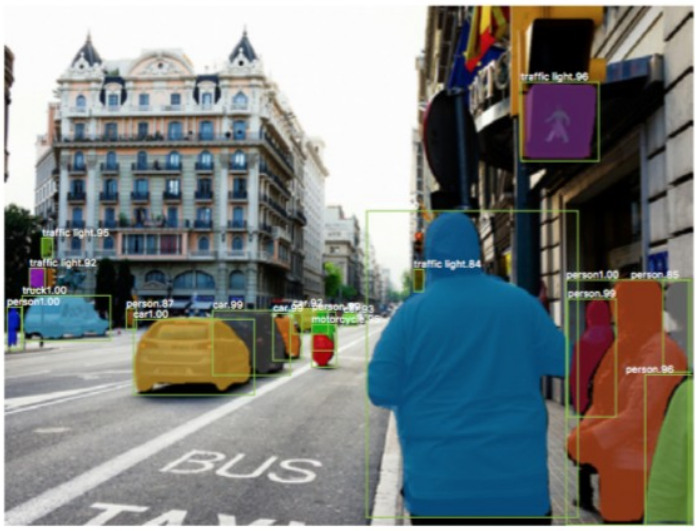
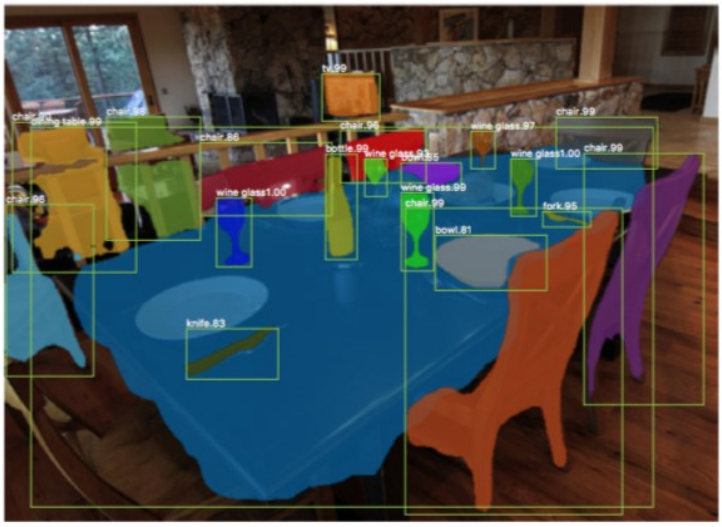
YUMAIN
Sensing & Predictive AI
www.yumain.fr

Systèmes Intelligents - IA TP
Afficheur 7 segments



Un exemple de réseau de neurones: Mask-RCNN. Il est constitué de plus de 90 millions de units avec leurs poids, et est un assemblage de plusieurs MLP et CNN. Il est capable de détecter des objets et de tracer leurs contours au sein d'une image. Il nécessite environ 10Go de RAM GPU pour un fonctionnement par batch de 1 image. Toutefois, même si c'est cool, ça n'a rien à voir avec le TP.

14H rue Pierre de Coubertin
21000 Dijon
FRANCE

Tel : +33 380 371 795

23, rue Paul Bert
92100 Boulogne Billancourt
FRANCE

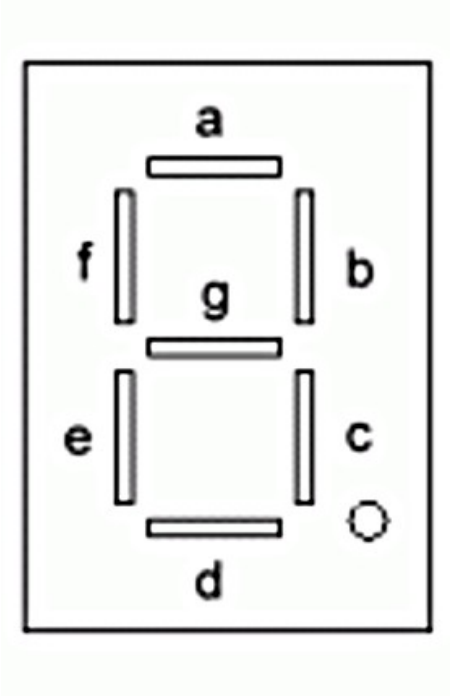
Tel : +33 146 032 660

Vous avez jusque là vu comment lire un fichier CSV, *caster* dans différents types en Python, écrire des classes, des fonctions et des méthodes, et avez presque terminé de développer votre premier perceptron assez générique pour résoudre différents problèmes assez simples.

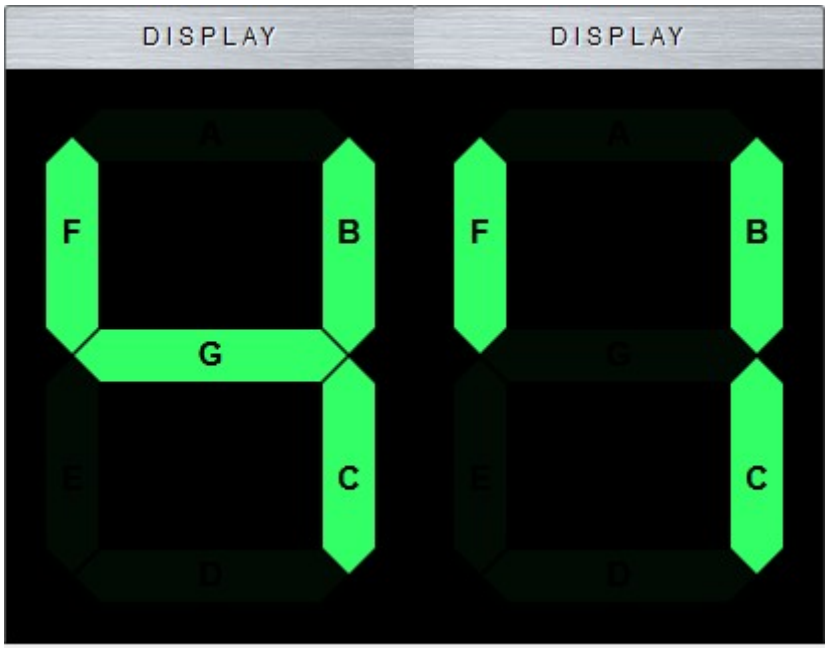
Afin de terminer votre classe perceptron, vous allez devoir y le calcul de la convergence:

1. La convergence, c'est le fait de stopper l'apprentissage dès que le réseau de neurones semble ne plus faire de progrès dans son apprentissage. Autrement dit, regardez la valeur d'erreur totale epoch après epoch. Si la valeur d'erreur ne change pas suffisamment (à vous de décider ce que "suffisamment" veut dire), alors cela signifie que vous avez atteint un point de convergence.

Vous allez désormais travailler sur le problème de l'afficheur 7 segments. Je vous conseille FORTEMENT de lire tout l'énoncé avant d'entreprendre quoi que ce soit.



Vous avez à votre disposition, dans votre imaginaire tout du moins, un afficheur 7 segment, dont chaque segment peut être alimenté afin de s'allumer. Nous souhaitons faire en sorte que, lorsque certains segments sont allumés, une IA soit capable de déterminer le chiffre qui est censé s'afficher. La raison pour laquelle cela *pourrait* être intéressant est que, comme les réseaux de neurones sont capables de généraliser ce qu'ils apprennent, un segment qui viendrait à ne pas s'allumer ne nous poserait pas de problème pour détecter le chiffre que l'on souhaitait afficher. Aux "yeux" de l'IA, les deux affichages ci-dessous sont des 4.



Première étape: sans prendre en compte le 0, cherchez un moyen de représenter votre modèle de données. Comment les inputs et outputs sont-ils formés ? Combien d'entrées possédera le perceptron ? Combien de sorties doit-il comprendre ?

Deuxième étape: Manuellement, stockez vos données d'inputs et vos données teacher dans un fichier CSV. Votre programme viendra le charger en mémoire quand le moment sera venu.

Troisième étape: C'est l'heure d'apprendre à utiliser Keras. Nous en avons rapidement parlé en cours ; il s'agit d'un framework français d'intelligence artificielle très réputé, au point qu'il fait désormais parti intégrante du workflow de Tensorflow (le framework IA de Google). A l'aide de Keras , faites un apprentissage de ce problème à l'aide d'un perceptron.

Un peu d'aide ? Rendez-vous sur ce lien !

La gestion des inputs/outputs va très certainement vous poser problème. Pour commencer, votre tableau d'input sera certainement de taille (9, 7) car vous avez 7 segments et 9 chiffres. Cela ne va pas convenir à Keras pour notre problème, car lui va s'attendre à un tableau de taille (9, 1, 7) ! Pour corriger le tir, vous devrez utiliser la méthode *numpy.reshape()*.

Vos outputs, eux, doivent très certainement être les chiffres de 1 à 9. C'est très bien en soit, mais pas optimisé pour notre problème. Pour ce problème, il vaut mieux avoir une traduction de nos outputs en one-hot vectors. Utilisez la méthode *keras.utils.to_categorical()* sur votre tableau de sortie et regardez à quoi la valeur retournée ressemble. C'est ce format que l'on utilisera.

Donc, pour récapituler, vous devrez faire :

inputs = numpy.reshape(inputs, (9, 1, 7))
outputs = keras.utils.to_categorical(outputs)

Pour la couche de sortie, utilisez une activation de type « softmax ». Cela va faire que chaque sortie de votre réseau sera en fait un pourcentage de confiance entre 0 et 1. Le neurone de sortie possédant la plus grande confiance sur une prédiction est le neurone doit afficher la valeur 1. Par exemple, si j'ai 4

neurones en sortie et que les valeurs sont, dans l'ordre : 0,05 – 0,8 – 0,05 – 0,1 alors je sais que la valeur de mon neurone de sortie numéro 2 doit être 1, et que les autres neurones doivent afficher 0. Ma prédiction devient alors non pas 0,05 – 0,8 – 0,05 – 0,1 mais 0 – 1 – 0 – 0.

Quatrième étape: Récupérez vos inputs et outputs correspondant via le code en ouvrant les fichiers CSV que vous avez créé manuellement. Jouez avec les paramètres de votre perceptron jusqu'à obtenir au moins 6 prédictions correctes. Créez un input erroné, et testez de faire un prédiction dessus. Votre prédiction est-elle cohérente ?

Cinquième étape: Remplissez bien votre compte-rendu de TD/TP. Comment avez-vous mis en place vos données, pourquoi un tel choix de paramètres, que faire pour améliorer ce dernier réseau de neurones, etc. Soyez concis.

~~Pour rappel, le rapport est à rendre pour le 20 décembre au soir, à 23h59 au maximum. Il y aura -4 points par jour de retard.~~ La date de rendu est à voir avec Manon Ansart.

14H rue Pierre de Coubertin
21000 Dijon
FRANCE
Tel : +33 380 371 795

23, rue Paul Bert
92100 Boulogne Billancourt
FRANCE
Tel : +33 146 032 660