Dauphine | PSL
UNIVERSITÉ PARIS

Rapport

One shot learning

Élèves :
Salma Elghourbal
Kaoutar BOULIF
Mohamed Zineddine CHEDADI

Enseignant :
Benjamin NEGREVERGNE

18 novembre 2020

Table des matières

Introduction

Dans un problème classique d'utilisation de réseaux de neurones en classification d'images, il est généralement nécessaire d'avoir un grand nombre d'exemples par classe dans le set d'entraînement. Par ailleurs, le nombre de classes est préfixé au moment de l'apprentissage ; le réseau de neurones ne reconnaît donc pas une nouvelle classe qui n'était pas dans l'ensemble d'entraînement. Avec une architecture de réseau de neurones classique, il faut rajouter beacoup d'exemples pour une classe supplémentaire si nous souhaitons que le réseau la reconnaisse, de modifier la sortie du modèle et de le réentraîner sur le nouveau dataset augmenté. Le processus est ainsi assez coûteux en termes de collecte de données et de temps d'apprentissage. En revanche, en "One shot classification", nous avons uniquement besoin d'un seul exemple d'entraînement par classe, d'où le "One shot". Le 'One shot learning' n'a pas pour objectif de retourner en output une classe pour une image donnée, **il apprend, plutôt une fonction de similarité (ou de dissimilarité) qui prend en entrée deux images et exprime leur degré de similarité**. Comment peut-on alors implémenter une solution qui effectue du "One shot learning" ?

Dans ce document, nous présentons différentes approches en "One shot learning", ensuite nous présentons les expériences réalisées sur deux Datasets MNIST et CIFAR10, nous commentons ces résultats et enfin exposons quelques limitations.

1 Approches d'apprentissage du one shot learning

Les réseaux neuronaux siamois (Siamese neural networks) sont une classe de réseaux de neurones contenant de multiples instances du même modèle qui partagent la même architecture et les mêmes poids. Les réseaux Siamese représentent un outil très puissant capable d'apprendre avec un volume de données limité ou incomplet tel est le cas dans les tâches d'apprentissage Zéro / One shot.

L'idée générale est de transformer les images en entrée du réseau Siamese dans un espace latent de dimension plus petite et d'apprendre "des embeddings ou des encodings" des images dans cet espace tels que :

- Si deux images appartiennent à la même classe, leurs "encodings" doivent être géométriquement proches, par exemple en termes de distance L2
- Si deux images sont de classes différentes, leurs "encodings" doivent être géométriquement éloignés, en termes de distance L2 par exemple.

Ainsi, si nous disposons d'un réseau Siamese qui permet de séparer 10 classes par exemple, et nous souhaitons qu'il reconnaisse une image test appartenant à une classe supplémentaire, il suffit d'ajouter une seule image appartenant à cette nouvelle classe dans la base de données et de calculer son encoding. Le réseau compare ensuite l'encoding de l'image test à classifier avec les encodings des images de la base de données et lui assigne la classe de l'image dont l'encoding est le plus proche. Si le réseau est bien entraîné, il est probable qu'il reconnaisse bien la classe de l'image test même si nous ne disposons qu'une seule image similaire dans nos données. Le Siamese Network permet ainsi d'effectuer la tâche de One shot learning.

Dans les trois sous-sections qui suivent, nous présentons trois manières de construire et d'entraîner des réseaux de neurones Siamese.

1.1 Contrastive loss

Le réseau Siamese est constitué de deux branches de réseaux de neurones similaires en poids et en entrée et sortie, chaque branche calcule un encoding de son entrée , nous rajoutons une couche "Distance" à la sortie des deux encodings qui calcule la distance euclidienne entre les deux. Nous entraînons les paramètres du modèle résultant pour qu'il optimise la fonction objective suivante , appelée **Contrastive Loss** :

$$L(a, b, f) = y_{ab} \cdot \frac{1}{2} \cdot d_f(a, b)^2 + (1 - y_{ab}) \cdot \frac{1}{2} \cdot max(0, \alpha - d_f(a, b))^2$$

a et b sont deux images, $d_f(a, b) = ||f(a), f(b)||_2$ avec f le réseau de neurones de base renvoyant l'encoding d'une image, α est un seuil et y_{ab} est le label à approcher qui vaut 1 si a et b sont de même classe, 0 sinon.

1.2 Binary cross entropy loss

Une variante du réseau précédent est présentée dans la figure suivante :

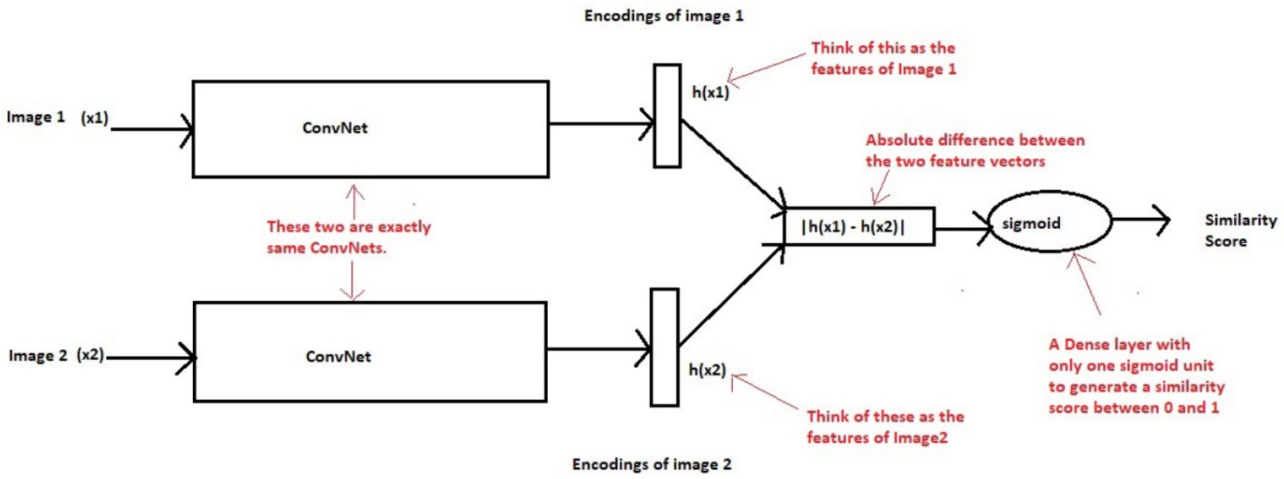

FIGURE 1 – Architecture d'un Siamese network pour la Binary Cross entropy

La différence ici est que nous rajoutons à la suite des encodings une distance L1 puis une couche à une seule sortie avec une fonction d'activation "Sigmoid" afin de retourner une probabilité d'appartenance à la même classe (mesure de similarité) qui est proche de 1 si les deux images sont similaires, proche de 0 sinon. Le modèle complet s'entraîne avec comme loss la **Binary crossentropy** usuellement employée dans une classification binaire [1]

1.3 Triplet loss

L'approche "Triplet loss" est décrite dans le papier de FaceNet [2]. Nous expliquons brièvement le principe, l'idée est de considérer :

— Une image de départ A, appelée l'Ancre (Anchor).

— Une image de la même classe de l'Ancre appelée l'image positive P (Positive).
— Une image de classe différente appelée l'image négative (Negative) N. (Voir la figure 2)

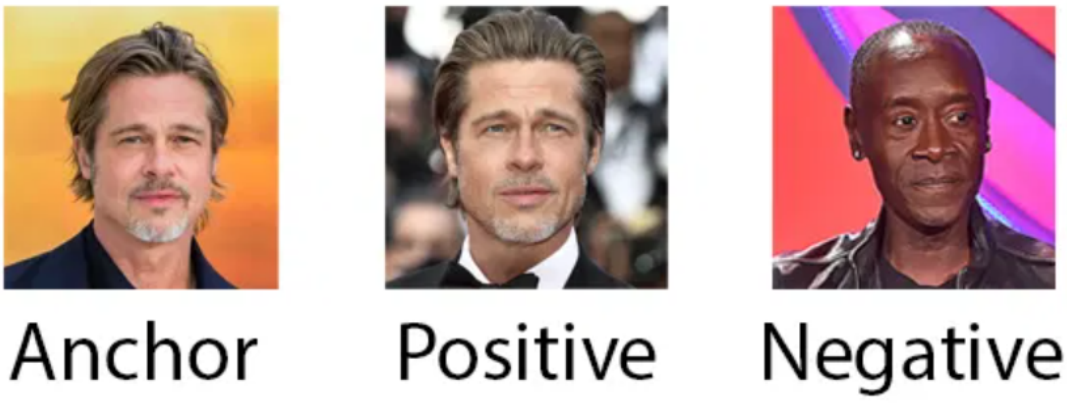

FIGURE 2 – Exemple de triplet

L'objectif est d'apprendre des encodings f tels que :

$$d_f(A, P) - d_f(A, N) + \alpha < 0$$

Le "margin" α fait en sorte que le modèle de base d'un Siamese Network à trois entrées n'apprenne pas une solution triviale à savoir $f = 0$. La "triplet loss" qui constitue la fonction à optimiser dans notre cas est définie par :

$$L(A, P, N) = max(d_f(A, P) - d_f(A, N) + \alpha, 0)$$

Les poids du réseau sont appris de manière à minimiser la triplet loss.

Impact du choix des triplets dans la base d'apprentissage :

Selon la définition de la fonction de coût, nous distinguons trois types de triplets :

— Easy triplets : les triplets dont la fonction de coût est égale à 0 vu que :

$$d_f(A, P) + \alpha < d_f(A, N)$$

— Hard triplets : les triplets dont l'exemple négatif est plus proche à l'ancre que l'exemple positif :

$$d_f(A, N) < d_f(A, P)$$

— Semi-hard triplets : les triplets pour lesquels :

$$d_f(A, P) < d_f(A, N) < d_f(A, P) + \alpha$$

Ainsi, nous avons plusieurs stratégies de choix des triplets et cela risque d'impacter la convergence de notre réseau ainsi que la vitesse d'entraînement. En effet, Il est juste de dire

que si notre fonction de perte est très petite pour ces triplets "easy", alors notre modèle ne va pas apprendre grand chose. En revanche, les triplets "hard" généreront des pertes élevées et auront un impact important sur les paramètres de notre réseau. Cela donnera trop de poids à toute donnée mal étiquetée. Nous devons donc choisir une stratégie qui mélange les triplets "easy", "hard" et peut-être "semi-hard".

1.4 Evaluation d'un modèle One shot learning

Nous avons présenté trois approches en One shot learning. La question qui se pose est : Comment comparer ces trois modèles ?

Il existe une technique de comparaison permettant de calculer une "accuracy" pour chacun des modèles pour pouvoir comparer leurs performances. Cette technique est appelée le **n way One shot learning**. Nous pouvons procéder de la manière suivante :

1. Choisir un n désignant le nombre de classes.
2. Sélectionner n images représentatives de ces n classes dans la base de données connue.
3. Choisir une image test dont le réseau ne connaît pas la classe a priori.
4. Passer l'image test et les images connues à l'entrée du réseau de neurones de base du Siamese Network
5. Trouver l'image la plus proche en distance L2 de l'encoding de l'image test
6. Vérifier si cette prédiction est correcte : l'accuracy est égale à 1 si cette prédiction est correcte, 0 sinon
7. Enfin, répéter ce processus k fois et calculer une **accuracy moyenne** du modèle en n way one shot learning.

2 MNIST

2.1 Introduction

Dans cette section, nous présentons les différents expériences et résultats effectués sur le dataset MNIST.

Pour rappel, MNIST est une base de données de chiffres écrits à la main. La base MNIST regroupe 60000 images d'apprentissage et 10000 images de test. Chaque exemple est une image de (28*28) pixels qui représente un chiffre entre 0 et 9 (10 classes).

Dans tous les modèles implémentés dans cette partie, nous utilisons le même modèle de base CNN avec les mêms poids (branche CNN) préentraîné pour la tâche classification des images MNIST. Pour l'utiliser dans chacun des modèles Siamese, il suffit de changer la fonction d'activation de sa dernière couche de 'softmax' à 'relu'. C'est un modèle qui prend en input une image et renvoie son encoding qui est de taille 10

2.2 Préparation des données d'apprentissage et de test

2.2.1 Base de données d'apprentissage

Deux types de dataset d'apprentissage ont été générés :

— Type 1 : construire des paires d'images semblables (image,image-same) et des paires d'images différentes (image,image-diff), sachant qu'on n'utilise que des images de classe entre 0 et 4.
— Type 2 : dans ce cas, on génère les paires semblables et les paires différentes à partir d'un ensemble d'images appartenant aux classes 0 à 4 auquel on rajoute un seul exemple pour chacune des classes 5 à 9.

2.2.2 Base de données de test

La préparation des données de test se fait selon 3 différents types : easy, medium et hard :

— Easy : contient que des paires d'images appartenant aux classes entre 0 et 4.
— Medium : contient que des paires d'images pouvant appartenir à toutes les classes entre 0 et 9.
— Hard : contient que des paires d'images appartenant aux classes entre 5 et 9.

2.3 Performances des modèles sur les pairs d'images

L'utilisation d'un modèle CNN simple a abouti à des valeurs d'accuracy très faibles. C'est pourquoi, nous avons opté pour la technique du Transfer Learning, en utilisant un modèle CNN pré-entrainé sur l'intégralité de la dataset MNIST, en apportant une légère modification sur la dernière couche du réseau qui est celle de choisir la fonction d'activation "ReLU" au lieu de "Softmax". L'utilisation du transfer learning concerne les modèles dont la loss est soit "contrastive loss" soit "binary cross entropy loss". Le modèle préentraîné avait 98.7% d'accuracy sur le "trainset" et 98.6% sur le "test set".

Ci-dessous sont présentés les performances des modèles sur les différents type de test set en fonction de la base de données d'apprentissage utilisé :

	Train set	Easy test set	Medium test set	Hard test set
Type 1 train set	99%	99%	92%	81%
Type 2 train set	99%	97%	91%	91%

FIGURE 3 – Accuracy du modèle avec la contrastive loss pour MNIST database

	Train set	Easy test set	Medium test set	Hard test set
Type 1 train set	97%	96%	81%	70%
Type 2 train set	98%	95%	87%	80%

FIGURE 4 – Accuracy du modèle avec la binary cross entropy loss pour MNIST database

On peut constater donc pour le cas de la database MNIST que le fait d'introduire un exemple d'images pour les classes de 5 à 9 a amélioré les performances pour le cas du modèle avec la binary cross entropy ainsi que pour le modèle avec la contrastive loss.

2.4 Comparaison des performances des modèles à base de triplets en fonction des types de triplets sélectionnés pendant l'entraînement

Nous entraînons le modèle à base de triplets en utilisant trois stratégies : la première stratégie consiste à selectionner en majorité les "easy triplets" dans la base d'entraînement, la deuxième consiste à sélectionner en majorité des "hard triplets" et enfin la troisième utilise des triplets random (aléatoirement sélectionnés). La caractérisation de chaque type de triplets a été faite dans l'introduction. Le tableau ci-dessous résume les accuracy des trois modèles résultants :

	Easy test set	Medium test set	Hard test set
Easy train triplets	98%	98%	97%
Random train triplets	98%	98%	97%
Hard train triplets	99%	98%	97%

FIGURE 5 – Accuracy des modèle des triplets

Pour une valeur de margin $\alpha = 1$, les performances pour chaque type de test set (easy, medium et hard) sont les mêmes sur les différents types de triplets utilisés (easy, random and hard) selon la figure 5.

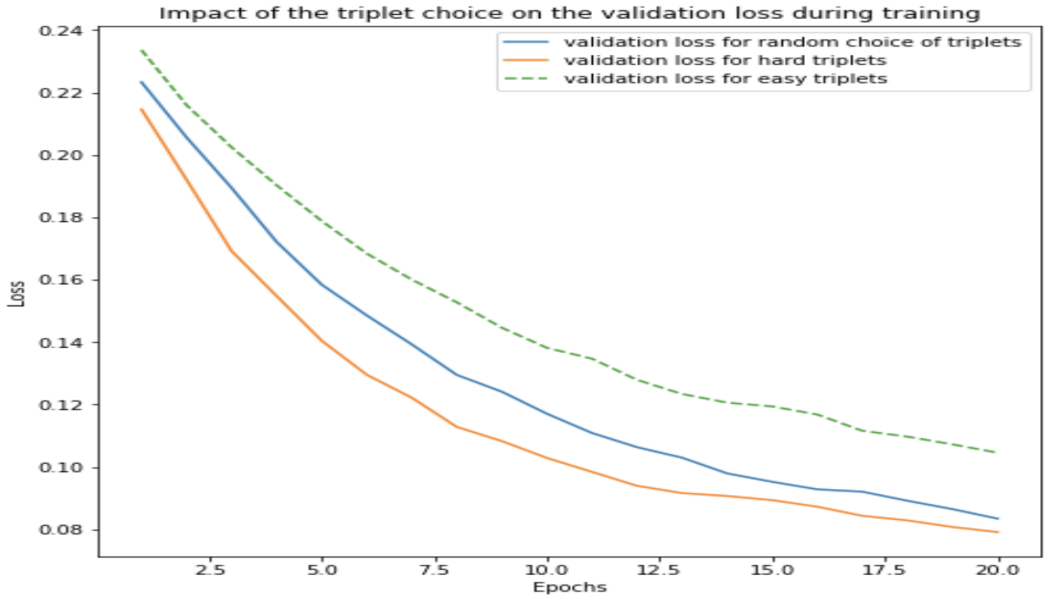

FIGURE 6 – Comparaison des validation loss des modèles des triplets

La figure 6 montre l'évolution de la fonction de coût sur le set de validation en fonction des epochs d'entraînement, et ce pour les trois modèles à base de triplet entraînés avec les trois stratégies de choix de triplets. Nous observons que la loss en validation dans le

cas des 'hard' triplets diminue plus rapidement que la loss en validation dans le cas d'un choix random des triplets. Enfin, La validation loss dans le cas des 'easy' triplets diminue le plus lentement vu que le modèle apprend moins que lorsque les triplets sont difficiles ou tirés aléatoirement.

2.5 Comparaison des modèles avec le Nway one shot learning

Une bonne méthode de juger les performances des modèles que nous avons utilisé est le N-way one shot learning (voir définition dans la partie I).

2.5.1 Comparaison entre les modèles des triplets

Le graphe ci-dessous illustre la comparaison du N-way one short learning accuracy en fonction du nombre de test n pour les différents trois types de génération des triplets d'images (anchor, positive, negative). Dans notre cas, nous avons choisi N = 10. A chaque itération, nous répétons le processus décrit dans la partie I 10 fois puis nous calculons une accuracy moyenne à chaque itération. Notons que la courbe est censée être discrète mais nous l'avons lissée pour plus de lisibilité.

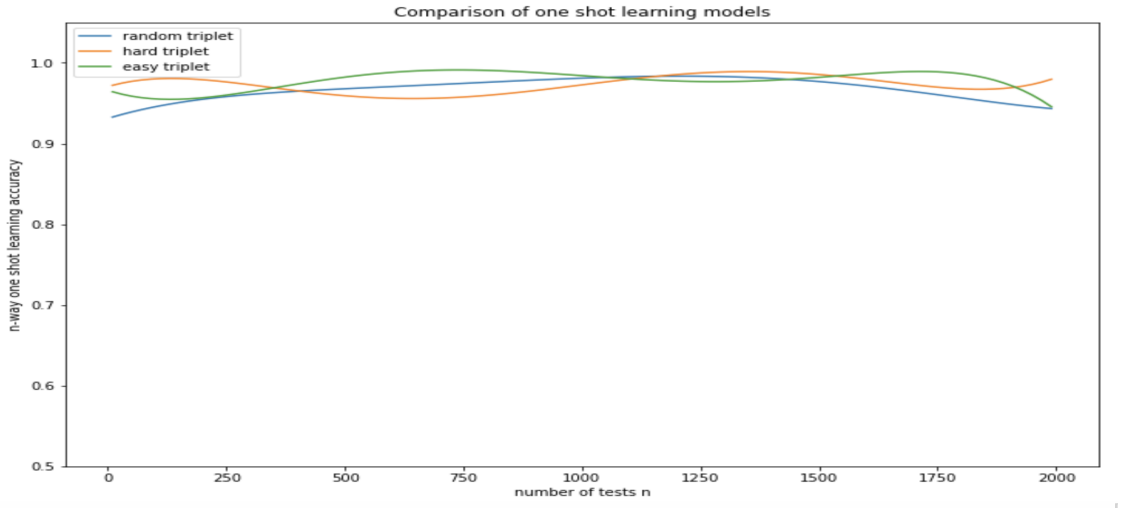

FIGURE 7 – Comparaison entre les modèles des triplets

En moyenne, les performances des trois types de triplets sont similaires. Sur le dataset MNIST, on peut dire que la répartition des triplets entre easy et hard triplets n'apporte pas vraiment un plus par rapport au choix aléatoire des triplets, au moins pour les valeurs de marge qu'on a testé. Il est tout à fait possible de commencer à voir des distinctions sur les comportements des différents triplets en choisissant des valeurs judicieuse de α. Pour des raisons liés au temps d'exécution, nous nous sommes limités à 3 valeurs de α qui sont 0.001, 1 et 1.5 et pour lesquels on a eu le même résultat.

2.5.2 Comparaison entre tous les modèles

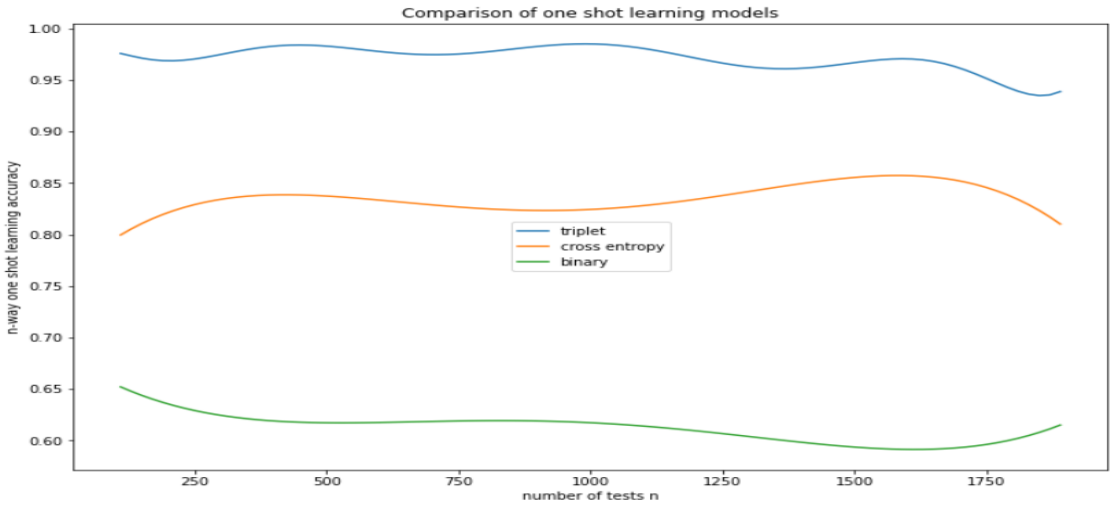

FIGURE 8 – Comparaison entre tous les modèles

Le graphe présenté ici permet de visualiser les performances des trois différents modèles utilisés dans notre étude (triplet, binary et cross entropy loss) en termes d'accuracy moyenne en 10-way One shot learning.

On peut voir que le modèle avec la triplet loss est largement plus performant que les deux autres modèles, avec une accuracy en moyenne égale à 95%. Il suit ensuite le modèle avec la contrastive loss avec une accuracy égale en moyenne à un peu plus de 75% et finalement le modèle le moins performant est celui avec la binary loss avec une moyenne de 70%.

Quelques illustrations

— Illustration d'un 10-way one shot learning :
Nous avons effectué trois exécutions du 10 way one shot learning en utilisant les modèles de base des trois approches implémentées. La figure 9 représente le résultat obtenu dans le cas du modèle entraîné à l'aide de la "Contrastive loss". Nous pouvons voir que l'image la plus proche de l'image test en terme de distance euclidienne des encodings est bien celle d'un exemple de la classe 4. Par conséquent, la prédiction de la classe de l'image de test est 4 ce qui est bien correct.

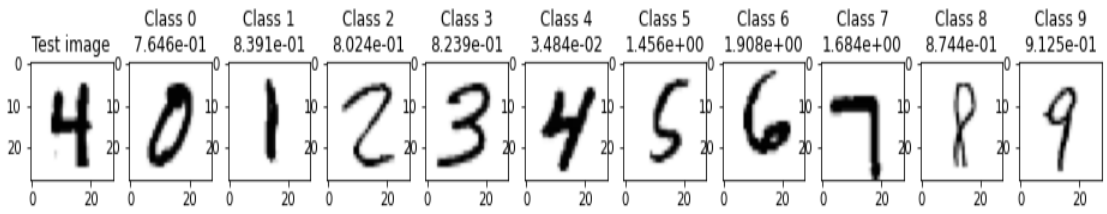

FIGURE 9 – Prédiction de la classe d'une image inconnue

— Illustration des représentations des exemples avant et après l'application d'un modèle à base de triplets :

Dans un premier temps, nous avons testé une méthode de réduction de dimension appelée "t-sne" sur les 'raw' pixels des images d'un échantillon de la base d'entraînement MNIST originale contenant les 60.000 images ainsi que sur les 'raw' pixels d'un échantillon de la base de test originale de taille 10.000 . Nous illustrons uniquement les représentations 2-D obtenues pour l'échantillon de la base d'entraînement dans la figure 10. Ensuite, nous avons calculé les 'encodings' des images de l'échantillon de la base d'entraînement ainsi que les encodings de l'échantillon de la base de test en utilisant le modèle de base entraîné du Siamese network qui utilise la Triplet loss. Dans la figure 11, nous traçons les représentations en 2D obtenues avec t-sne des encodings des images de l'échantillon d'entraînement.

Dans la figure 10, les exemples de la même classse sont généralement proches les uns des autres (classe 0, classe 8, classe 6) mais ces clusters ne sont pas parfaits et ils ne sont pas loins les uns des autres : les exemples de la classe 2 ne forment pas un seul cluster, les clusters 9 et 4 sont mélangés, etc..
Dans la figure 11, les encodings des exemples forment des clusters bien éloignés les uns des autres et resserrés sur eux-mêmes, ce qui montre qu'on a pu identifier les bons features permettant de ramener le problème de classification à un problème de mesure de distance.

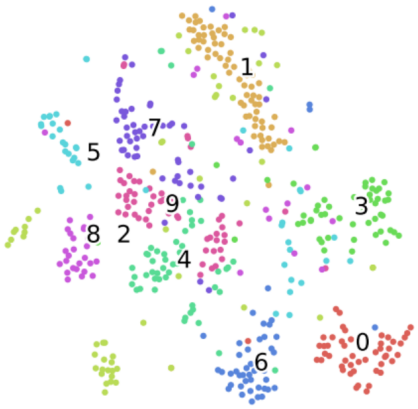

FIGURE 10 – t-sne sur un échantillon de la base train avant application d'un modèle One shot with triplet loss

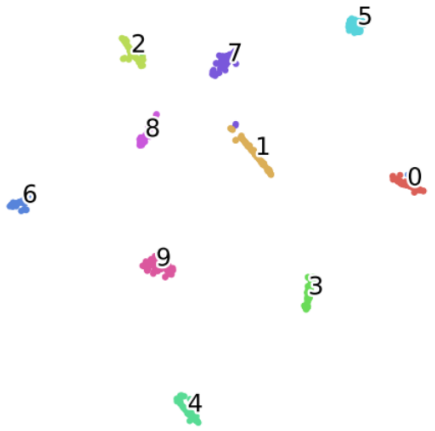

FIGURE 11 – t-sne sur les encodings d'un échantillon de la base train obtenus par un modèle One shot with triplet loss

3 CIFAR 10

3.1 Introduction

Dans cette section, nous présentons les différentes expériences et résultats effectués sur le dataset CIFAR 10.
Pour rappel, CIFAR 10 est une base de données d'images colorées et un peu pixélisées. Elle regroupe 50000 images d'apprentissage et 10000 images de test. Chaque exemple est une image de (32*32*3) pixels qui représente une classe (chat, avion, chien ...) étiquetée d'un chiffre entre 0 et 9 (10 classes).
Dans tous les modèles implémentés dans cette partie, nous utilisons le même

modèle de base CNN avec les mêms poids (branche CNN) préentraîné pour la tâche classification des images CIFAR10. Pour l'utiliser dans chacun des modèles Siamese, il suffit de supprimer sa dernière couche à 10 sorties. C'est un modèle qui prend en input une image et renvoie son encoding qui est de taille 128

3.2 Préparation des données d'apprentissage et de test

Comme pour MNIST, on a créé différentes base de données pour l'apprentissage (2 types de données pour les pairs et 3 types de données pour les triplets) et 3 bases de données pour le test.

Les différents types de base de données sont créés comme décrit dans la section **2.2**

3.3 Performances des modèles sur les paires d'images

On a utilisé le "transfer learning" pour les modèles qui se basent sur la "contrastive loss" et la "binary cross entropy loss". En effet on a essayé au début plusieurs modèles de base CNN mais malheureusement les performances sur la classification des pairs d'images (0 si les pairs de classes différents et 1 si les paires de la même classe) étaient décevantes. Ainsi, on a utilisé un modèle de base CNN (avec une légère modification sur la dernière couche) pré-entrainé sur tout le dataset CIFAR 10 qui avait 89% d'accuracy sur le "train set" et 85% sur le "test set".

Les tableaux ci-dessous montrent les performances des modèles sur les différents "test set" selon la base de données d'apprentissage utilisée :

	Train set	Easy test set	Medium test set	Hard test set
Type 1 train set	90%	81%	73%	71%
Type 2 train set	89%	80%	60%	66%

FIGURE 12 – Accuracy du modèle avec la contrastive loss

	Train set	Easy test set	Medium test set	Hard test set
Type 1 train set	96%	86%	75%	70%
Type 2 train set	96%	77%	69%	71%

FIGURE 13 – Accuracy du modèle avec la binary cross entropy loss

Contrairement à ce qu'on a remarqué pour MNIST, introduire une image par classe (pour les classes 5-9) n'a pas amélioré les performances mais les a dégradées pour les deux modèles (contrastive et binary cross entropy).

3.4 Comparaison des performances des modèles à base de triplets en fonction des types de triplets sélectionnés pendant l'entraînement

Tout comme les modèles précédents, on a utilisé le même modèle de base CNN (entraîné sur tout le dataset CIFAR) pour le modèle qui a les triplets d'images comme Input.

On résume les performances de ce modèle dans le tableau suivant : (j'attire votre attention au fait que les mentions "easy, medium, hard" sont différentes entre le test et le train sets)

	Easy test set	Medium test set	Hard test set
Easy train triplets	71%	68%	72%
Random train triplets	71%	69%	75%
Hard train triplets	70%	71%	75%

FIGURE 14 – Accuracy des modèle des triplets

Ce qui est impressionnant avec ce modèle, sachant qu'il est entraîné avec des "train set" qui se constituaient qu'avec des images de classes 0-4, est le fait qu'il est indifférent des classes des images utilisées dans le test set.

3.5 Comparaison des modèles avec le N way one shot learning

Revenant au problème du one shot learning, on a utilisé les modèles d'avant afin de générer les "encodings" des images pour exécuter un KNN qui nous permet de prédire leurs classes. Comme décrit en introduction, on a comparé les performances des modèles avec la technique du N way one shot learning (il est important de noter que les graphes ci-dessous sont des graphes plus lisses mais qui restent fidèles à ceux qu'on avait eu avec le n way one shot learning. En effet le temps d'exécution était long par rapport au nombre des répétitions du processus du N way one shot learning, ainsi on n'a répété ce processus que quelques itérations).

3.5.1 Comparaison entre les modèles des triplets

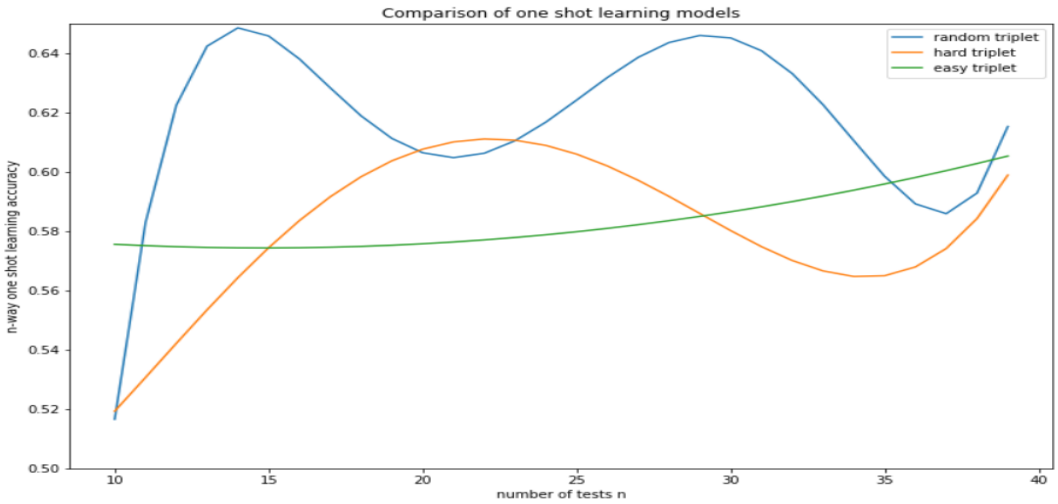

FIGURE 15 – Comparaison entre les modèles des triplets

En négligeant les extrémités, on peut bien voir que les résultats sont proches ainsi que les modèles random et hard triplet sont meilleurs que le modèle easy triplet. On peut aussi remarquer que la moyenne d'accuracy sur les modèles est de l'ordre de 62%, alors qu'avec

le modèle de base entrainé sur tout le dataset CIFAR 10 arrivait à une accuracy d'environ 85% sur le test set. Cette dégradation de performance est naturelle puisque dans le train set on avait que les classes 0-4.

3.5.2 Comparaison entre tous les modèles

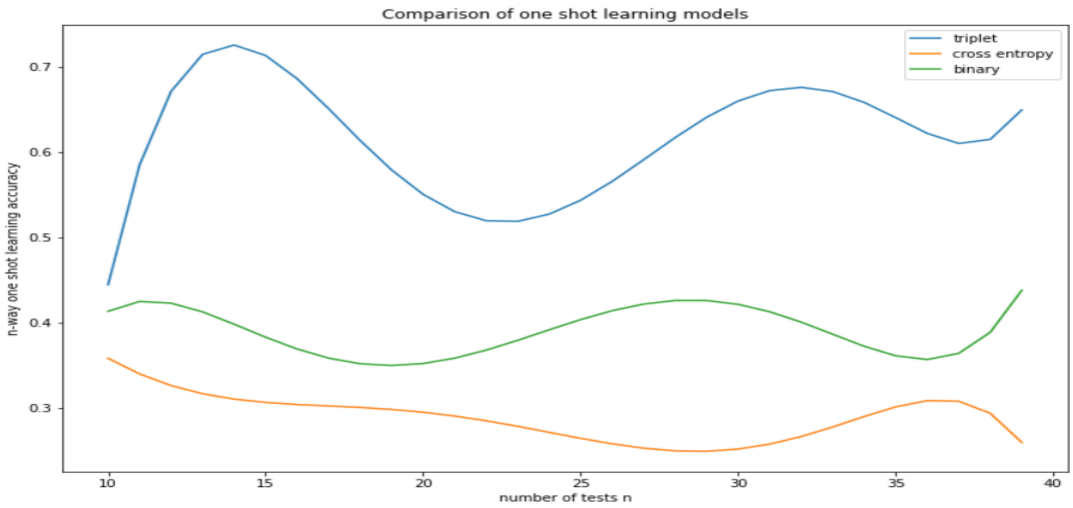

FIGURE 16 – Comparaison entre tous les modèles

Contrairement à MNIST, les modèles sont beaucoup moins performants sur CIFAR. On peut clairement distinguer que le modèle des triplets est largement meilleur que les modèles qui se basent sur les pairs d'images. Contrairement encore une fois à MNIST, le modèle de la binary cross entropy loss est mieux que celui de la contrastive. Les moyennes des accuracy sur tous les tests prouvent la même chose avec une moyenne de 62% pour le modèle des triplets, 40% pour le modèle de la binary loss et 29% pour le modèle de la contrastive loss.

Conclusion

Pour conclure, on remarque que les performances des réseaux siamois, sur les deux bases de données (MNIST et CIFAR), sont inférieures aux performances des modèles où on considère qu'on dispose assez d'images représentatives de toutes les classes (classification normale). Mais l'atout de ces réseaux, surtout pour les modèles des triplets, réside dans le fait que les performances ne se dégradent pas totalement et restent utilisables dans un contexte réel. Finalement, la différence des performances entre MNIST et CIFAR peuvent être expliquée par la simplicité du dataset MNIST ainsi que par la petite taille de ses images (28*28*1).

Références

[1] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[2] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet : A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. IEEE Computer Society, 2015.