seulement 100 labels, ces méthodes sont équivalentes voir sensiblement meilleures en terme d'accuracy que certaines des méthodes proposées par Yann LeCun et al. au début des années 2000. Ainsi, un classifieur linéaire simple, un neural network à deux couches ou encore un KNN entraînés sur les 60000 labels obtiennent des résultats équivalents à ceux de nos réseaux.

Cela souligne encore une fois la capacité de ces réseaux mais également du deep-learning comparé à ces autres méthodes: en une vingtaine d'années, les chercheurs ont réussi à rattraper voir dépasser ces résultats avec moins de **5%** d'erreur en utilisant seulement **0.16%** des labels utilisés il y'a 20 ans.

## 4 Pour aller plus loin: meilleur résultat

Dernière étape de notre course à l'accuracy, nous avons souhaité augmenter celle-ci le plus possible en réutilisant le modèle entraîné sur les 100 labels pour lequel nous avions le meilleur résultat soit notre méthode du DC-GAN semi-supervisé avec **95.29%** d'accuracy sans data augmentation.

### Code source.

nous avons envoyé le restant des images du jeu de données soit 59900 images pour en prédire les labels avec un probabilité de **95.29%**. Sur 59900 images, nous estimons labéliser incorrectement environ 2820 images. Pour éviter d'envoyer trop d'images mal classifiées, nous avons décidé de ne garder que les images dont la probabilité de prédiction de la sortie softmax était supérieure à 70%. Ces images disposant à présent d'un label estimé, nous avons ré-entraîné ce même modèle avec cette fois la totalité des images et leur nouveaux labels afin de voir si envoyer plus d'images

labélisées mais avec un petit taux d'erreur dans ces labels permettait d'augmenter l'accuracy.

Au final, avec cette méthode nous arrivons à augmenter l'accuracy jusqu'à atteindre une valeur de **98.74%**.

Une fois ce résultat obtenu, nous avons cherché à explorer là où le modèle généré se trompait, grâce à la matrice de confusion et du score par label.

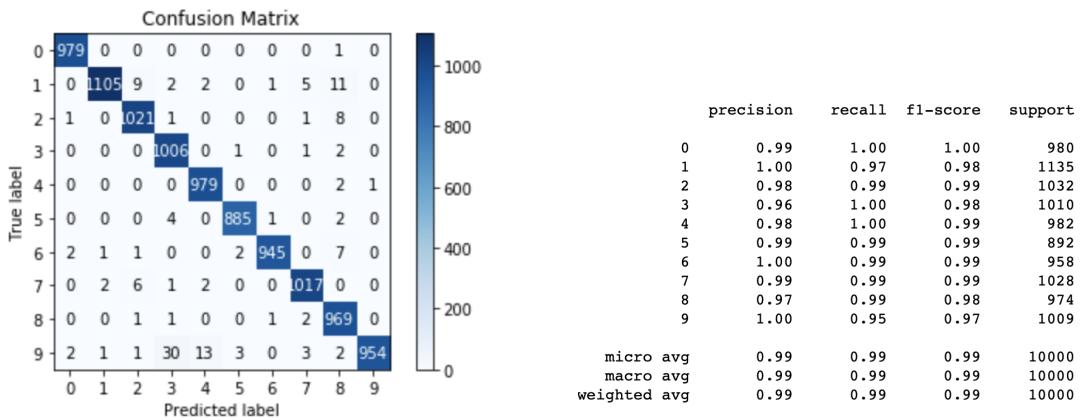


Figure 16: Matrice de confusion et accuracy par classe

On remarque que sur certains chiffres, le modèle peine en moyenne à les classifier correctement contrairement à d'autres pour lesquels la classification ne semble pas poser de problème. Pour pallier à ce problème, nous avons récupéré ce modèle à **98.74%** en modifiant la fonction de coût pour pénaliser légèrement plus le réseau lorsqu'il se trompait sur les chiffres dont la détection était en moyenne moins bonne que pour les autres. Cette opération nous a permis d'augmenter notre accuracy de **0.26%** et ainsi atteindre la barre symbolique des **99%** d'accuracy avec seulement 100 labels!

Method	Number of labels	Accuracy without augmentation
Info-GAN	100	87,92%
Baseline	100	90,12%
WGAN	100	92,84%
DC-GAN	100	94,16%
Our DC-GAN Semi supervised	100	95,29%
M1+M2	100	96,67%
Ladder Network	100	98,94%
Our DC-GAN Semi supervised + Train on predicted	100	99,00%
Improved-GAN	100	99,07%

Figure 17: Tableau comparatif des méthodes utilisées

Nous avons souhaité situer ce résultat dans l'état de l'art actuel pour le jeu de données MNIST avec cette fois seulement 100 labels. Ainsi, nous avons réussi à approcher l'état de l'art pour ce problème avec notre modèle de DC-GAN semi-supervisé.

## 5 Conclusion

Lors de ce projet, nous avons eu l'occasion de tester et comparer les retours de plusieurs des architectures classiques de deep-learning et plus précisément de manipuler plusieurs architectures de- réseaux antagonistes génératifs, plus communément appelés GAN. Nous avons également eu la possibilité de les utiliser dans des cas d'utilisation concrets sur un des jeux de données classique du machine learning.

Comme prévu, les résultats obtenus grâce à l'utilisation de ces réseaux et la réutilisation des différents discriminateurs entraînés sur les 100 labels sélectionnés sont bien meilleurs que les résultats obtenus via l'utilisation d'un simple réseau convolutionnaire avec ces mêmes labels. Ce qui en fait une méthode non-négligeable et intéressante à mettre en place lors de nos prochains projets ou le manque de données labélisées peut-être un problème.

L'utilisation de méthodes de traitement d'images afin de générer plus d'images labélisées lors de la seconde étape de nos entraînements aura également été prospère, nous permettant à chaque fois d'améliorer le modèle précédemment récupéré. Ainsi la data augmentation aura également été un acteur moteur de notre périple vers une accuracy toujours plus élevée.

Pour aller plus loin, nous avons souhaité réutiliser ces méthodes et ce que nous avons découvert durant ce projet dans un cadre industriel lors de nos projets en entreprise. Ainsi, nous avons eu l'occasion d'utiliser des GAN pour générer des visages au sein d'une mission de reconnaissance faciale au sein d'Idemia: afin d'améliorer la détection de faux-visages sur des caméras. Ceci nous permettant de situer la capacité de ce type de réseaux sur des projets industriels, ce qui au final est tout aussi important, voir plus que des bancs de tests sur un jeu de données utilisé à des fins de comparaison de l'état de l'art de toutes les méthodes de Machine Learning actuelles.

En conclusion, ce projet nous aura à la fois permis de comprendre plus profondément pourquoi le monde du Deep-Learning et plus précisément des GAN est en plein essor, mais également de répondre à nos missions en entreprise de manière plus adaptée.

## References

- [1] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio *Understanding the exploding gradient problem.*
- [2] Y. Bengio, P. Simard, P. Frasconi *learning long term dependencies with gradient descent is difficult.*
- [3] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio *On the difficulty of training recurrent neural networks.*
- [4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton *ImageNet Classification with Deep Convolutional Neural Networks.*
- [5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza *Generative Adversarial Nets.*
- [6] Alec Radford, Luke Metz, Soumith Chintala *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.*
- [7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel *InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets.*
- [8] Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen *Progressive Growing of GANs for Improved Quality, Stability, and Variation.*
- [9] Andrew Brock, Jeff Donahue, Karen Simonyan *Large Scale GAN Training for High Fidelity Natural Image Synthesis.*
- [10] Tero Karras, Samuli Laine, Timo Aila *A Style-Based Generator Architecture for Generative Adversarial Networks.*
- [11] Martin Arjovsky, Soumith Chintala, Léon Bottou *Wasserstein GAN.*
- [12] Tero Karras, Timo Aila Samuli, Laine Jaakko Lehtinen *Progressive growing of GANs for improved quality, stability and variation.*
- [13] Antti Rasmus, Harri Valpola, Mikko Honkala *Semi-Supervised Learning with Ladder Networks.*
- [14] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung *Improved Techniques for Training GANs.*
- [15] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling *Semi-supervised Learning with Deep Generative Models.*

## 6 Annexe

**Lien Github du projet:** <https://github.com/yannistannier/gans-mnist>