Université libre de Bruxelles

Projet de fin d'année

Partie 4

Rapport

Auteur:
Mark Diamantino Caribé

Superviseur de la partie 4:
Dr. Charlotte Nachtegael

Abstract

Rapport du projet de fin d'année (partie 4).

1 Introduction

Le projet se focalise sur la compréhension et la réalisation d'un réseau de neurones s'inspirant des réseaux de neurones biologiques et visant à la classification des données. L'action de distribuer et de cataloguer des données par classe résulte fondamentalement de l'aptitude d'un réseau à détecter les différences. La récupération des données, l'entraînement et l'évaluation sont les trois phases dominantes d'une méthode de classification amplement naturelle. Les mêmes sont employés pour la conception d'un réseau artificielle.

1.1 Récupération des données

Le réseau de neurones prévoit une phase d'acquisition passive d'un jeu de données : l'utilisateur doit soumettre ceci au réseau. En considérant que le principe de la classification repose sur l'identification des différences, il est fréquent de soumettre un ensemble d'information ample, diversifiée et génériquement représentante des classes concernées. De la même manière, il est important de normaliser l'ensemble des éléments afin d'effectuer l'apprentissage sur le même format.

1.2 Entraînement

Au début de la phase d'entraînement, les connexions entre les différents neurones sont inexistantes. Afin de procéder avec les calculs sans avoir des résultats inutilisables, l'implémentation prévoit l'initialisation d'un nombre fini de connexions avec des valeurs proches de zéro. La matrice/vecteur des poids est un ensemble de connexions. Dans la deuxième phase de l'entraînement, le *forward pass*, le réseau doit pouvoir effectuer une prédiction sur une classe donnée à l'aide des connexions créées. L'intérêt de cette étape réside dans la possibilité de comparer les classes prédites avec les réelles. En effet, dans la troisième phase, le *backpropagation*, si la prédiction diffère de la classe effective, il faut effectuer une correction des connexions afin que cela ne se reproduise plus.

1.3 Evaluation

En effectuant une prédiction sur tous les éléments du jeu de données, il est possible de mettre en pratique les notions apprises lors de la phase d'entraînement. Cependant, il faut effectuer plus d'un entraînement et d'une évaluation afin d'avoir une mesure globale de l'apprentissage de notre réseau. L'implémentation prévoit une phase de validation croisée appelée «cross-validation» dans laquelle les résultats de plusieurs entraînements et prédictions sont observés pour en établir la moyenne de classes prédites sur la totalité des éléments.

2 Implémentation

Les types de donnés varient en fonction des applications du réseau. Deux exemples d'application fréquents sont la reconnaissance vocale et la reconnaissance des images. Parmi les différents jeux de données proposées, le réseau de neurones réalisé dans cette version du projet vise à la reconnaissance des chiffres et exploite le MNIST (*'Modified National Institute of Standards and Technology dataset'*). Il s'agit d'un l'ensemble homogène de 60 000 vecteurs de 784 composantes dont les valeurs sont comprises entre 0 et 255. Chaque vecteur représente une image de 28x28 pixels d'un chiffre entre 0 et 9 écrit à la main par un des 500 individus participants. La valeur de chaque composante correspond à une couleur dans l'échelle des gris. Le choix de ce jeu de données résulte de son format aisément exploitable et de la régularité du nombre d'éléments par classe. Cette dernière caractéristique implique un apprentissage intelligent et une prédiction équilibrée.

2.1 Lecture et conversion des données

Lire et convertir le jeu de donnée choisi du format originel *'csv'* à des matrices en un temps potentiellement bas constitue un aspect essentiel en considérant la quantité d'informations contenues. Dans l'implémentation effectuée la conversion se fait en utilisant les méthodes standard de python et ensuite en convertissant les vecteurs en format Numpy array. Numpy présente une deuxième solution avec l'utilisation de sa méthode 'genfromtext' qui résulte toutefois trois fois plus lente (Table 1).

Standard	Numpy.genfromtext
10.104 s	31.3099 s
11.0054 s	31.4969 s
9.8683 s	31.3123 s

Table 1: Temps d'exécution des conversions du 'train.csv'.

2.2 Phases d'entraînement

Le réseau de neurones peut être structuré de plusieurs configurations. Deux méthodes sont présentes dans l'énoncé du projet: le multiperceptron-multicouche et le multiclasse-multicouche. Dans la première configuration, la prédiction des classes est effectuée par moyen d'un ensemble de dix sous-réseaux à plusieurs couches dans lequel chacun se focalise sur la reconnaissance des différences entre un chiffre et les restants. Chaque sous-réseau en ayant qu'une seule sortie renvoie une valeur que symbolise l'appartenance à la classe pour laquelle il s'est entrainé. Dans le deuxième cas, un seul réseau de neurones à plusieurs couche est capable d'effectuer également des prédictions en ayant autant de sorties que de classes. L'implémentation choisie dans ce projet est la deuxième, la multiclasse et multicouche. Comme suite aux modifications par rapport à l'énoncé, le forward pass et la backpropagation sont ensuite décrits.

2.2.1 Forward pass

Notons x le vecteur représentante l'image en entrée. Notons d le nombre de couches hormis la couche d'entrée. Numérotons les couches de 0 à d et $\{s_0, s_1, ..., s_d\}$ leurs tailles respectives. Notons pour chaque $i \in \{1, \ldots, d\}$ la matrice des poids $W^{(i)}$ entre les couches *i-1 et i*. La matrice $W^{(i)}$ est une matrice s_{i-1} x s_i [1]. Appelons f la fonction d'activation choisie. Notons $h^{(i)}$ l'application de la fonction d'activation au vecteur $a^{(i)}$ des activations des neurones de la couche i.

$h^{(0)} \leftarrow x$
for i $\in \{1, \ldots, d\}$ **do**
  $a^{(i)} \leftarrow h^{(i-1)} W^{(i)}$ [2]
  $h^{(i)} \leftarrow f(a^{(i)})$
end for

La prédiction sur la classe de l'image x est l'argmax du vecteur sortie $h^{(d)}$. Nous pouvons avoir une mesure formelle de l'erreur à l'aide du codage *'one-hot'* et du carré de la distance euclidienne entre deux vecteurs. Le codage *one-hot* est une représentation de l'étiquette correspondant au chiffre de l'image x. Pour encoder un chiffre en format *one-hot* nous créons un vecteur de la taille des classes, dans lequel toutes les composantes sont zéro sauf celle dont sa position correspond à la classe à prédire, qui vaut 1.
Exemple : *classes = 10*, *étiquette = 5*, *one-hot* encoding $y = \{0,0,0,0,0,1,0,0,0,0\}$. Nous pouvons ensuite calculer le carré de la distance euclidienne entre y et $h^{(d)}$ pour avoir une mesure de l'erreur.

2.2.2 Backpropagation

Afin de mettre en pratique un algorithme de correction pour chaque matrice des poids, il est possible d'effectuer une substruction avec une deuxième matrice de la même dimension. Les matrices en questions sont les gradients $\bigtriangledown W^{(1)}$ et $\bigtriangledown W^{(2)}$.

$g \leftarrow h^{(d)} - y$
for i $\in \{d, \ldots, 1\}$ **do**
  $g \leftarrow g \odot f'(a^{(i)})$ [3]
  $\bigtriangledown W^(i) \leftarrow h^{(i+1)T} \bullet g$ [4]
  $g \leftarrow g \bullet W^{(i)T}$
end for

[1] En utilisant plutôt la notation s_{i+1} x s_i, nous pouvons constater que la taille de la dernière matrice des poids serait de s_{d+1} x s_d . Etant donné que s_d est la taille de la dernière couche s_{d+1} est un nombre inexistant.

[2] Posons $d = 2$ et $s_0 = 784, s_1 = 50, s_2 = 10$ et i = 1, Nous pouvons remarquer qu'il n'est pas possible effectuer le produit matriciel entre $W^{(1)}$ et $h^{(0)}$. La taille de $W^{(1)}$ est de 784x50 et de h est 1x784, ce qui rend impossible le produit matriciel car le nombre de colonnes de la première matrice doit être égale au nombre de de lignes de la deuxièmes.

[3] Produit de Hadamard

[4] Produit matriciel

2.3 Fine-tuning

Dans le cas du réseau en question améliorer le temps d'exécution d'une étape de training et le résultat global de la prédiction sont deux ambitions auxquelles il est possible aspirer par moyen du fine-tuning. L'objectif du fine-tuning est d'expérimenter le modèle réalisé avec des paramètres distincts afin de trouver la combinaison optimale.

2.3.1 Procèdure

Afin d'effectuer une expérience fidèle à la réalité, le training-set et le test-set choisis sont respectivement le *'train_small.csv'* (2000 images) et le *'train_tiny.csv'* (100 images). Les reglages des autres paramètres sont exprimés dans la *Table 2*. Trois différents tests sont effectués afin d'examiner l'influence de trois paramètres: H (neurones de la couche intermédiaire), learning rate (l'importance à accorder à une éventuelle correction des poids lors de la backpropagation) et weight init. sur le temps moyen d'exécution d'une étape de training et sur le score global à la fin de chaque itération du cross-validation.

Paramètre	Valeur
H	50
Learning rate	0,0001
Weight Init.	0,0001
Training epochs	100
Itérations du cross-validations	20
Fonction d'activation	Softsign

Table 2: Valeurs standard utilisées pour les paramètres.

2.3.2 Représentation graphique

En tenant compte de la différence entre les valeurs des paramètres tels que le learning rate et les résultats globaux qui peuvent être 900000 fois plus élevés, il est essentiel de normaliser les données afin de pouvoir les représenter clairement. Par conséquent, toutes les données seront exprimées en fonction de leurs valeurs sur la valeur maximale de chaque paramètre lors des étapes de cross-validation. Dans les graphiques, les valeurs des paramètres, le score et le temps, seront exprimées en fonction des itérations du cross-validation.

2.3.3 H

Dans l'expérience consacrée à l'incidence de H (initialisé à 0), ce dernier est incrémenté de 39 lors de chaque itération du cross-validation afin que sa valeur soit aussi proche que possible du nombre de neurones de la couche d'entrée, c'est-à-dire 784. Dans la *Figure 1*, nous pouvons constater que le meilleur rapport entre global score et average time est obtenu à la 12e itération du cross-validation, au point d'intersection entre *H* et du 'Smallest Loss' (la distance euclidienne moyenne entre les prédictions et les chiffres réels des images). Nous pouvons également remarquer qu'incrémenter le nombre de neurones de la couche intérieure implique un ralentissement causé par l'abondance d'opérations sur les matrices des poids. Néanmoins, le score global accroît modérément jusqu'une valeur fixe: 91,9192%.

C.v. It.	Learning Rate	Weights Init.	H	Train Epochs	Smallest Loss	Average Time	Global Score
1	0,0001	0,0001	39,0000	100,0000	0,5205	0,3508	86,8687
2	0,0001	0,0001	78,0000	100,0000	0,5331	0,3477	86,8687
3	0,0001	0,0001	117,0000	100,0000	0,4572	0,4806	87,8788
4	0,0001	0,0001	156,0000	100,0000	0,4383	0,6248	87,8788
5	0,0001	0,0001	195,0000	100,0000	0,3973	0,6501	87,8788
6	0,0001	0,0001	234,0000	100,0000	0,3809	0,7716	87,8788
7	0,0001	0,0001	273,0000	100,0000	0,3588	1,0074	89,8990
8	0,0001	0,0001	312,0000	100,0000	0,3526	1,2100	87,8788
9	0,0001	0,0001	351,0000	100,0000	0,3434	1,5941	88,8889
10	0,0001	0,0001	390,0000	100,0000	0,3293	1,5757	90,9091
11	0,0001	0,0001	429,0000	100,0000	0,3283	1,6603	87,8788
12	0,0001	0,0001	468,0000	100,0000	0,3174	1,8287	91,9192
13	0,0001	0,0001	507,0000	100,0000	0,3140	2,0354	89,8990
14	0,0001	0,0001	546,0000	100,0000	0,3069	2,2084	89,8990
15	0,0001	0,0001	585,0000	100,0000	0,3030	2,1800	89,8990
16	0,0001	0,0001	624,0000	100,0000	0,2984	1,9379	89,8990
17	0,0001	0,0001	663,0000	100,0000	0,2931	3,8268	90,9091
18	0,0001	0,0001	702,0000	100,0000	0,2885	2,0858	90,9091
19	0,0001	0,0001	741,0000	100,0000	0,2874	4,6949	91,9192
20	0,0001	0,0001	780,0000	100,0000	0,2825	2,4061	91,9192

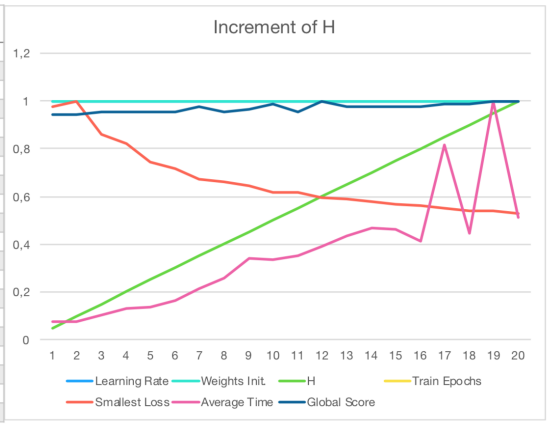

Figure 1: Fine-tuning de H.

2.3.4 Weights Init.

La valeur utilisée dans l'initialisation des matrices des poids (initialisée à 0) est incrémentée de 0,0005 jusqu'à 0,01. Nous pouvons remarquer dans la *Figure 2* qu'incrémenter Weights Init. contribue à une imperceptible diminution du score global et un accroissement du temps d'exécution. À la troisième itération du cross-validation, le rapport entre *Global Score* et *Average Time* est optimal.

C.v. It.	Learning Rate	Weights Init.	H	Train Epochs	Smallest Loss	Average Time	Global Score
1	0,0001	0,0005	50,0000	100,0000	0,5105	0,7153	88,8889
2	0,0001	0,0010	50,0000	100,0000	0,4914	0,6323	86,8687
3	0,0001	0,0015	50,0000	100,0000	0,4833	0,6356	89,8990
4	0,0001	0,0020	50,0000	100,0000	0,4501	0,6358	85,8586
5	0,0001	0,0025	50,0000	100,0000	0,4627	0,6256	86,8687
6	0,0001	0,0030	50,0000	100,0000	0,4491	0,6147	88,8889
7	0,0001	0,0035	50,0000	100,0000	0,4466	0,6079	87,8788
8	0,0001	0,0040	50,0000	100,0000	0,4400	0,6192	85,8586
9	0,0001	0,0045	50,0000	100,0000	0,4510	0,6412	89,8990
10	0,0001	0,0050	50,0000	100,0000	0,4550	0,6479	88,8889
11	0,0001	0,0055	50,0000	100,0000	0,4567	0,6391	87,8788
12	0,0001	0,0060	50,0000	100,0000	0,4769	0,6482	88,8889
13	0,0001	0,0065	50,0000	100,0000	0,4711	0,6532	82,8283
14	0,0001	0,0070	50,0000	100,0000	0,4830	0,6517	87,8788
15	0,0001	0,0075	50,0000	100,0000	0,4851	0,6416	88,8889
16	0,0001	0,0080	50,0000	100,0000	0,4886	0,6725	85,8586
17	0,0001	0,0085	50,0000	100,0000	0,4907	0,6571	84,8485
18	0,0001	0,0090	50,0000	100,0000	0,5191	0,6327	83,8384
19	0,0001	0,0095	50,0000	100,0000	0,5120	0,6261	80,8081
20	0,0001	0,0100	50,0000	100,0000	0,5328	0,6242	86,8687

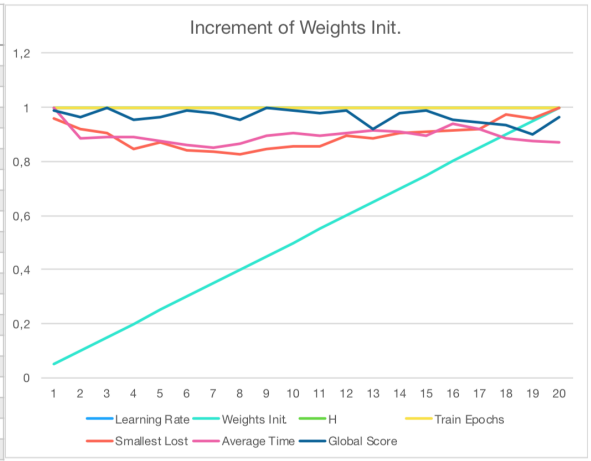

Figure 2: Fine-tuning des initialisations des poids.

2.3.5 Learning rate

Le *learning rate* est initialisé à 0 et ensuite incrémenté de 0,0005 jusqu'à 0,01. Dans la *Figure 3*, nous pouvons remarquer une considérable incidence du *learning rate* sur toutes les variables sauf le *Average Time*. À la troisième itération du cross-validation, le rapport entre *Global Score* et *Average Time* est optimal.

C.v. It.	Learning Rate	Weights Init.	H	Train Epochs	Smallest Loss	Average Time	Global Score
1	0,0005	0,0001	50,0000	100,0000	0,3554	0,3865	86,8687
2	0,0010	0,0001	50,0000	100,0000	0,3468	0,3862	90,9091
3	0,0015	0,0001	50,0000	100,0000	0,3484	0,3760	85,8586
4	0,0020	0,0001	50,0000	100,0000	0,4191	0,3836	88,8889
5	0,0025	0,0001	50,0000	100,0000	0,4206	0,3750	80,8081
6	0,0030	0,0001	50,0000	100,0000	0,4218	0,3876	77,7778
7	0,0035	0,0001	50,0000	100,0000	0,4024	0,3821	82,8283
8	0,0040	0,0001	50,0000	100,0000	0,5783	0,3836	72,7273
9	0,0045	0,0001	50,0000	100,0000	0,4157	0,3804	82,8283
10	0,0050	0,0001	50,0000	100,0000	0,6258	0,3771	77,7778
11	0,0055	0,0001	50,0000	100,0000	0,4476	0,3850	79,7980
12	0,0060	0,0001	50,0000	100,0000	0,5914	0,3750	68,6869
13	0,0065	0,0001	50,0000	100,0000	0,4600	0,3880	82,8283
14	0,0070	0,0001	50,0000	100,0000	0,4604	0,3762	79,7980
15	0,0075	0,0001	50,0000	100,0000	0,4370	0,3801	70,7071
16	0,0080	0,0001	50,0000	100,0000	0,4681	0,3929	80,8081
17	0,0085	0,0001	50,0000	100,0000	0,6820	0,3776	63,6364
18	0,0090	0,0001	50,0000	100,0000	0,6866	0,3764	72,7273
19	0,0090	0,0001	50,0000	100,0000	0,6754	0,3812	66,6667
20	0,0100	0,0001	50,0000	100,0000	0,3554	0,3865	86,8687

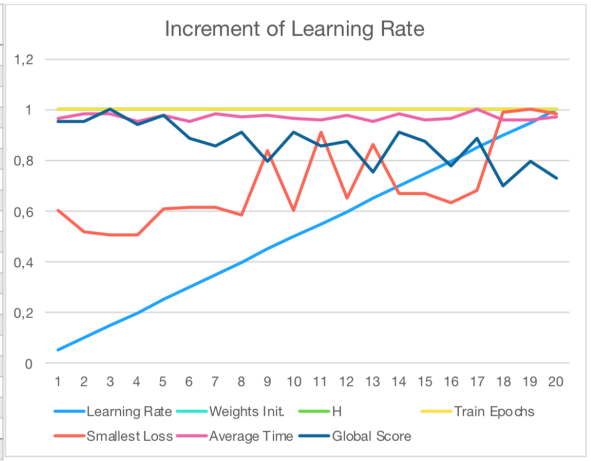

Figure 3: Fine-tuning des initialisations des poids.

2.3.6 Résultats

En considérant les analyses précédentes, nous pouvons conclure que pour avoir un résultat considérable, il n'est pas conseillé d'augmenter le learning rate et le *weights init.* Pour ce qui concerne H, il est possible d'obtenir des scores considérables et un temps d'exécution optimal si H est au tour de 0,6 * nombre des neurones de la couche d'entrée. En utilisant ces concepts et donc les paramètres suivants: $H = 468$, *Learning Rate* = 0,0015 et *Weights Init.* = 0,003; 90,90 est le résultat de trois tests, il s'agit d'un score optimal par rapport à la moyenne.

3 Interface graphique

L'objectif principal de l'interface graphique est de mettre en relation l'utilisateur avec un outil puissant et aussi relativement familier tel que le réseau de neurones. Ceci est effectué en mettant en évidence une affinité avec notre réseau biologique lors de la lecture, la reconnaissance optique de chiffres (OCR). Deux autres outils sont compris dans l'interface graphique afin de bénéficier d'une vue d'ensemble: la modalité 'Standard' et la section 'Tools' dédiée aux conversions entre image et vecteur format MNIST.

3.1 Standard

La modalité *Standard* permet à l'utilisateur d'effectuer des validations croisées en visualisant les résultats. L'affichage des informations telles que l'itération de l'entraînement, du cross-validation, la 'perte' (*loss*) moyenne pour chaque entraînement et le résultat global est accompli dans la partie gauche de l'interface dans l'onglet *Standard*. Les mêmes informations sont affichées également dans le terminal.
Toute information peut être enregistrée dans un fichier TEXT (*.txt*); il faut marquer d'une coche *Do you want to save logs ?* avant de démarrer la validation croisée. Le fichier *'Logs.txt'* se trouvera à l'intérieur du même dossier. La matrice de confusion sera affichée dans la partie droite de l'interface dans l'onglet *Standard*.
En plus des paramètres standards, dans la partie supérieure il est possible de choisir un autre dataset comme test-set, choisir une des quatre différentes fonctions d'activation et enregistrer ou utiliser des matrices de poids précédemment enregistrées. Pour l'enregistrement des poids, il faut marquer d'une coche *Save Weights ?*. À la fin de la validation croisée, les poids seront exportés en format *Comma-separated values* (*.csv*) dans deux fichiers distincts: *'W1.csv'* et *'W2.csv'*. Pour les utiliser, il faut cliquer sur *Load Weights* et ensuite cocher *Use them?*. Cette option a été développée afin de faciliter les opérations d'OCR.

3.2 OCR

La modalité OCR permet de reconnaître des chiffres écrits à la main. Afin d'utiliser cette option, en prémisse, il est nécessaire d'écrire sur une feuille de papier blanc des chiffres avec un stylo noir ou à encre de couleur sombre. Il faut ensuite effectuer une numérisation et couper l'image afin qu'il n'y ait pas plus d'une ligne de chiffre par image. Cette contrainte dérive de difficulté de représenter la séquence de manière ordonnée du haut vers le bas. En effet, afin d'afficher les chiffres avec le même l'ordre d'écriture de gauche à droite, l'algorithme trie les éléments en fonctions des coordonnés sur la même ligne. Par la suite, il est nécessaire de charger la photo par moyen de l'option 'Load Picture', taper le nombre de chiffres présents dans l'image et choisir une des deux modalités OCR: *Automatic* (afin d'effectuer la reconnaissance de plusieurs chiffres), *Manual* pour travailler avec une image ayant déjà en encadrement proche du MNIST. Les éventuels poids en format *CSV* seront utilisés s'ils ont été chargés et activés correctement comme dans la procédure décrite dans la sousection 3.1. Cliquer sur *Start OCR* permet de démarrer la reconnaissance optique, la séquence reconnue sera affichée dans le quadrant inférieur. Afin d'expérimenter cette modalité, un couple de poids et deux images sont inclus dans le dossier.

3.3 Tools

La section *Tools* La section *Tools* permet d'effectuer deux conversions: d'un vecteur en format MNIST (784 composantes) à image en niveaux de gris et vice versa. Dans le premier cas, l'utilisateur doit écrire le vecteur comme une liste de 784 éléments dans la subdivision gauche de la section *Tools*, il est nécessaire de ne pas oublier les crochets. Après avoir cliqué sur *CONVERT to greyscale image*, l'image sera affichée dans la partie droite. Dans le deuxième cas, l'utilisateur doit charger l'image en s'assurant qu'elle est carrée ou avec un format d'image proche du 1:1. Après avoir cliqué sur *CONVERT to MNIST format*, le vecteur correspondant sera affiché dans la subdivision gauche.

4 Conclusion

Initialement, nous avons eu l'occasion d'effectuer des liaisons élémentaires avec le réseau de neurones biologique à travers des phases d'acquisition ds données, entraînement et évaluation. Les mêmes étapes ont été traitées dans la deuxième section, dédiée à l'implémentation.
Nous avons pu constater qu'il y a plusieurs structures possibles pour un réseau de neurones: multiperceptron-multicouche et le multiclasse-multicouche, et que même si dans la structure implémentée nous n'avons qu'un seul et unique réseau au lieu de dix perceptrons, il suffit d'avoir plusieurs sorties pour réaliser un réseau très performant.
Par moyen d'un algorithme d'automatisation du processus appelé fine-tuning, nous avons découvert que incrémenter le nombre de neurones de la couche intermédiaire est une solution valide pour obtenir des résultats optimaux. Toutefois, à cause du temps d'exécution et des ressources consommées, la valeur de H ne devrait pas dépasser *0,6*nombre des neurones de la couche d'entrée.*
Nous avons également implémenté une des nombreuses applications réelles du machine learning, le OCR.
Malgré les limitations liées à l'individuation des chiffres dans une image, à l'encadrement et au rapport chiffre/image que doit être proche de la moyenne des rapports dans le dataset MNIST, la prédiction OCR résulte être meilleur que le pourcentage de reconnaissance aléatoire.
Afin d'améliorer cette application, il faudrait ajouter plusieurs couches internes (au lieu d'incrémenter H) et expérimenter les différentes méthodes pour une lecture des chiffres sur plusieurs lignes en maintenant l'ordre d'écriture.