Université Paris Descartes

Projet de fin de module

Prédiction des données MNIST avec 100 labels

Team:
LIEPCHITZ Laura
ABIDAR Bouchra

Teacher:
HANCZAR Baise

2019-2020

Contents

1

Introduction

L'utilisation du Deep Learning pour la reconnaissance d'images est de plus en plus fréquente. Toutefois, la majorité des données produites ne sont pas étiquetées. Pour autant elles contiennent de grandes quantités d'informations. Dans ce cadre il est intéressant de développer des approches capables de s'entraîner sur un ensemble de données réduit. D'autant plus que la labelisation à main des données est très coûteuse.
Dans ce projet, l'objectif est de créer plusieurs architecture permettant de prédire le chiffre sur une image avec un nombre limité d'étiquettes puis de les comparer entre eux.
Ce rapport s'organise selon quatre parties. Nous commencerons tout d'abord par présenter les modèles retenus pour répondre à cette problématique. Nous présenterons ensuite le jeu de données MNIST sur lequel se base cette expérimentation. Nous continuerons avec la méthodologie adoptée. Enfin nous détaillerons et comparerons les résultats obtenus par les différents modèles.

2

Les modèles retenus

2.1 CNN

Un réseau de convolution (CNN) est un type de réseau neuronal très utilisé pour la classification d'images ou la reconnaissance visuelle. Le but principal d'un réseau de convolution est d'extraire des caractéristiques d'une image donnée.
Le principe d'un réseau de convolution, est l'enchainement d'étapes. Chaque étape consiste à appliquer des filtres suivi d'une phase de Pooling sur le résultat de chaque filtre (Figure 1).

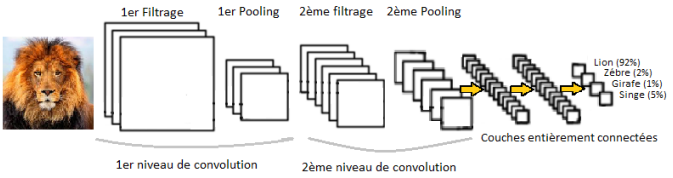

Figure 1: Réseau de convolution

Une fois les étapes de convolutions effectuées, les patterns obtenus en sortie sont injectés comme données d'entrée dans un réseau de neurones classique. Le but de celui-ci est de classifier les données en déduisant une probabilité sur les différents résultats possibles.

Les réseaux neuronaux convolutifs ont de nombreuses applications dans la reconnaissance d'images, de vidéos ou le traitement du langage naturel.

2.2 GAN

En 2014, Ian Goodfellow et son équipe ont présenté les Generative Adversarial Networks (GANs) [1]. Ils ont prouvé que les GANs étaient capables de synthétiser de nouvelles images basées sur les images d'entrée servant pour l'entrainement du réseau (Figure 2.

Figure 2: Visualization of samples from the model[1]

Les GANs constituent un exemple de réseau utilisant l'apprentissage non supervisé pour entraîner deux modèles en parallèle: un modèle *Génératif* et modèle *Discriminant*.

Le modèle génératif des GANs produit des données qui capturent les caractéristiques de l'ensemble de données d'entraînement en utilisant une couche appelée couche de déconvolution. Le générateur apprend les motifs des images pour discerner le contenu d'une image. L'input du générateur est un vecteur de nombres aléatoires.

Le générateur apprend grâce aux commentaires qu'il reçoit du discriminateur. L'objectif de ce dernier est de déterminer si un exemple particulier est réel (provenant de l'ensemble de données d'apprentissage) ou généré (créé par le générateur) (Figure 3.

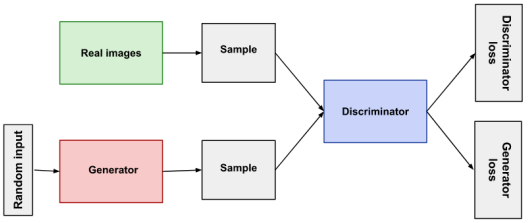

Figure 3: L'apprentissage generator-discriminator

Au cours de l'entraînement, la rétropagation est utilisée sur les deux réseaux afin d'ajuster les paramètres et ainsi générer des images de sortie plus réalistes. L'objectif est de mettre à jour

les paramètres du réseau génératif jusqu'à ce que le réseau discriminant ne parvienne plus à discerner les données d'apprentissage de celles générées par le générateur.

2.2.1 SGAN

Le GAN semi-supervisé (SGAN) est un réseau GAN dont le discriminant est un classifier multi-classes. Au lieu de distinguer seulement deux classes («réelle» et «générée»), le discriminant apprend à distinguer entre N + 1 classes, où N est le nombre de classes dans l'ensemble de données d'apprentissage, et la classe supplémentaire servant à repérer les images générées par le générateur. Dans SGAN, le discriminateur n'est plus un classifier binaire mais un multi-classes. DANS SGAN, le génératif prend un vecteur aléatoire en entrée, il produit un faux. Le discriminant reçoit trois types d'entrées de données: de fausses données du générateur, de vrais exemples non étiquetés x et de vrais exemples étiquetés (x, y) où y est l'étiquette correspondant à l'exemple donné. Le discriminateur produit une classification. Cette dernière a pour objectif de distinguer les faux exemples des vrais et, pour les vrais exemples, d'identifier la bonne classe.

2.2.2 DCGAN

Une variation du GAN est le réseau contradictoire génératif convolutif profond ou DCGAN. Il a été décrit pour la première fois par Radford [4]. La figure 4 est un exemple d'image de chambres générées à partir d'un DCGAN.

Figure 4: Images de chambres générées à partir d'un réseau DCGAN[3]

Dans DCGAN, le discriminant fonctionne comme un CNN classique et le génératif utilise des couches dé-convolutionnelles(figure 5).

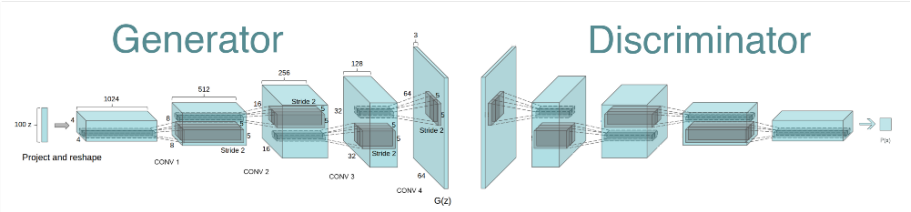

Figure 5: Architecture DCGAN[5]

2.3 Auto-encodeur

Un auto-encodeur consiste à apprendre à reconstruire les données fournis en entrée en passant par une ou plusieurs couches cachées entièrement connectées, de taille réduite par rapport à l'entrée, mais contenant assez d'information. La figure 6 illustre cette stratégie.

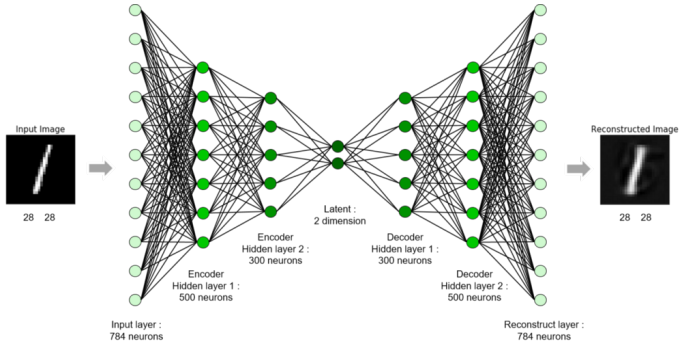

Figure 6: Stratégie d'un auto-encodeur

Certains auto-encodeurs sont capables de générer de nouvelles données qui ressemblent énormément aux données originales. C'est un modèle (*génératif*).

3

Données MNIST

La base de données MNIST est une base de données d'images de chiffres manuscrits, constituée par plusieurs chercheurs et étudiants du United States Census Bureau [6]. Elle regroupe une base d'entrainement de 60000 images et une base de test de 10000 images. Ces images sont en noir et blanc, normalisées centrées et de dimension 28x28. Ce jeu de données est facilement accessible depuis la librairie Keras. La figure 7 est un exemple des images présentes dans la base de données MNIST.

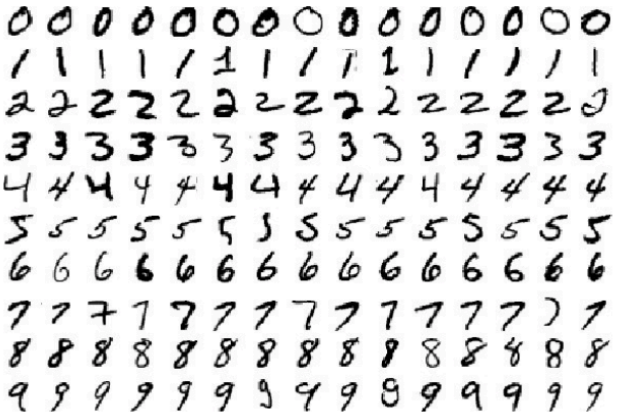

Figure 7: du jeu de données MNIST

4

Méthodologie

4.1 Baseline

Nous avons adopté une architecture auto_encodeur basique avec une seule couche cachée comme référence baseline (cf. le code sur github).

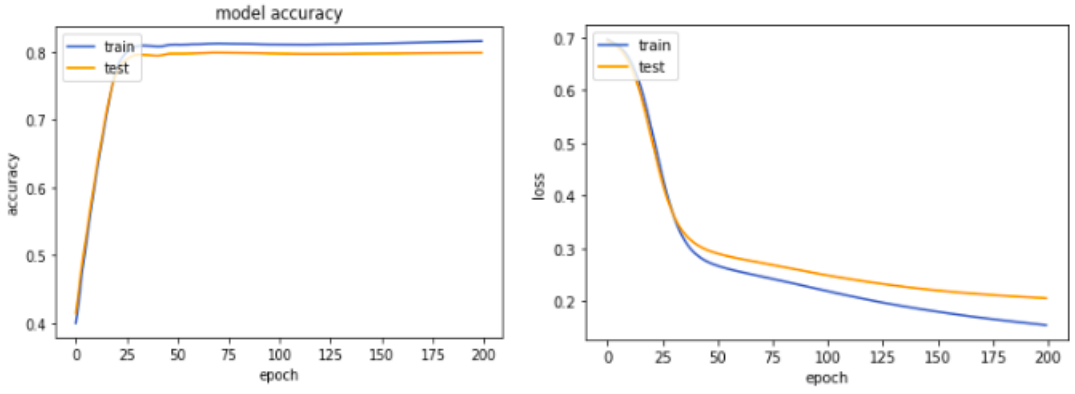

Figure 8: Visualisation des métriques de performances- Baseline

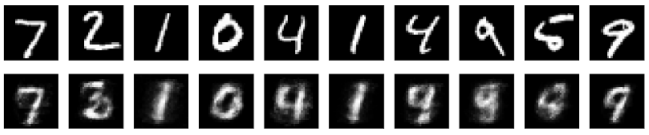

Figure 9: *Un comparaison entre les données réelles(au dessus) et celles généré par l'auto-encodeur(AU-DESSOUS)*

L'auto-encoder a été entraîné sur la totalité de l'ensemble d'apprentissage (60000 images) pour avoir une idée des performances vers lesquelles tendre une fois que les données d'entraînement seraient limitées à 100 labels.

4.2 Entraînement des modèles

4.2.1 CNN

Une des approches expérimentées dans ce projet est le *CNN*. L'entraînement a été basé sur 100 labels choisis aléatoirement. La figure 10 propose une visualisation des métriques pendant l'entraînement de modèle CNN avec 100 labels.

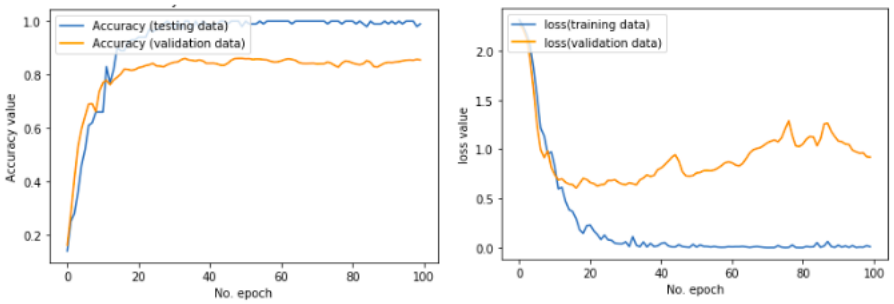

Figure 10: *Visualisation des métriques de performances - CNN avec 100 labels sans data augmentation*

Pour améliorer les performances de du réseau entrainé sur seulement 100 labels, la technique de *data augmentation* a été mise en oeuvre.
La Data augmentation est utilisée pour enrichir les images d'apprentissage en implémentant plusieurs stratégies pour augmenter la taille de l'ensemble de données en créant des variations de ces images notamment :

- Augmentation la taille des images
- Utilisation des filtres sur les images
- Application des rotation sur les images
- Changer la luminosité des images
- Augmenter aléatoire du zoom dans les images

Dans le cadre de ce projet, la data augmentation permet d'enrichir la base d'apprentissage en se basant sur 100 labels. La figure(11) présente l'évolution de l'accuracy ainsi que la convergence du loss durant l'entraînement de CNN avec data augmentation.

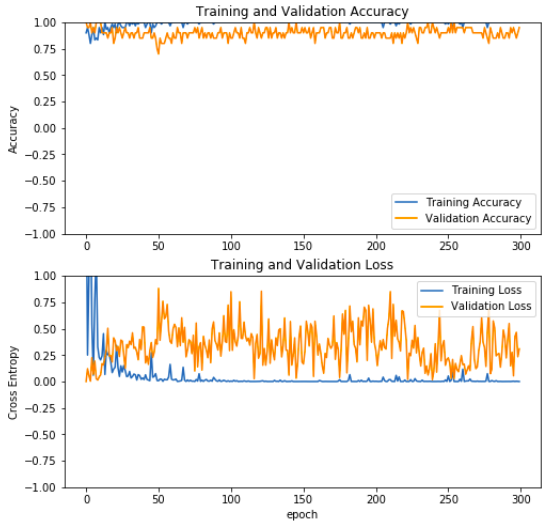

Figure 11: Visualisation des performances métriques

4.2.2 SGAN

L'objectif d'utiliser les réseaux SGAN est de tester une méthode de classification semi-supervisé en utilisant 100 labels. Dans ce type des réseaux, le générateur prend un vecteur de nombres aléatoires et produit des fausses images qui sera transmis au discriminateur dans le but de s'entraîner à classifier les images réelles et celles générées par le générateur. Pour le discriminateur, nous l'avons implémenté selon trois étapes (cf le code):

1. 1 ère étape : Discriminateur basique
2. 2 ème étape : Discriminateur supervisé : nous utilisons le réseau discriminateur basique pour créé la partie supervisée du discriminateur avec la fonction softmax qui produit un vecteur de probabilité.
3. 3 éme étape : Discriminateur non supervisé : dans cette partie nous avons implémenté la partie non supervisée du modèle discriminateur avec une fonction sigmoid pour une classification binaire

4.2.3 DCGAN

Aprés l'entrainement de DCGAN (cf. le code), nous récupérons le discriminateur et nous remplaçons la dernière couche par une couche Dense de 10 sorties qui correspondent aux 10 classes du jeu de données MNIST (figure 12).

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1_input (InputLayer)  (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        320
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 7, 7, 64)          256
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 4, 4, 128)         512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
logits (Dense)               (None, 10)                20490
=================================================================
Total params: 113,930
Trainable params: 113,546
Non-trainable params: 384
```

Figure 12: Modèle créé en se basant sur le discriminator de DCGAN

Les figures 13, 14, 15 et 16 sont des exemples des images générées par le générateur en fonction des epochs.

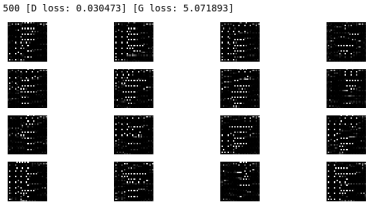

Figure 13: Image générée par DCGAN epoch :500

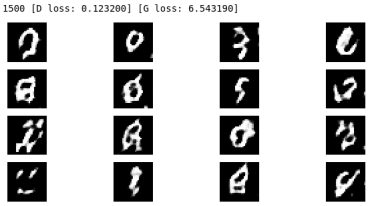

Figure 14: Image générée par DCGAN epoch :1500

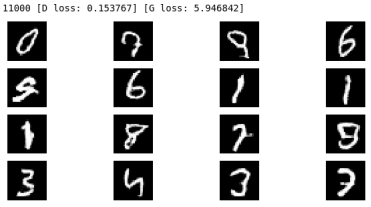

Figure 15: Image générée par DCGAN epoch :11000

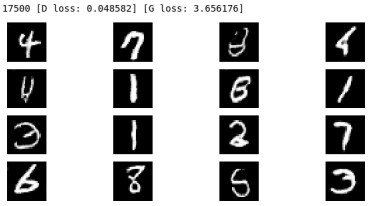

Figure 16: Image générée par DCGAN epoch :17500

5

Étude comparative

Cette partie a pour objectif de présenter les performances de l'ensemble des modèles étudiés dans ce projet. Ces performances sont regroupées dans le tableau 1. L'ensemble des modèles a été entraîné sur les 100 labels imposés par le sujet. Certains ont toutefois été également entraînés sur la totalité de la base d'entraînement pour avoir une idée des performances vers lesquelles tendre une fois que les données d'entraînement seraient limitées à 100 labels.

Table 1: Tableau comparatif des méthodes utilisées

Algorithmes	**Taille (training set)**	**Accuracy(%)**
Baseline	100 labels	79
	60000 labels	81
CNN	100 labels	83
	100 labels (Data augmentation)	85
	60000 labels	99
SGAN	100 labels	91
DCGAN	100 labels	87

L'ensemble des modèles ont obtenu des performances supérieures à celle de la méthode baseline. SGAN est le modèle le plus performant avec une accuracy de **91%**. Ce score est supérieur à celui obtenu avec l'auto-encodeur avec un entraînement sur 60000 images.

Sur 100 labels, CNN obtient une accuracy de **83%** avec un entraînement basé sur la data augmentation l'accuracy atteint les **85%**. Cela confirme l'utilité d'utiliser ce genre de technique pour améliorer les performances des modèles surtout dans un cas ou l'ensemble d'apprentissage est réduit.

6

Conclusion

Nous avons eu l'occasion d'expérimenter plusieurs approches pour prédire le chiffre sur une image en se basant sur une sous-partie des données étiquetées (100 exemples etiquetés). Cela est plus adaptée aux problématiques réelles. En effet, de manière générale les données ne sont pas étiquetées et il est très coûteux de le faire à la main. De plus, dans le but d'économiser le temps de calcul il est alors intéressant d'avoir un ensemble d'entrainement réduit. L'objectif est alors d'allier cela à la conservation de performances satisfaisantes.

Durant ce projet, nous avons mis en pratique plusieurs modèles et techniques vus dans le cours de deep learning. Le but étant de se baser uniquement sur une sous partie des données d'entraînement labelisées. Nous avons proposé d'implémenter des approches utilisées dans la littérature pour répondre à ce besoin, notamment :

- utilisation de la data augmentation
- utilisation des approches semi-supervisée comme présenté dans le *SGAN*
- realisation d'un pré-entraînement sur la totalité de la base d'apprentissage d'un modèle GAN et récupéreration du meilleur discriminateur.

Le modèle le plus performant est le SGAN qui obtient une accuracy de **91%**, ce qui est un score satisfaisant sur uniquement 100 labels.

7

Annexe - Code

L'ensemble du code se trouve sur ce dans:
https://github.com/liepstik/deep_learning_MNIST_with_100_labels

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio *Generative Adversarial Nets*.2014

[2] Jakub Langr, Vladimir Bok *GANs in Action: Deep Learning with Generative Adversarial Networks*.2019

[3] https://github.com/Newmu/dcgan_code

[4] Alec Radford Luke Metz, Soumith Chintala *UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS*.2016

[5] *https://mc.ai/a-friendly-introduction-to-dcgan/*

[6] LeCun, Yann; Cortez, Corinna; Burges, Christopher C.J.*"The MNIST Handwritten Digit Database"*. Yann LeCun's Website yann.lecun.com.