# One Shot Learning

*Ghita Benjelloun, Chen Dang, Joachim Dublineau*

## Projet 2 - 24 novembre 2019

## Table des matières

# 1 Bases de données

## 1.1 MNIST

La base de données MNIST pour Mixed National Institute of Standards and Technology est une base de données de chiffres écrits à la main et très utilisée pour l'apprentissage automatique.

FIGURE 1 – Exemples de chiffres de la base MNIST

Cette base de données contient 60 000 données de train et 10 000 données de test. Ce sont des images en noir et blanc, normalisées centrées de taille 28*28*1.

## 1.2 CIFAR 10

Cifar 10 est une base de données d'images classées en 10 labels différents.

Elle contient en tout 60.000 images (6000 par classes) réparties en 50.000 images pour l'entraînement et 10.000 images pour le test. Les images sont de taille 32*32*3 ce qui les rend de qualité assez mauvaise mais très facile à manipuler pour entraîner des réseaux sans avoir un temps de calcul trop important. Les meilleurs réseaux obtiennent sur Cifar 10 aux alentours de 95% d'accuracy.

Cette base d'image possède des images suffisament différentes pour que l'entraînement soit pertinent sans necessiter de data augmentation. Cependant, nous y ferons appel pour voir quels résultats nous pouvons obtenir.

## 1.3 Fruits 360

Fruits 360 est une base de données d'images de 120 différents fruits et légumes [1]. Elle contient 61488 images en training set et 20622 en test set, chaque image contient un fruit ou un légume. Les images sont de taille 100*100*3. Différentes variétés du même fruit (pomme par exemple) sont stockées dans des classes différentes.

Dans notre étude, nous ne utilisons que 10 classes parmi les 120 classes. Les classes que nous avons utilisé sont : "Apple Red 1", "Apricot", "Avocado", "Banana","Cherry 1", "Clementine", "Cocos", "Grape Blue", "Lemon", "Mango".

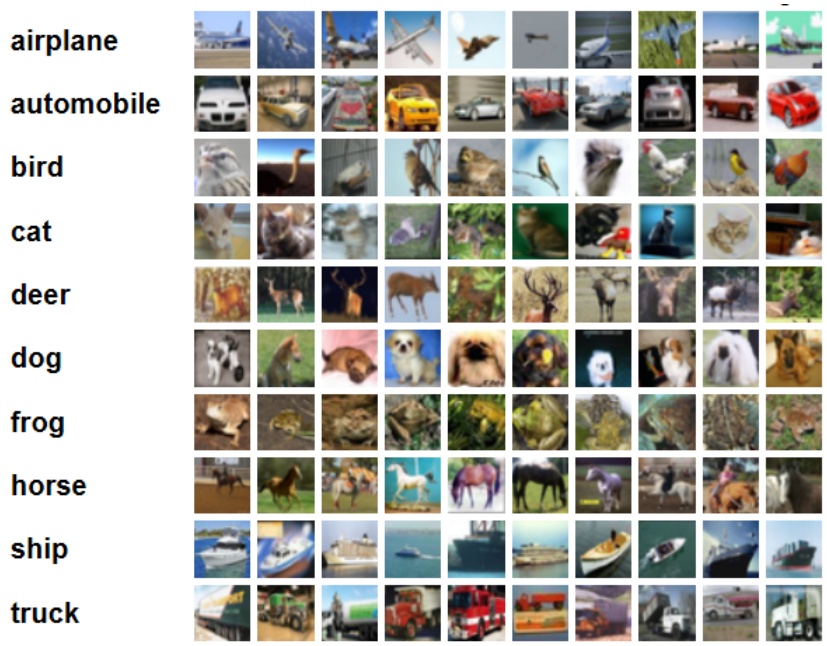



FIGURE 2 – Structure de Cifar 10

# 2 One Shot Learning

## 2.1 Entrainement du réseau

On va entraîner le réseau neuronal sur un jeu de données similaires aux données sur lesquelles on veut faire du one shot learning, par exemple si on veut détecter des objets, on va s'entraîner sur une base de données avec des images d'objets afin que le réseau apprenne à distinguer des objets différents en plus d'apprendre plus précisément à faire une prédiction de l'objet.

Au final, on obtient sur la couche juste avant la prédiction, une représentation de notre donnée d'entrée dans un espace abstrait. On va se servir de cette représentation ensuite pour évaluer des objets jamais appris (zero shot learning) ou des objets que nous avons rencontré que quelques fois.

Si la représentation est pertinente, les classes seront biens séparées dans l'espace de représentation et nous aurons peu de mal à faire une prédiction sur des objets que nous avons rencontré que peu de fois, voir une fois (one shot learning).

## 2.2 Extraction de la représentation

Supposons que nous disposons d'un réseau convolutif entrainé sur 5 classes de Cifar10 et que nous lui mettons en entrée une image de classe inconnue (ici une grenouille).

L'avant dernière couche (ici la verte n3 units) est extraite comme vecteur de représentation pour notre image de classe inconnue.

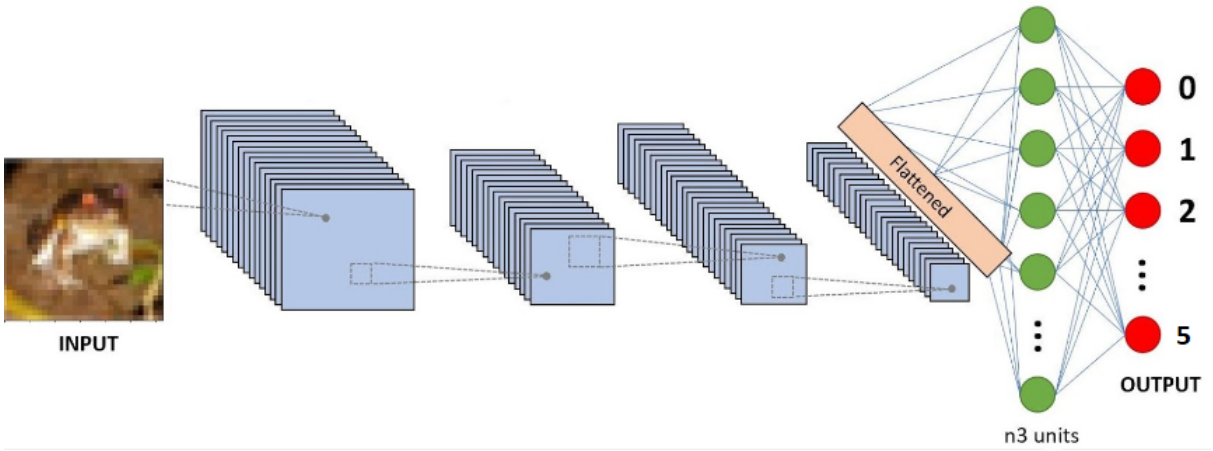



FIGURE 3 – Extraction de l'embedding

### 2.3 Prédiction à partir du vecteur de représentation

Plusieurs méthodes s'offrent à nous à partir de là. Si nous avons un seul réprésentant pour chaque classe (du vrai one-shot learning) alors nous attribuons à notre image, la classe correspondant à la classe de son plus proche voisin dans l'espace de représentation.

Si nous avons plusieurs réprésentants (few shots learning), on utilise le barycentre des vecteurs de représentation comme référence.

On peut aussi également prédire qu'une classe est inconnue en définissant un seuil de distance par rapport aux références. Si le vecteur de représentation est trop loin des références, alors il s'agit d'une nouvelle classe. On parle alors de zéro shot learning.

## 3 Résultats

### 3.1 MNIST

#### Choix et performances des réseaux

##### Entraînement simple - simple dense network

Nous avons commencé par entraîner notre jeux de données sur 5 classes de 0 à 4 et nous avons extrait les features du modèle par utilisation du simple réseau dense avec la categorical_crossentropy. Le résultat de séparation des classes par le réseau dense est le suivant :

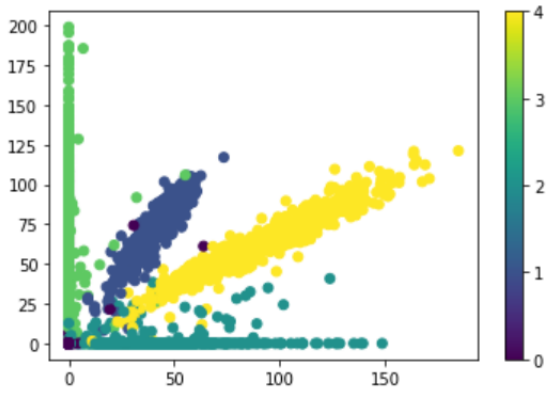



FIGURE 4 – plot des features de classes de 0 à 4 avec le réseau dense et la categorical cross entropy

Nous pouvons déduire que les classes ne sont pas bien séparées, par contre l'utilisation des réseaux siamois avec la contrastive loss va permettre une séparation bien meilleure dans l'espace latent. Ceci est illustré dans le paragraphe suivant

**Utilisation du siamese network et la contrastive loss** Le réseau siamois utilisé comporte 2 réseaux convolutionnels parralèles dont la composition est la suivante : 1 - Une couche de convolution de profondeur 128 2 - Une couche de Max Pooling 3 - Une couche de convolution de profondeur 256 4 - Une couche de Max Pooling 5 - Une couche de convolution de profondeur 512 6 - Une couche flatten Une projection sur un espace de deux dimension à l'aide d'une couche Dense de profondeur 2 donne le résultat suivant de la séparation entre les classes. Nous pouvons en déduire l'impact positif de l'utilisation de la contrastive loss

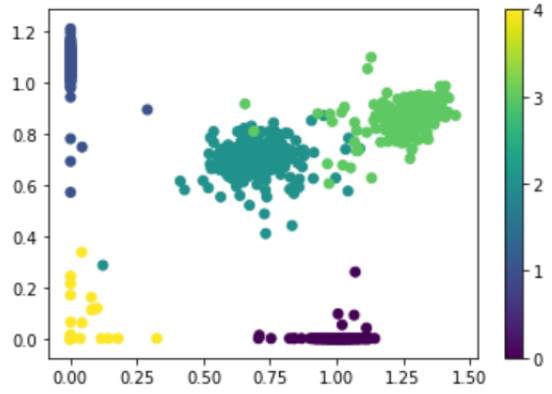



FIGURE 5 – plot des features de classes de 0 à 4 avec le réseau siamois et la contrastive loss

dans l'entraînement du réseau.

A la sortie des deux réseaux convolutionnels parralèle nous calculons la distance entre les deux vecteurs et nous entraînons ce résultat dans une couche Dense de profondeur 1 utilisant la fonction sigmoid.

**Résultats du one-shot learning** Après préparation et tuning de notre modèle, nous l'avons testé sur les classes de 7 à 9 est le résultat des performances du One Shot donne 52%

en moyenne pour chaque classe de 7 à 9 par rapport à l'ensemble de toutes les classes, ce qui est largement supérieur que le random (10%)

## 3.2 CIFAR 10

### Choix et performances des réseaux

#### CNN Classique

Nous avons essayé plusieurs formes de réseaux convolutif, avant de parvenir à ce réseau qui obtient 90% d'accuracy sur le test set après 100 epochs d'entraînement sur les 5 premières classes de CIFAR 10. Le fait de pas mettre trop de Max Pooling sur des images de faible qualité évite de perdre trop d'information. Ainsi nous avons finalement décidé de mettre que 2 max pooling en ayant fixé 5 couches de convolution.

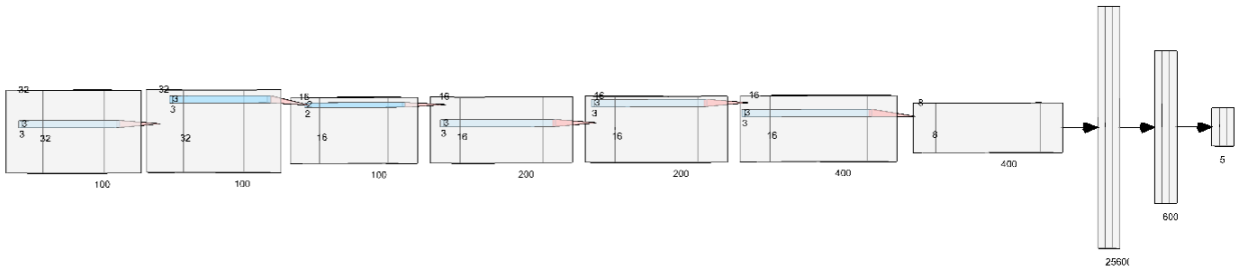



FIGURE 6 – CNN (100, 100, MaxPool, 200, 200, 400, MaxPool, Dense : 25600, 600, 5)

Hyperparamètres d'entrainement :
- Learning rate = 0.0001
- Batch size = 32
- epochs = 100
- Weight Penalty = 0.0001
- Optimizer = Adam
- Loss = Categorical Cross-Entropy
- Data Augmentation

Performances sur le set d'entraînement :
- accuracy : 95%
- loss : 0.32

Performances sur le set de test :
- accuracy : 91%
- loss : 0.46

#### CNN Embedded in a Siamese Neural Network

Nous avons également fait un réseau siamois qui embarque à le précédent réseau CNN : nous obtenons avec celà les performances suivantes.

Hyperparamètres d'entrainement :
- Learning rate = 0.0001
- Batch size = 32
- epochs = 50
- Weight Penalty = 0.0001
- Optimizer = Adam
- Loss = Constrative Loss

Performances sur le set d'entraînement :
- accuracy : 90%
- loss : 0.11

Performances sur le set de validation :
- accuracy : 76%
- loss : 0.23

**Tests en one-shot learning**

Nous avons également testé les performances de nos réseaux en one-shot learning. Regardons tout d'abord la pertinence de la représentation, avec la méthode de t-SNE nous pouvons représenter en 2 dimensions de vecteurs de dimension supérieure (en l'occurrence 600). Ainsi nous pouvons voir si notre représentation permet de bien séparer les classes non apprises.

Avec le CNN voici ce que nous obtenons : Figure 7

Avec le Siamese voici ce que nous obtenons : Figure 8

Il est visible que les classes 5, 6 et 7 sont très entremélées et que il y a une séparation assez nette entre les 5, 6, 7 et les 8 et 9. Le réseau fait donc en quelques sortes au moins bien la différence entre les objets (ship et trucks) et les animaux (dog, frog et horse).

Voici les résultats obtenus en termes d'accuracy pour les différentes classes pour le CNN classique :
- Class Dog 5 accuracy : 0.41
- Class Frog 6 accuracy : 0.48
- Class Horse 7 accuracy : 0.42
- Class Ship 8 accuracy : 0.60
- Class Truck 9 accuracy : 0.57
- average accuracy : 50%

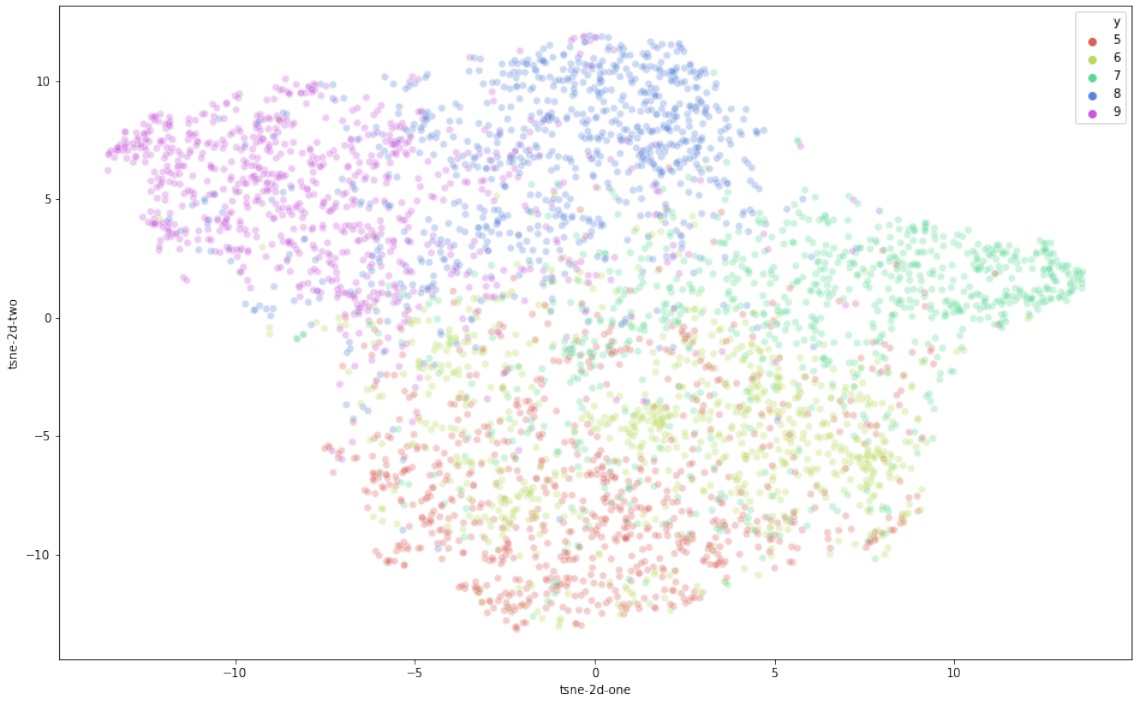



FIGURE 7 – t-SNE sur les embedding du réseau CNN simple

Et voici les résultats obtenus en termes d'accuracy pour les différentes classes pour le réseau siamois :

- Class Dog 5 accuracy : 0.47
- Class Frog 6 accuracy : 0.35
- Class Horse 7 accuracy : 0.42
- Class Ship 8 accuracy : 0.58
- Class Truck 9 accuracy : 0.63
- average accuracy : 49%

Il est intéressant de voir que le réseau siamois sans data augmentation et avec moins d'epochs obtient des performances assez similaires au réseau CNN simple.

Voici la matrice de confusion obtenue avec une référence par classe, en changeant 300 fois de références et en faisant la prédiction de la référence la plus proche pour le réseau CNN :

| | class 5 | class 6 | class 7 | class 8 | class 9 |
|---|---|---|---|---|---|
| préd. 5 | 123042 | 81701 | 54963 | 13487 | 8824 |
| préd. 6 | 114572 | 145033 | 88235 | 22944 | 14952 |
| préd. 7 | 46887 | 48260 | 126198 | 23519 | 12572 |
| préd. 8 | 8930 | 14883 | 21067 | 180697 | 92885 |
| préd. 9 | 6269 | 9823 | 9237 | 59053 | 170467 |

La même matrice pour le réseau siamois donne :

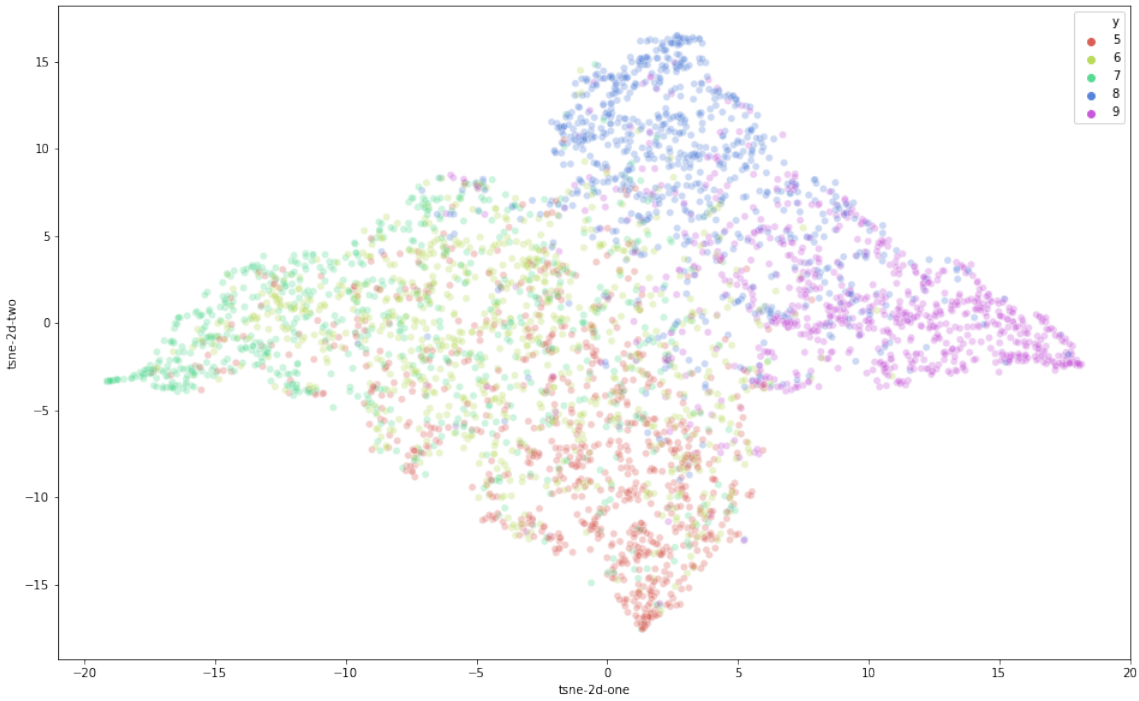



FIGURE 8 – t-SNE sur les embedding du réseau Siamese

| | class 5 | class 6 | class 7 | class 8 | class 9 |
|---|---|---|---|---|---|
| préd. 5 | 139849 | 82652 | 59805 | 13114 | 9722 |
| préd. 6 | 93945 | 104535 | 93192 | 26959 | 18852 |
| préd. 7 | 49480 | 76670 | 124959 | 17846 | 10349 |
| préd. 8 | 8485 | 21792 | 15078 | 175130 | 71163 |
| préd. 9 | 7941 | 14051 | 6666 | 66651 | 189614 |

On peut voir qu'il y a beaucoup de confusion entre dog, frog et horse ainsi que beaucoup de confusion entre ship et truck. Ces confusions ne sont pas totalement absurdes étant donnée la qualité des images.

**Test en few shots learning**

En définissant les références à partir de 3 "shots" et non plus un seul "shot", nous pouvons bien améliorer les performances de la prédiction en se prémunissant contre des références qui seraient très proche dans l'espace de représentation d'autres classes. Voici les performances que nous obtenons pour 3 shots avec le CNN :

- Class Dog 5 accuracy : 0.53
- Class Frog 6 accuracy : 0.55
- Class Horse 7 accuracy : 0.58
- Class Ship 8 accuracy : 0.75
- Class Truck 9 accuracy : 0.70
- average accuracy : 62%

| | class 5 | class 6 | class 7 | class 8 | class 9 |
|---|---|---|---|---|---|
| préd. 5 | 153199 | 25933 | 12810 | 2435 | 1247 |
| préd. 6 | 33226 | 54897 | 22827 | 4672 | 2696 |
| préd. 7 | 11030 | 13171 | 57597 | 4654 | 2032 |
| préd. 8 | 1916 | 4505 | 5777 | 74683 | 23859 |
| préd. 9 | 529 | 1394 | 889 | 13456 | 70066 |

Avec le modèle Siamois, voici ce que nous obtenons en 3-shots learning.

- Class Dog 5 accuracy : 0.59
- Class Frog 6 accuracy : 0.35
- Class Horse 7 accuracy : 0.53
- Class Ship 8 accuracy : 0.69
- Class Truck 9 accuracy : 0.73
- average accuracy : 58%

| | class 5 | class 6 | class 7 | class 8 | class 9 |
|---|---|---|---|---|---|
| préd. 5 | 58529 | 26311 | 13800 | 2515 | 3019 |
| préd. 6 | 24697 | 35131 | 28017 | 5286 | 3976 |
| préd. 7 | 14704 | 29264 | 53260 | 2658 | 1557 |
| préd. 8 | 1426 | 6693 | 4130 | 69218 | 18192 |
| préd. 9 | 544 | 2501 | 693 | 20223 | 73156 |

## 3.3 Fruits 360

### 3.3.1 Choix et performances des réseaux

**Préparation des données**

Dans notre étude, seulement 10 fruits dans le dataset ont été utilisés, il sont : "Apple Red 1", "Apricot", "Avocado", "Banana","Cherry 1", "Clementine", "Cocos", "Grape Blue", "Lemon", "Mango". Les images de taille 100*100 pixels ont été redimensionnés à la taille 32*32 pixels pour diminuer le temps de calcul. Les valeurs dans la matrix sont normalisées de 0 à 1.

Le training set de notre étude est choisi parmi le training set de 5 premières classes de fruits, le validation set est le test set de 5 premières classes, et le test set est pris dans le training set de 5 dernières classes. Nous avons aussi utilisé le Data augmentation pour augmenter le nombre d'images dans notre training set.

**Réseau Siamese**

Nous avons choisi le réseau siamese pour étudier cette base de données. De nombreuses architectures du CNN qui est le modèle de base ont été testé pour essayer de trouver la meilleure performance sur le dataset. Nous avons aussi essayé le transfert learning : un réseau VGG19 pré-entraîné sur ImageNet a été utilsié comme le modèle de base. Tous les couches convolutives de VGG19 ont été importé et on a ajouté deux couches denses après le modèle. Cependant, le procédure de convergence est trop lent, et le résultat qu'on a obtenu n'est pas assez satisfaisant. Enfin, nous avons abondonné l'idée de transfert learning, et choisi cette architecture de CNN comme montrée dans Figure 9 :

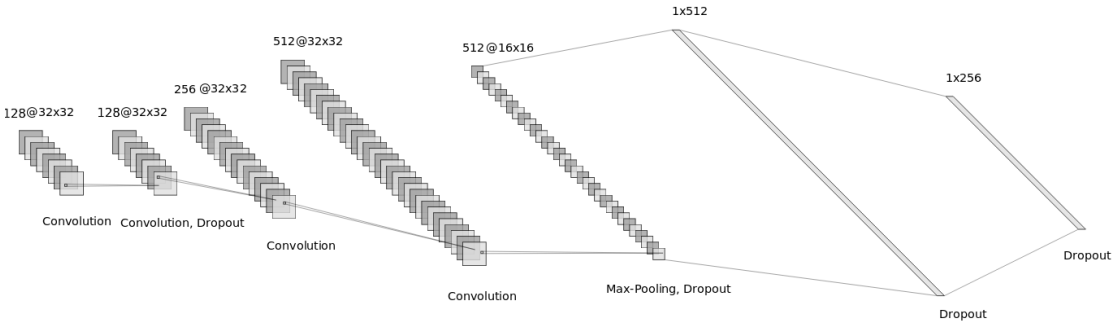



FIGURE 9 – Architecture du CNN dans le réseau siamese

**Résultats et performance**

Nous avons fait beaucoup de tests pour augmenter l'accuracy de notre modèle. Pourtant pendant notre essais, l'accuracy de training set est toujours sur 50%, par contre l'accuracy sur le validation set continue à augmenter. On suppose que c'est une erruer, et on a pris l'accuracy sur le validation set comme indicateur de la performance du système. La convergence du loss et l'accuracy du modèle de meilleure performace est montré dans le Figure 10 :

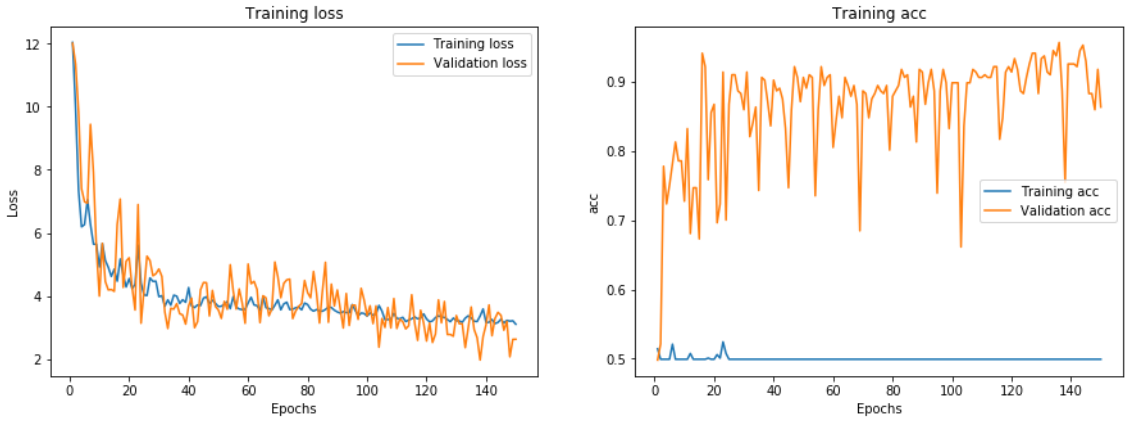



FIGURE 10 – Loss et accuracy du réseau siamese

Finalement, on a 0.85 d'accuracy sur le training set et 0.91 d'accuracy sur le test set. En général, la meilleure performance en test set n'est pas normal. Pourtant nous avons bien vérifier que le modèle n'a pas rencontré les données de test set. Par conséquent, on suppose que cela soit dû à la grande distance entre les différentes classes du test set.

Pour visualiser l'embeddings générés par notre modèle, on a extrait les résultats de la dernière couche de CNN, et utilisé t-SNE sur les embeddings, comme montré le Figure 11 :

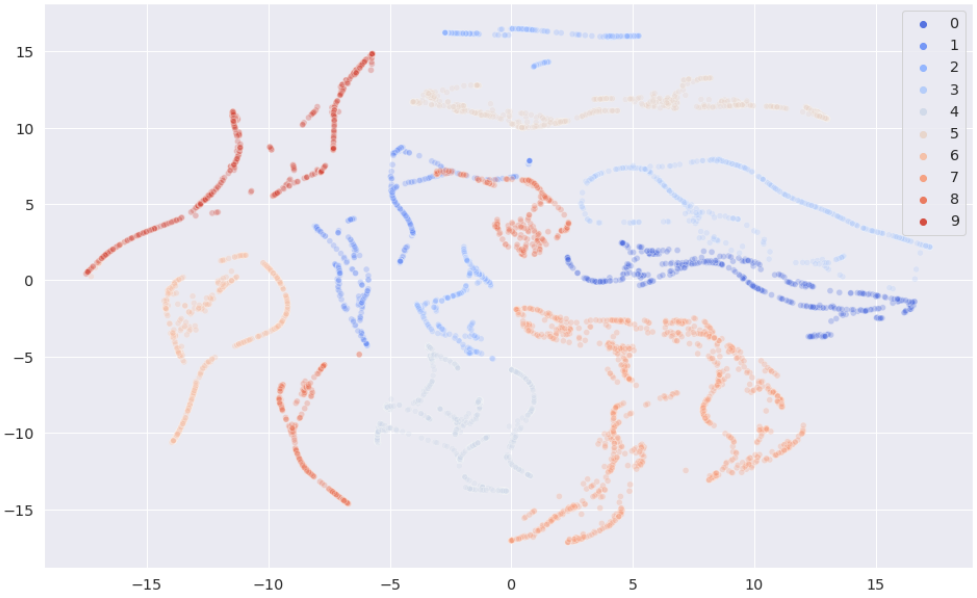



FIGURE 11 – t-SNE sur les embeddings des fruits

On a remarqué que les embeddings de chaque fruit forme plutôt une courbe au lieu d'un groupe. Nous pensons que cela est dû au fait que les photos de chaque fruit sont prises sous différents angles d'un ou deux fruits(comme montré le Figure 12). Par conséquent, les coordonnées d'un même fruits de différents angles formes une courbe dans l'espace d'embeddings.

FIGURE 12 – Photos d'un banane sous différents angles dans le dataset

### 3.3.2 Tests en one-shot learning

Nous avons ensuite fait des tests sur one shot learning. Pour evaluer la performance, on a choisi aléatoirement 4096 images dans 5 dernières classes de dataset pour prédire la classe. Et pour chaque image, on a choisi aléatoirement une image dans chaque 10 classes. Donc c'est un problème de 10-way one shot classification. Enfin, on a obtenu 0.77 d'accuracy sur la classification. La matrice de confusion est comme le Figure (13) :

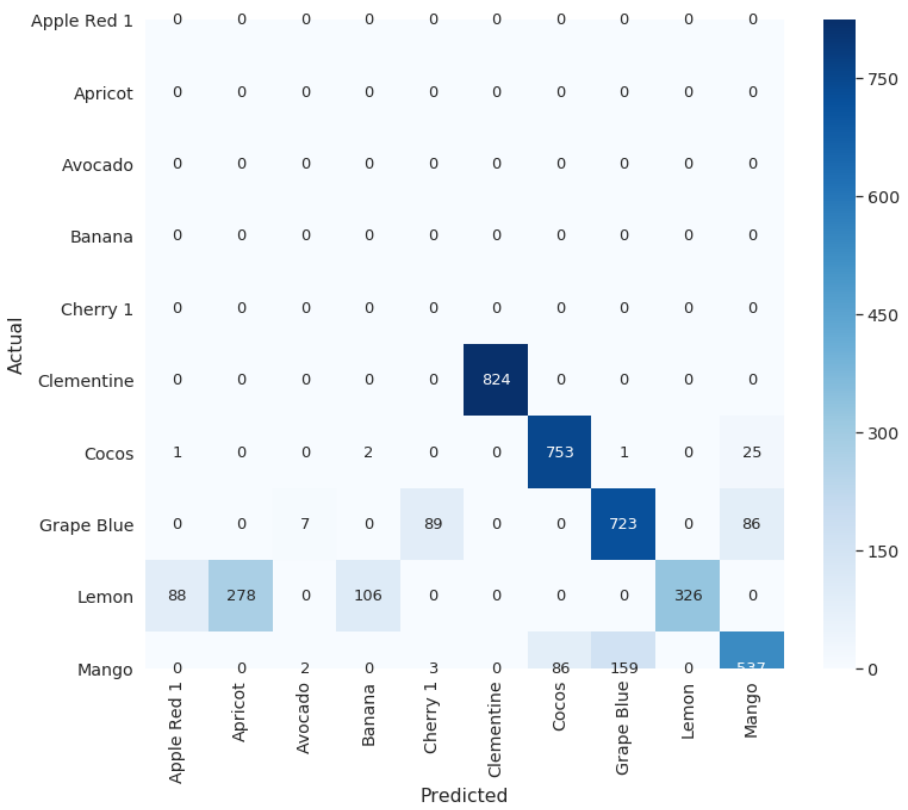



FIGURE 13 – La matrice confusion du classification de test set

Nous avons aussi fait la classification dans tout ensemble de données. Les images à classifier sont choisi de manière pareil que précédent : 4096 images ont été choisi parmi 10 classes. L'accuracy qu'on a eu est 0.69. La matrice de confusion est comme suivant :

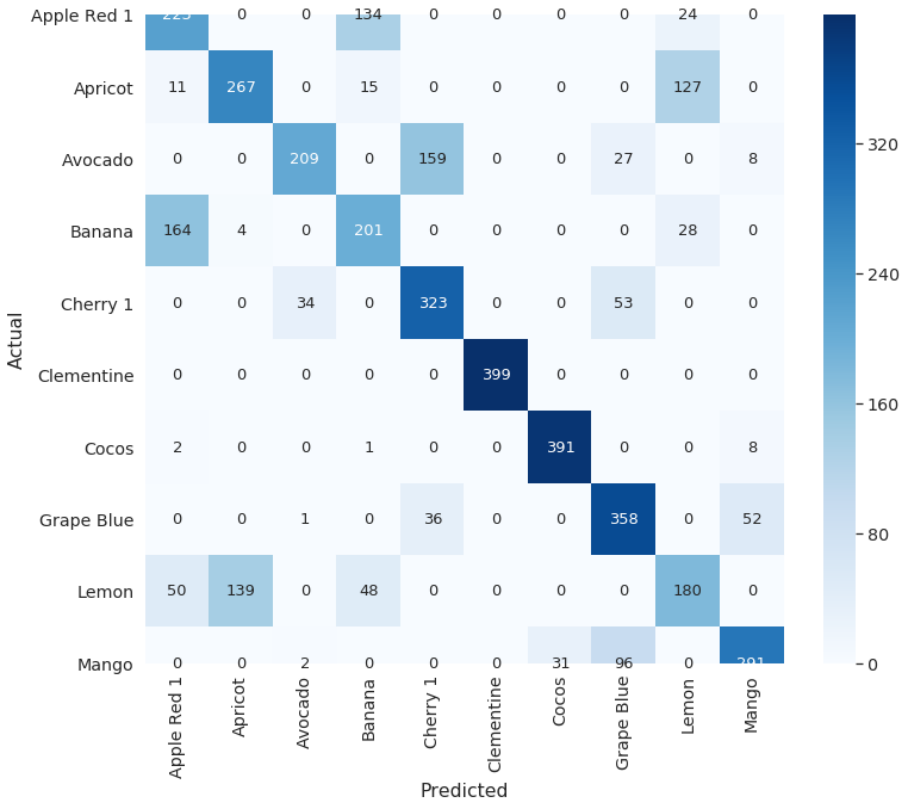



FIGURE 14 – La matrice confusion du classification de tout ensemble de données

### 3.3.3 La reconnaissance d'objets

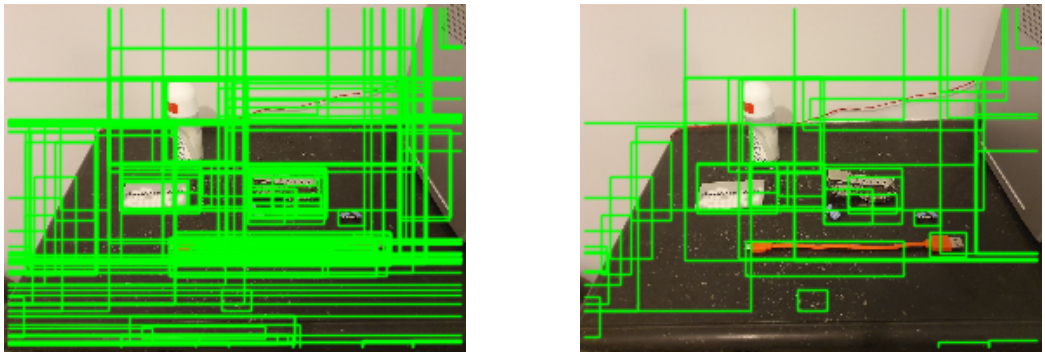
Nous avons aussi essayé de faire la reconnaissance d'objets avec noter réseau siamois. Nous ne mettons l'objet de reconnaissance que comme les dix fruits dans notre étude.

**Recherche sélective**

D'abord, pour détecter les objets dans une image, on a utilisé une méthode de recherche sélective [2], qui combine la force de la recherche exhaustive et de la segmentation. Cette méthode est aussi utilisé dans le réseau R-CNN et Fast R-CNN pour proposé des régions d'intérêt(roi).

Au lieu de faire l'algorithme apprendre de proposer les roi, nous avons sélectionné les roi par un algorithme qui élimine les régions trop fines. Si deux régions se resemble beaucoup, on prend la région la plus grands. Le Figure (15) montre les roi proposées par recherche sélective et les roi sélectionnées par notre algorithme. Le temps d'exécution de notre algorithme pour la sélection de roi de cette image est 0.08s sur le CPU fourni par Google Colab.

Nous avons aussi remarqué que les régions qu'on a obtenu ne sont pas très appropriées parfois. Pourtant les objets dans l'images qui nous intéressent sont tous encadrés. Dans l'étape prochain, on va utiliser les petites images dans ces régions pour faire la classification.

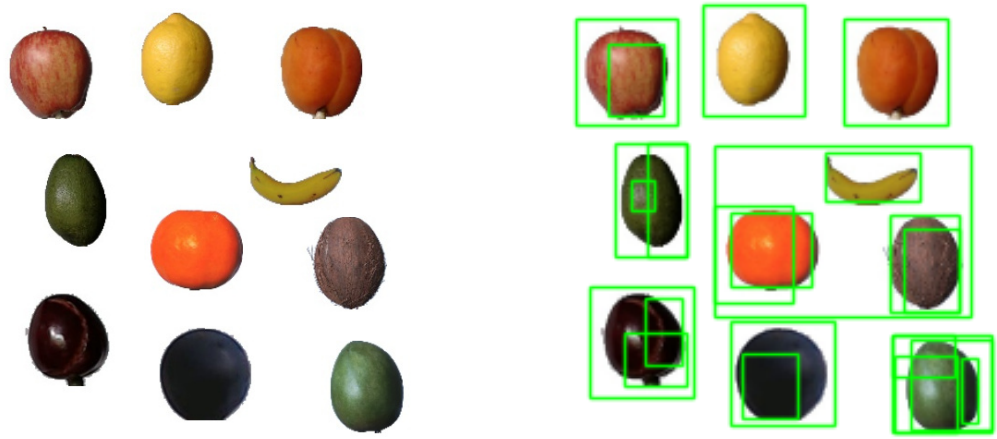


(a) roi proposé (b) roi sélectionné

FIGURE 15 – Comparaison de roi proposées et roi sélectionnées

**La reconnaissance de fruits**

Pour évaluer la performance sur la reconnaissance de fruits, nous avons choisi un fruit par classe dans les 10 classes dans de dataset Fruits 360, et crée une image pour faire la détection d'objets.



(a) image de fruits de tous les classes (b) roi détecté

FIGURE 16 – image utilisé pour la détection d'objet

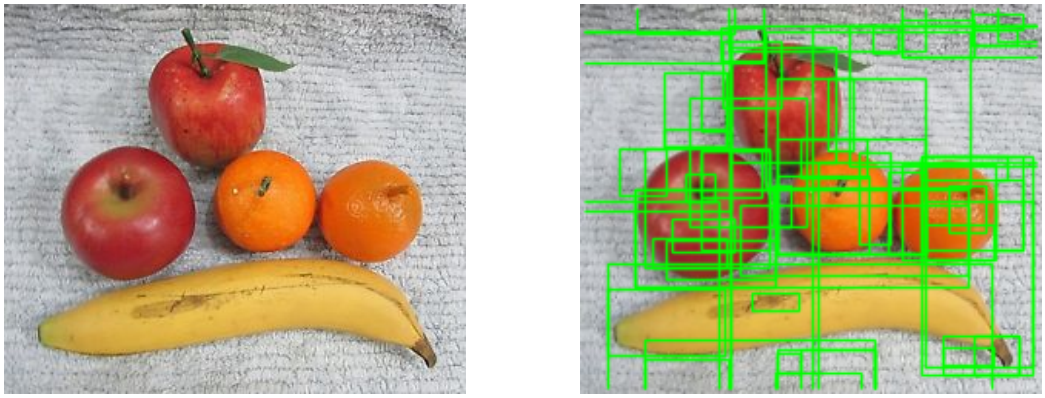
Pour chaque petites images dans cette image, on a fait la classification avec notre réseau siamese. Le résultat est montré comme suivant :

| Objet | Apple | Apricot | Avocado | Banana | Cherry | Clement. | Cocos | Grape | Lemon | Mango |
|---|---|---|---|---|---|---|---|---|---|---|
| Préd. | Clement. | Apricot | Cherry | Banana | Cherry | Clement. | Cocos | Grape | Apricot | Mango |

Parmi tous les 10 images, 3 fruits sont mal classifiés. C'est cohérent avec notre résultat de 0.69 d'accuracy sur tout ensemble de 10 classes.

Après, nous avons essayé de faire la classification des fruits avec les images sur Internet. Cependant nous avons rencontré beaucoup de problèmes, par exemple, la différent saturation et variétés des fruits a beaucoup dérangé la classification. En plus, recommandation de zone inappropriée par notre algorithme peut aussi affecter les performances du système. Enfin, on vais juste donner un exemple pour montre la performance de notre modèle en pratique.

Le figure 17 montre une image qui contient 2 pommes, 2 clémentines et une banane qu'on a trouvé sur Internet, et les roi proposés. Le figure 18 montre les roi qui sont appropriés et la classification corresondant à chaque objet.



(a) image de fruits sur Internet (b) roi détecté

FIGURE 17 – image utilisé pour la détection d'objet

FIGURE 18 – Les roi choisis manuellement, les classifications sont : apricot, apple red, clementine, lemon, banana

# 4 Conclusion

Nous avons pu voir à travers ce projet que l'utilisation des vecteurs de représentation offre de grandes possibilités. Il peut permettre de représenter de manière abstraites les informations contenues dans une donnée. Le one-shot learning est une bonne application de ces vecteurs de représentation. On peut ainsi à partir d'un seul enregistrement par classe (référence), prédire la classe d'une entrée inconnue. Nos modèles ont été entrainé sur plusieurs bases de données MNIST, CIFAR10 et Fruit 360 et on a à partir de ça, pu obtenir de bonnes performances en one-shot learning que ce soit en faisant appel à des réseaux CNN simples, ou à des réseaux siamois entraîné avec la contrastrive loss. Cependant les performances de ces modèles reste limitées et pour les améliorer, il faudrait complexifier un peu les modèles et optimiser certains hyperparamètres comme la marge de la contrastive loss.

Nous nous sommes ensuite intéressé à la détection d'objets. Le but était de voir si le principe du one shot-learning pouvait bien s'appliquer à cela. Nous avons rapidement déterminé que la méthode de la fenêtre glissante n'était pas la méthode optimale et nous somme partie sur une méthode de détection de zone d'intérêt. Avec un réseau entrapiné sur une base de données de fruit, nous sommes capables de détecter les objets, et ensuite de les classifier. Cependant, il est encore difficile de distinguer un fruit d'un non-fruit. Il faudrait améliorer

les performances du modèle pour être capable de faire cela avec à la clé, pourquoi pas, la possibilité de faire du zéro-shot learning.

## Références


[1] Horea Muresan, Mihai Oltean, Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018.

[2] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, A.W.M. Smeulders, "Selective Search for Object Recognition", In International Journal of Computer Vision, 2013.