Louis Boudailliez
Grégoire Caussou

Projet : Estimation d'incertitude pour les réseaux de neurones

Sommaire

1. Présentation du sujet

Les décisions prises par les réseaux de neurones manquent d'explicabilité et l'on peine à associer un degré de confiance à chaque prédiction. Dès lors, il devient difficile d'appliquer ces méthodes à des problématiques réelles ou chaque décision peut avoir des conséquences dramatiques. C'est notamment le cas de la robotique collaborative où une mauvaise décision pourrait endommager des objets, voire blesser un opérateur. L'enjeu est donc de trouver des moyens pour garder les performances élevées de ces techniques tout en associant une mesure de l'incertitude pour les utiliser plus sereinement.

L'objectif de ce projet est donc de detecter les mauvaises prédictions d'un réseau de neurone que nous aurions préalablement entrainé.

Dans un premier temps, nous allons nous approprier la base de donnée MNIST et développer un réseau de neurone permettant de determiner la valeur du chiffre manuscrit donné en entré.

Ensuite nous allons évaluer ses performances avec les méthodes de validation croisé et matrice de confusion.

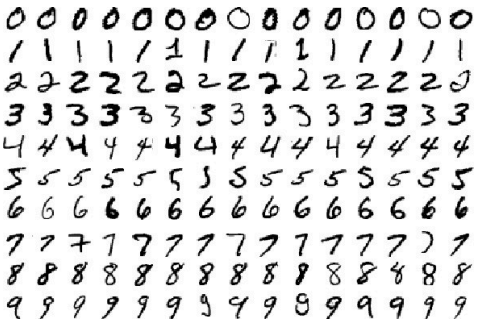
Nous allons en parallèle entrainer un nouveau réseau de neurone mais en n'utilisant seulement 8 classes pour l'entrainement et la validation, et nous allons ensuite l'utiliser pour pour determiner la valeur des deux classes que nous avons supprimé et en analyser les résultats.

Enfin, nous allons implémenter la méthode « Monte-Carlo Dropout » pour calculer une incertitude sur les résultats du réseau de neurone.

Nous pourrons alors verifier si les classes supprimé dans le second réseau sont rejetés par une incertitude trop élevée.

2. Développement d'un réseau de neurone classique

Nous avons décidé d'utiliser la base de donnée MNIST composée de 70 000 chiffres manuscrits (image en noir et blanc de taille 28x28) ainsi que du chiffre correspondant. La fonction de ce réseau neuronal est de prédire le chiffre manuscrit.



Cette base de donnée est présente l'avantage d'avoir un très grand nombre d'échantillons exploitable depuis Keras et dont une structure neuronal très efficace est deja connue (précision de 99%)

```
Test loss: 0.027417584759772 58
Test accuracy: 0.991
```

Les performances de ce réseau sont mise en évidence par deux méthode : la validation croisée (*cross validation*) et la matrice de confusion

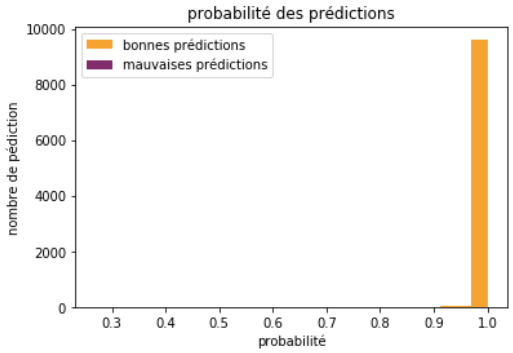
screen de la cross validation

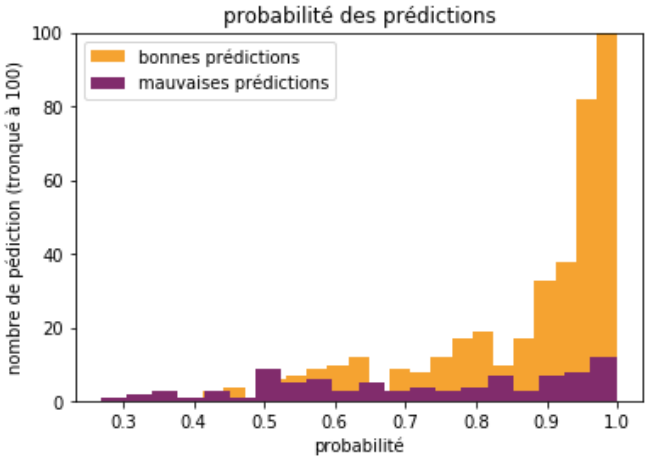
```
[[ 975    0    2    0    0    0    2    1    0    0]
 [   0 1135    0    0    0    0    0    0    0    0]
 [   1    2 1025    0    1    0    0    3    0    0]
 [   0    0    2 1003    0    4    0    0    1    0]
 [   0    0    0    0  977    0    3    0    0    2]
 [   2    0    1    4    0  883    2    0    0    0]
 [   4    2    0    0    1    4  947    0    0    0]
 [   0    3    5    1    0    0    0 1016    1    2]
 [   3    1    2    0    0    0    1    2  963    2]
 [   1    2    0    1    7    6    0    5    1  986]]
```

3. Première méthode de détection des mauvaises prédictions

Une première méthode de détection des mauvaises prédiction s'appuie sur les valeurs des neurones de sortie (un par classe donc 10). Chacun renvoie une valeur entre 0 et 1 et la somme des dix valeurs vaut 1, ce qui peut donc être vu comme des probabilité d'appartenir à chaque classe. En fixant une valeur seuil, on peut chercher à minimiser (voir supprimer!) les mauvaises prédictions.

Ainsi, nous avons visualisé les bonnes et mauvaises prédictions associées à la probabilité d'appartenir à la classe prédite.





Mais on constate que certaines mauvaise prédictions sont quand même données avec une très grande certitude (probabilité proche de 1). Dès lors, même en utilisant une valeur seuil très élevée (0,99 par exemple) certaines mauvaise prédictions de sont pas éliminées, et bon nombre de bonne prédiction sont rejetées à tort.

```
la matrice de confusion pour seuil de confiance = 0.9 est
[[9776  134]
 [  33   57]]
la matrice de confusion pour seuil de confiance = 0.95 est
[[9711  199]
 [  25   65]]
la matrice de confusion pour seuil de confiance = 0.99 est
[[9539  371]
 [  11   79]]
```

Par exemple ici, pour un seuil de 99%, 11 mauvaises prédictions ne sont pas rejetés (12%) tandis que 371 bonnes prédictions le sont (3,7%).

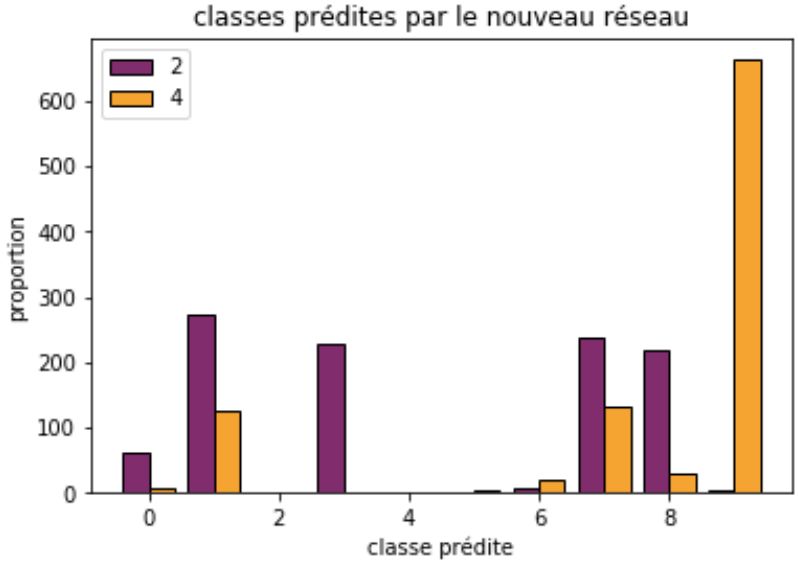
4. Évaluation de la première méthode

Pour évaluer cette méthode de détection des mauvaises prédictions, nous allons entrainer un nouveau réseau de neurone réalisant la même fonction : déterminer la valeur d'un chiffre manuscrit, mais en lui retirant deux classes. On va donc l'entrainer sans jamais lui montrer de chiffre 2 et 4 (choisis arbitrairement)

```
[[ 977    0    0    0    0    0    1    1    1    0]
 [   0 1134    0    1    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0]
 [   0    0    0 1001    0    6    0    2    1    0]
 [   0    0    0    0    0    0    0    0    0    0]
 [   2    0    0    2    0  886    2    0    0    0]
 [   5    2    0    1    0    1  946    0    2    1]
 [   0    2    0    1    0    0    0 1024    1    0]
 [   3    1    0    0    0    0    0    2  965    3]
 [   1    2    0    1    0    2    1    5    1  996]]
```

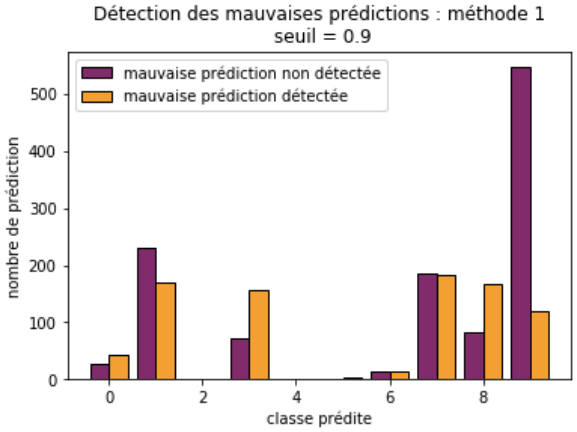
Cette matrice de confusion illustre le bon fonctionnement de ce nouveau réseau de neurone (nous avons laissé les colonnes 2 et 4 pour montrer qu'elles sont bien vides)

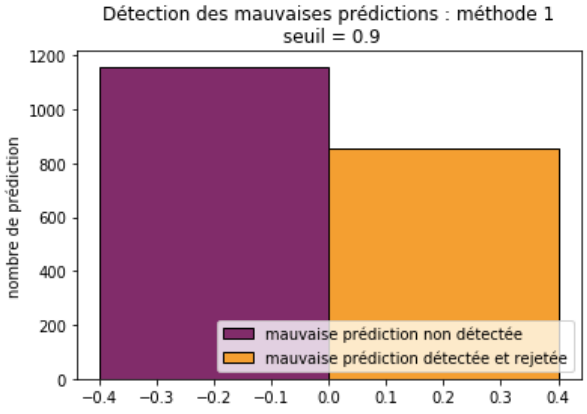
Nous allons donc présenter des images de 2 et de 4 à notre nouveau réseau de neurone et analyser sa réaction ce qui nous servira de témoins



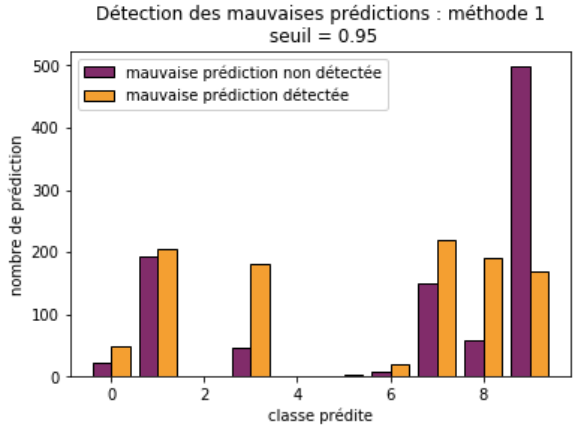
On constate que les deux nouvelles classes sont traités par le nouveau réseau : les 2 sont globalement reconnus comme 1, 3, 7 et 8, tandis que les 4 sont majoritairement reconnus comme des 9.

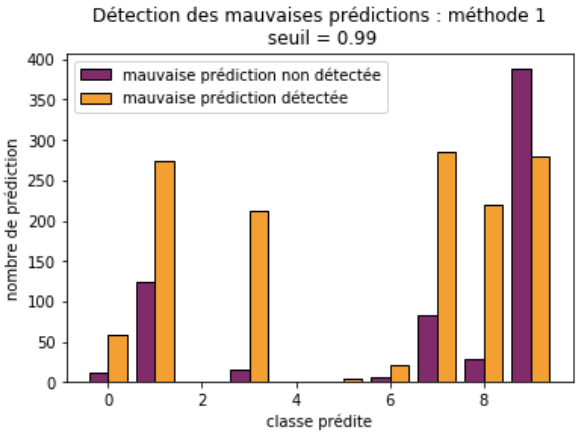
Nous allons maintenant appliquer cette première méthode présentée en 3. au traitement des deux classes inconnues par notre réseau de neurone :

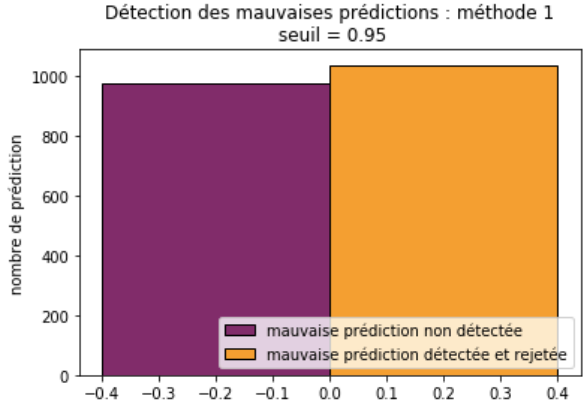




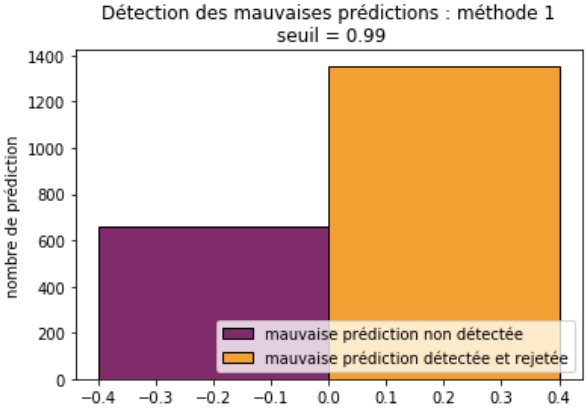

```
43 % de mauvaise prédictions détectées
57 % de mauvaise prédictions non détectées
```








```
51 % de mauvaise prédictions détectées
49 % de mauvaise prédictions non détectées
```




```
67 % de mauvaise prédictions détectées
33 % de mauvaise prédictions non détectées
```

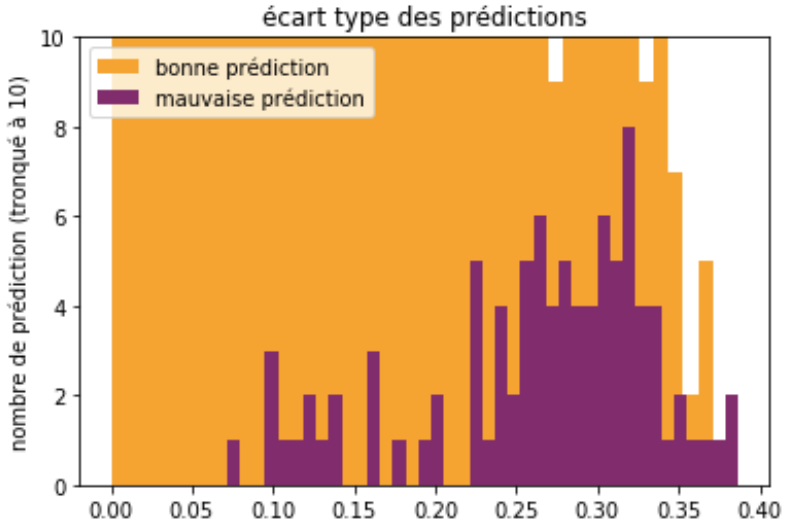
On constate sans grande surprise que plus le seuil est élevé et proche de 1, plus les mauvaises prédictions sont détectées et mieux elle sont triées. Par exemple, plus de 600 « 4 » étaient initialement lu comme des « 9 » par le réseau de neurone ; après application de cette première méthode avec un seuil de 0,99, environ 500 d'entre eux sont détectés comme de mauvaises prédiction. Cette première méthode est donc efficace pour detecter les mauvaises prédictions.

Cependant, des mauvaises prédictions pourront toujours passer le filtre, de plus une valeur trop élevée du seuil provoque l'inefficacité de réseau : en effet trop de bone prédictions sont détectées comme mauvaise et ne sont donc pas exploitables. Nous allons donc nous intéresser à une autre méthode au point suivant.

5. Deuxième méthode de détection des mauvaises prédictions : la méthode de *Monte-Carlo Dropout*

La méthode de *Monte-Carlo Dropout* est une approche probabiliste de prédiction, qui s'appui sur plusieurs prédictions différentes d'une même entrée, puis d'un regroupement de ces nombreuses prédictions. Pour réaliser cela, on utilise un *dropout* qui va aléatoirement masquer certains poids pour chacune des prédictions, puis on va en calculer la moyenne des sortie et leur écart-type. En effet si une prédiction est fiable, elle va peu varier lorsque le réseau sera « bruité » par un masquage aléatoire de certain poids.

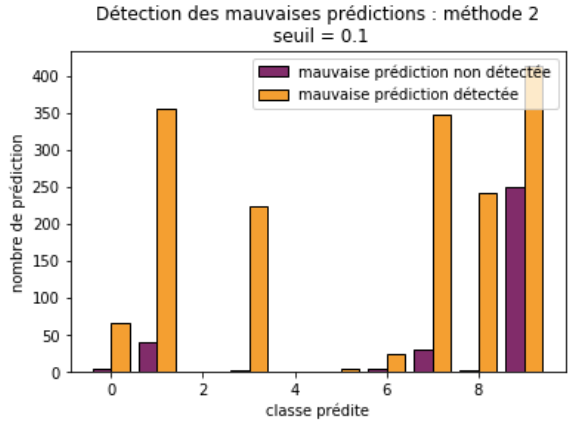
La moyenne la plus haute correspond alors à la classe la plus probable pour l'entré, et son écart-type traduit la confiance que l'on peut accorder à cette prediction (si les probabilités d'appartenir à la classe n'est pas sensible au bruit, c'est qu'elle est plutôt certaine)

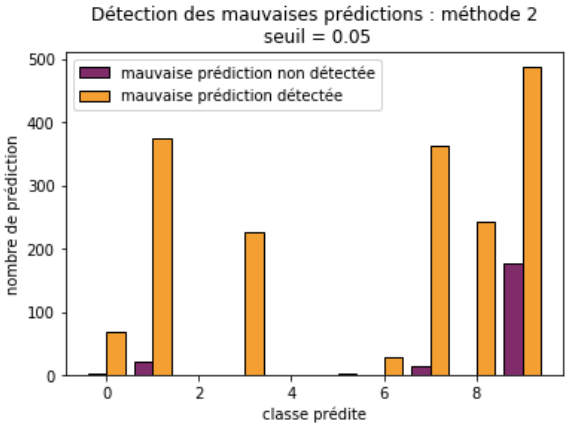


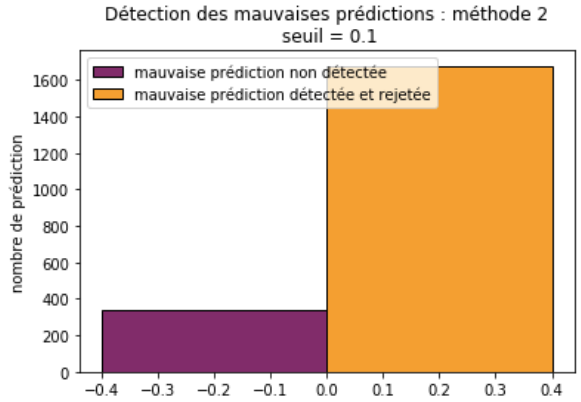
On constate sur cet histogramme que toutes les mauvaises prédictions ont un écart type supérieur à 0,05 ce qui en fait un critère de détection des mauvaises prédictions : si l'écart type d'une prédiction ainsi calculée est inférieure à cette valeur seuil, on peut affirmer qu'elle est certaine, sinon il y a des risques qu'elle soit mauvaise.

6. Évaluation de la deuxième méthode

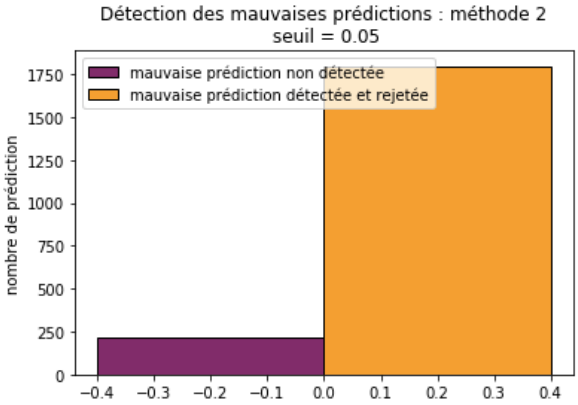
Nous allons comme précédemment évaluer cette méthode grâce au deuxième réseau de neurone entrainé sans les chiffres 2 et 4.







83 % de mauvaise prédictions détectées
17 % de mauvaise prédictions non détectées



89 % de mauvaise prédictions détectées
11 % de mauvaise prédictions non détectées

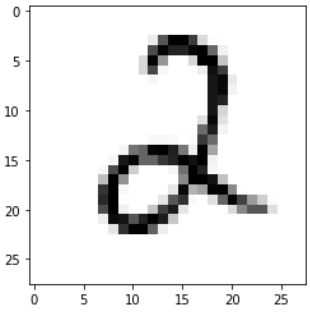
On constate de même que plus le seuil est élevé et proche de 1, plus les mauvaises prédictions sont détectées et mieux elle sont triées. On constate par exemple qu'en abaissant suffisamment le seuil, on parvient a éliminer tous les « faux » 3 et 6 (toujours au dépriment de bonne predictions qui ne sont pas retenues). Mais il faut aussi garder à l'esprit que cette evaluation se fait sur des classes inconnues par notre réseau de neurone, si l'on souhaite uniquement éviter la confusion entre deux classes connues, une valeur seuil semble viable comme vu au point 5.

Comme pour la première méthode, de mauvaises prédictions pourront toujours passer le filtre, mais uniquement si elle appartiennent à une classe inconnue. En choisissant bien notre valeur seuil, une detection quasi systématique des erreurs semble être possible. Dans tous les cas, toutes les valeurs ne sont pas à rejeter puisque les mauvaises predictions restent minoritaires. On pourrait alors imaginer un algorithme qui donne un interval de certitude : fort, moyen, faible, en fonction de valeurs seuil pour les écarts type habillement choisies.

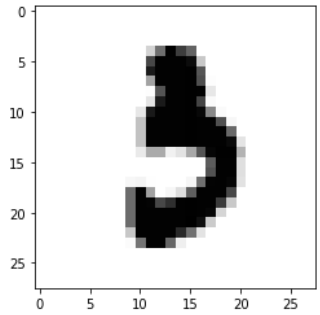
7. Utilisation de la deuxième méthode

Dans cette dernière partie, nous allons appliquer la méthode de Monte-Carlo Dropout au réseau de neurone initial, nous devons d'abord determiner deux valeurs seuil pour delimiter les zones de certitude forte, moyenne et faible. Pour cela nous avons choisi de récupérer touts les écarts types des mauvaises prédictions pour les entrés de test. Puis nous avons choisi la plus faible en tant que premier seuil et la médiane pour second seuil.

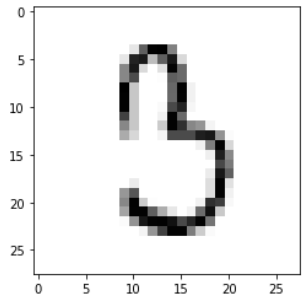
Nous pouvons alors effectuer quelque tests en affichant une image d'entré au hasard dans la partie test de la base de donnée ainsi que la prédiction et la certitude associée :



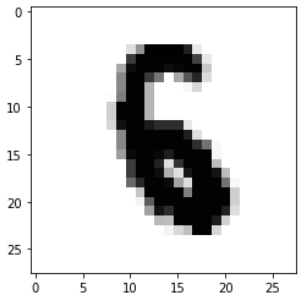
'Le chiffre semble etre un 2 avec une certitude forte'



'Le chiffre semble etre un 3 avec une certitude moyenne'



'Le chiffre semble etre un 5 avec une certitude faible'



Le chiffre semble etre un 5 avec une certitude faible