Vision numérique - Labo 7

adrien.lescourt@hesge.ch, tarik.garidi@hesge.ch

2021-2022

OCR

L'objectif de ce projet est de réaliser une application permettant d'utiliser un réseau de neurone pour créer un logiciel de reconnaissance de caractères (OCR). Pour simplifier l'apprentissage, seule les chiffres seront considérés.

L'entrée du système est le dessin de l'utilisateur à la souris pour représenter le chiffre sur une grille de 20x20 pixels, la sortie est la détection de l'OCR.

Architecture

En ingénierie logiciel, il est important de minimiser le couplage entre les différents composants d'une application. L'OCR doit être composé d'une frontend et d'un backend disctinct. La communication entre les deux devra être réalisée en HTTP, via une `API REST`.

Input utilisateur

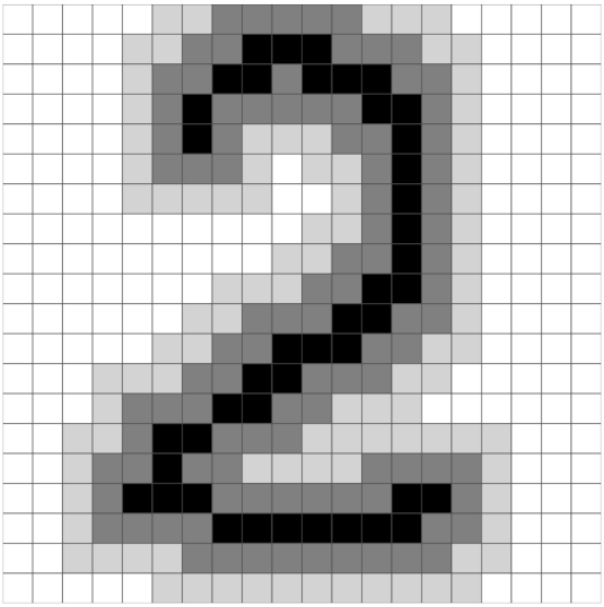

Figure 1: Input

Output du système

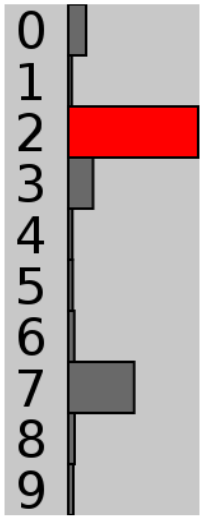

Figure 2: Output

Travail à accomplir

Votre application contiendra les 3 composants suivants:

Frontent web permettant

- De dessiner des caractères **à la souris** via un canvas html5
- D'entrainer le réseau en envoyant des caractères labellisés au backend. Vous devez utiliser votre propre dataset.
- De reconnaitre un caractère en utilisant le backend préalablement entrainé

Interface de communication

- API HTTP basée sur la libraire python fastAPI
- Comporte toutes les routes nécessaires à l'entrainement du réseau, ainsi qu'à son utilisation

Backend

- Utilise l'API `keras` de tensorflow pour le réseau de neurones
- Un réseau neuronal non convolutif, comportant entre 1 et 3 couches cachées
- Fournit une interface python permettant l'entrainement et l'utilisation du réseau

Rendu

Ce travail doit être réalisé en binome. Il doit être effectué sur gitedu et être terminé avant le dimanche 19 juin à minuit. Un dépôt git dans un groupe gitlab dédié vous sera affecté à chacun.

Réseau de neurones

Les réseaux de neurones artificiels constituent un modèle de calcul inspiré du fonctionnement des neurones biologiques. Dans ce modèle, un réseau possède un certain nombre d'entrées et doit déterminer l'état de ses sorties en fonction de ses entrées. Pour ce faire, on utilise plusieurs couches de neurones, chaque neurone d'une couche étant connecté par des synapses à tous les neurones de la couche suivante. Chaque neurone additionne alors toutes ses entrées (la valeur des synapses en entrée) et injecte cette somme dans une fonction de combinaison non linéaire avant de transmettre la valeur calculée à la couche suivante. Les couches d'entrée et de sortie peuvent elles être linéaires ou non.

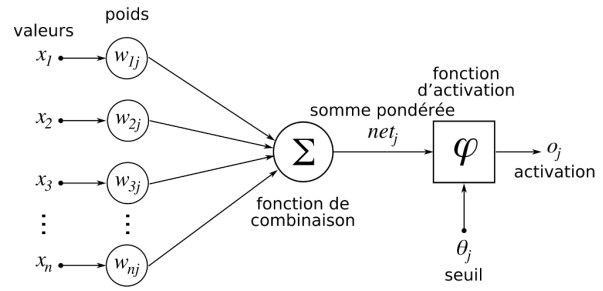

Figure 3: Structure d'un neurone

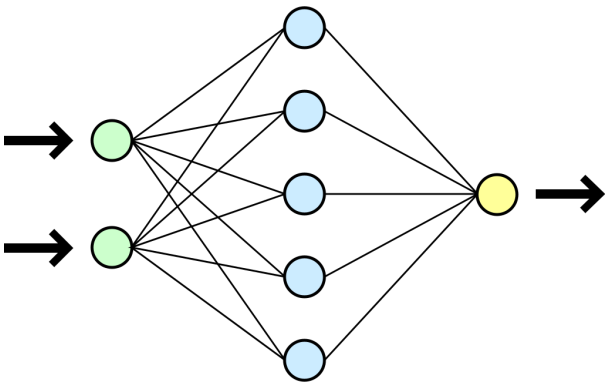

Figure 4: Réseau neuronal simple à trois couches

La difficulté dans ce genre d'algorithme est de calculer la pondération des synapses de sorte que le réseau fournisse la sortie attendue en fonction de l'entrée. Pour cela, une phase d'apprentissage est nécessaire. Durant cette phase, des données sont fournies en entrée, puis la sortie est comparée avec la sortie attendue et les poids des synapses sont modifiés en fonction des erreurs du réseau. Pour cette phase d'apprentissage, il est possible d'utiliser un algorithme de rétropropagation du gradient de l'erreur. L'idée de cet algorithme est de corriger les poids synaptiques en fonction de l'erreur en sortie. Un synapse qui contribue fortement à une erreur se verra fortement modifié. En appliquant itérativement cette technique sur un réseau, on espère atteindre une configuration correcte du réseau. A chaque itération, une fonction de coût représente la qualité du réseau.