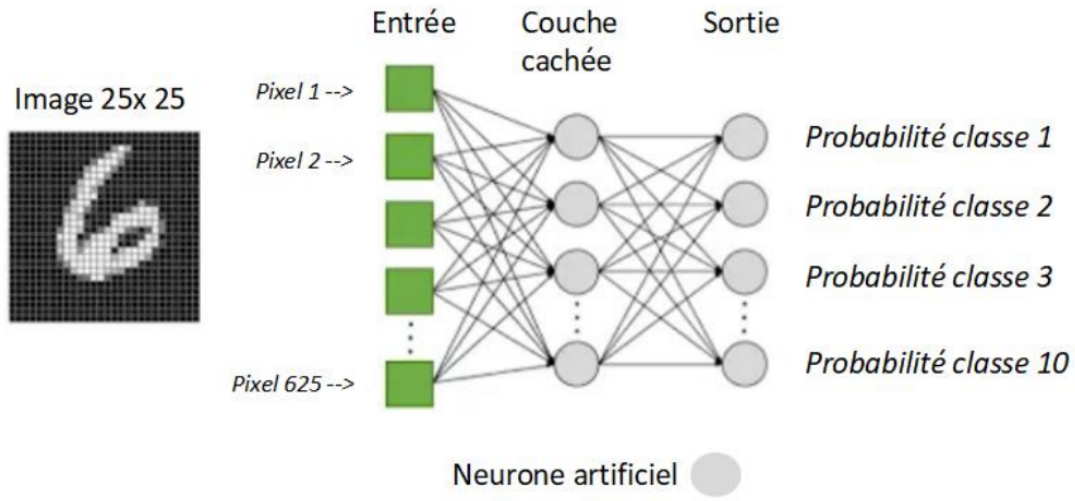
Reconnaissance Optique des Caractères (OCR)



2

Date: Juin 2023

Auteur : Ahmadi Zabiullah

Mail: zabiullah.ahmadi@etu.hesge.ch

cours: Vision numérique

Introduction:

L'objectif de ce rapport est de présenter en détail le travail pratique de détection de chiffres en utilisant la technologie des réseaux de neurones et son application dans le domaine de la reconnaissance optique des caractères (OCR). Ce travail pratique est étroitement lié à notre cours de mathématiques, en particulier à la vision numérique, qui privilégie l'étude des techniques de traitement des images pour la reconnaissance et l'analyse de contenu visuel.

Dans le cadre de ce travail pratique, j'ai essaye de mettre en pratique les connaissances acquises dans notre cours de vision numérique pour développer une application OCR basée sur des réseaux de neurones. j'explorai différentes techniques de traitement d'images pour prétraiter les chiffres dessinés par l'utilisateur, afin d'optimiser la précision de détection. Ces techniques incluent le seuillage, la détection de contours, la manipulation de matrices, ainsi que des méthodes plus avancées telles que le centrage et le redimensionnement des chiffres.

Organisation du projet:

Le projet est organisé en utilisant une architecture basée sur une interface de communication, avec un frontend en HTML et JavaScript, et un backend développé avec le framework FastAPI en Python. Cette approche permet de séparer clairement les responsabilités entre les différents composants et facilite la communication entre eux.

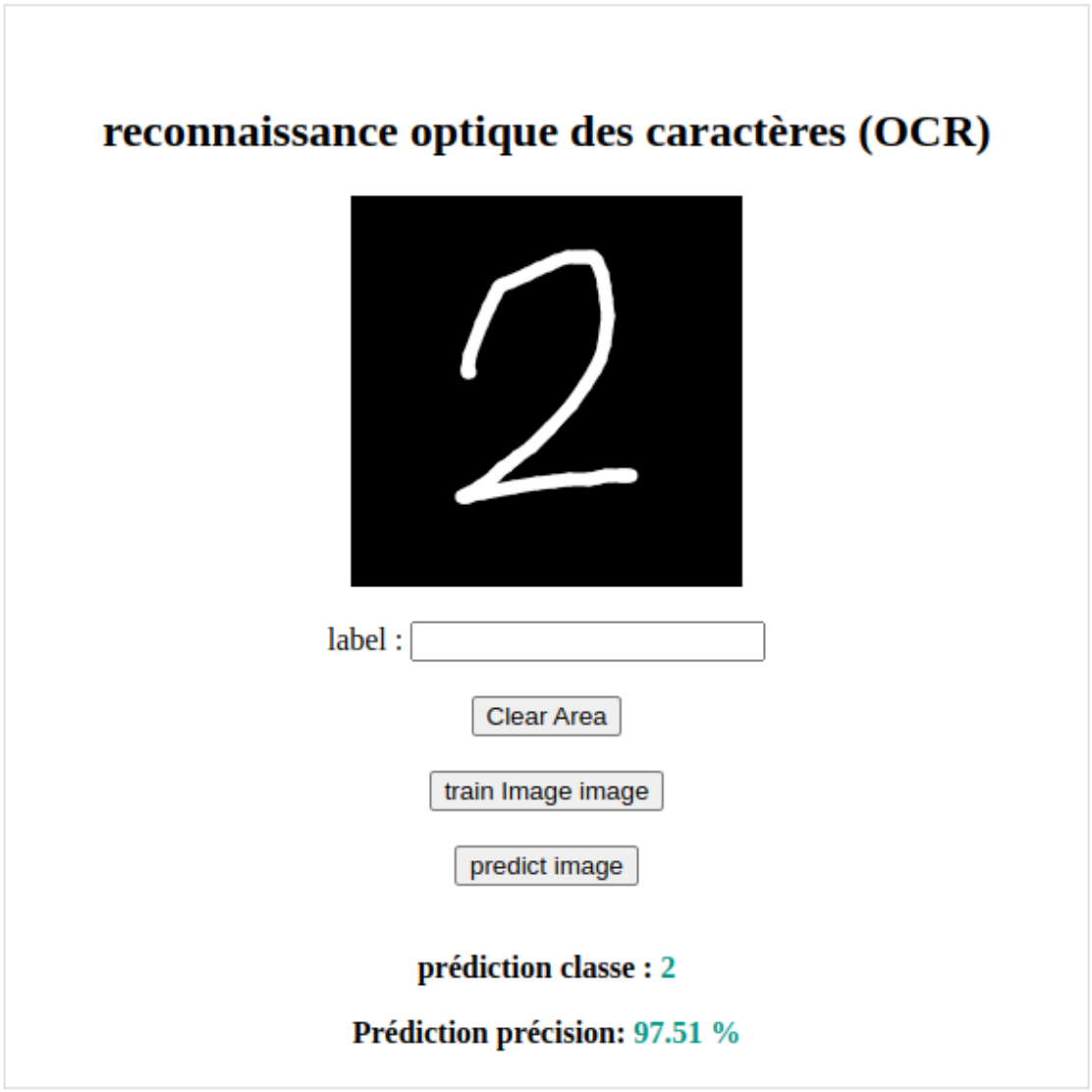
- **Frontend:**

  Le frontend de l'application est implémenté en utilisant HTML et JavaScript. Il offre à l'utilisateur une interface graphique pour dessiner des chiffres à la souris sur un canvas HTML5.

  Le code JavaScript gère les événements de la souris et utilise les API du canvas pour dessiner et interagir avec l'utilisateur. De plus, le frontend utilise les principes de l'architecture **REST** pour communiquer avec le backend via des appels d'API. Deux appels d'API sont effectués depuis le frontend.

  - **sendImage()** : prédire le chiffre dessiné
  - **trainImage()** entraîner le modèle avec la nouvelle Image.

l'interface graphique



- **Backend:**

Le backend de l'application est développé avec le framework FastAPI en Python. Il fournit deux routes principales qui écoutent les requêtes provenant du frontend. La première route détecte l'image envoyée depuis le frontend et prédit le chiffre correspondant en utilisant le modèle de réseau de neurones préalablement entraîné. La deuxième route est utilisée pour entraîner le modèle avec la nouvelle image dessinée par l'utilisateur. Le backend effectue également un prétraitement sur l'image, en centrant le contenu du chiffre (digit) pour améliorer les performances de détection.

- **Prédire l'image**

```
@app.post("/predict")
```

- **Entraîner le modèle**

```
@app.post("/train")
```

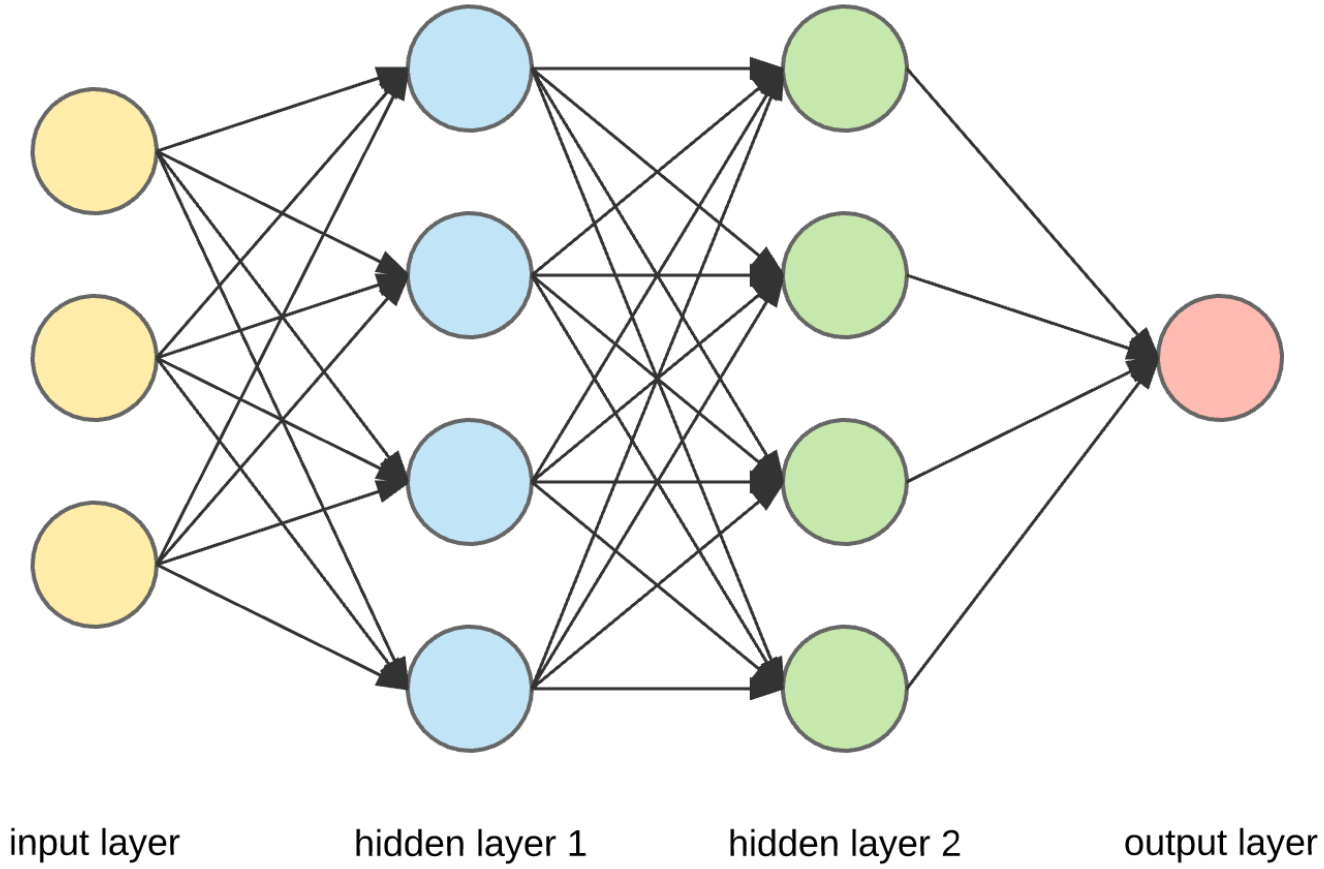
Entraînement du modèle

Dans le processus de l'entrainement du model les images labellisées pour chaque chiffre sont chargées à partir du répertoire **"static/data"**. Chaque image est ensuite redimensionnée à une taille de 20x20 pixels afin d'obtenir une taille cohérente des données d'entrée pour le modèle de réseau de neurones.

Pour faciliter l'entraînement du modèle, les valeurs des pixels des images sont normalisées entre 0 et 1. Cette étape consiste à diviser chaque valeur de pixel par 255, qui est la valeur maximale possible pour un pixel codé sur 8 bits. Ainsi, les valeurs des pixels sont mises à l'échelle dans la plage de 0 à 1, ce qui facilite la convergence de l'entraînement du modèle.

Ensuite, les données sont séparées en ensembles d'entraînement(**80%**) et de test(**20%**). Cela permet d'évaluer les performances du modèle sur des données indépendantes qui n'ont pas été utilisées lors de l'entraînement.

Une fois les données séparées, le modèle de réseau de neurones est créé. Dans ce cas, un modèle séquentiel est utilisé, ce qui signifie que les couches sont empilées les unes sur les autres de manière séquentielle. Le modèle comprend une couche d'aplatissement (Flatten) pour convertir les images 2D en un vecteur 1D, suivi de deux couches cachées (hidden layers) avec une activation ReLU (Rectified Linear Unit), qui permet d'introduire de la non-linéarité dans le modèle. Enfin, une couche de sortie avec une activation softmax est ajoutée pour obtenir des probabilités pour chaque classe de chiffre.



Le modèle est compilé en spécifiant la fonction de perte "categorical_crossentropy", qui est appropriée pour les problèmes de classification multiclasse, l'optimiseur Adam et la métrique d'évaluation de l'exactitude (accuracy). L'optimiseur Adam ajuste les poids du modèle pendant l'entraînement pour minimiser la fonction de perte et améliorer les performances.

Le modèle est ensuite entraîné sur les données d'entraînement en utilisant la méthode **model.fit**. Les paramètres d'entraînement tels que le nombre d'époques (epochs), qui représente le nombre de fois où le modèle parcourt l'ensemble de données, la taille des lots (batch size), qui détermine le nombre d'échantillons utilisés pour mettre à jour les poids du modèle, et les données de validation sont spécifiés. Pendant l'entraînement, le modèle ajuste les poids itérativement afin de minimiser la fonction de perte et améliorer les performances de prédiction.

difficulté rencontré

L'une des difficultés rencontrées lors du développement de l'application a été le prétraitement de l'image. Le backend doit extraire le chiffre dessiné par l'utilisateur et le centrer pour obtenir une meilleure précision de détection. Cela implique de détecter les contours du chiffre, de calculer la position du chiffre dans l'image et de le centrer sur un fond blanc. Plusieurs techniques de traitement d'images, telles que le normalisation, la détection de contours et la manipulation de matrices, sont utilisées pour réaliser cette tâche.

Qeulques examples

Préditions

reconnaissance optique des caractères (OCR)

label :

Clear Area

train Image image

predict image

prédiction classe : 5

Prédiction précision: 99.37 %

reconnaissance optique des caractères (OCR)

label :

Clear Area

train Image image

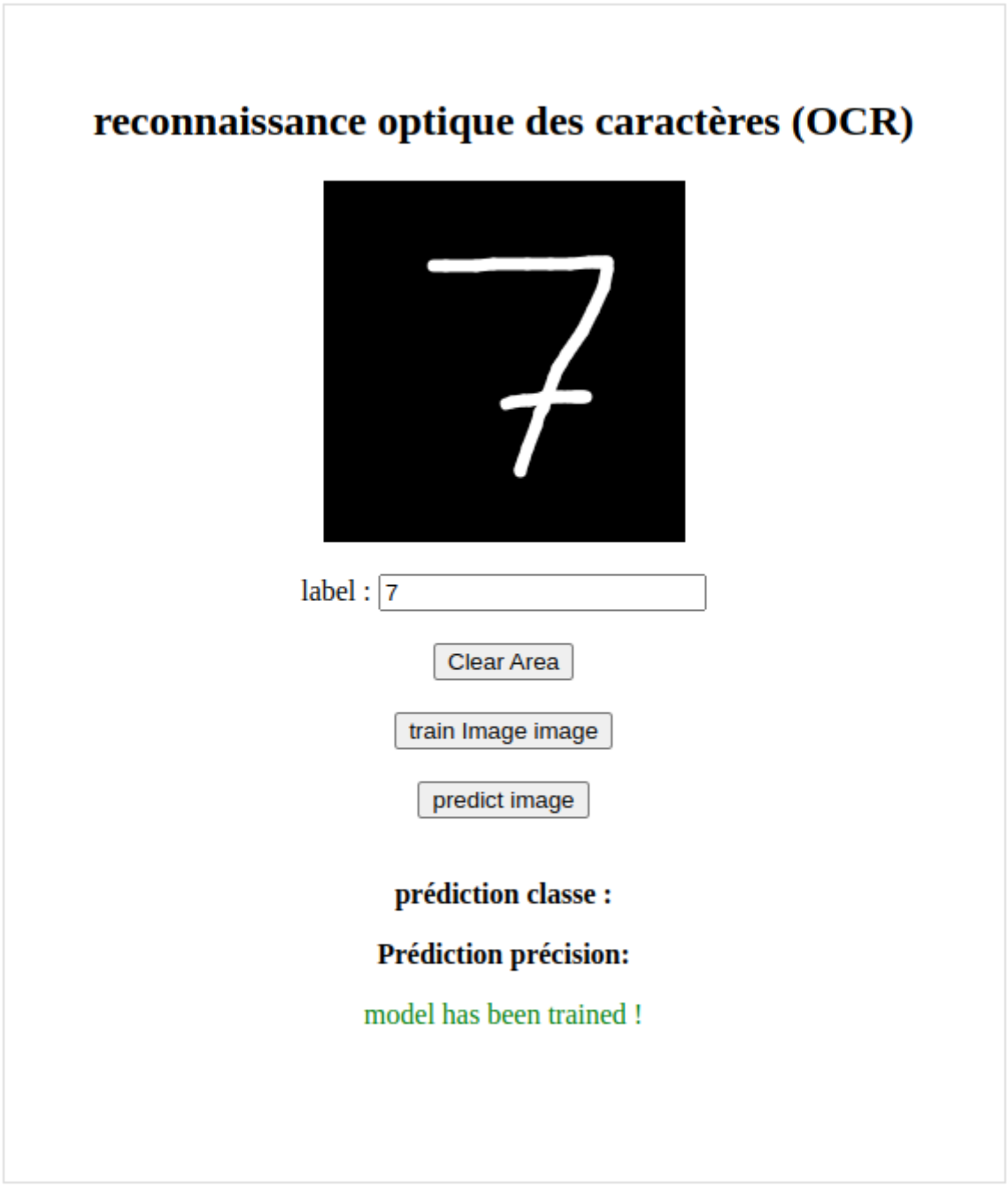
predict image

prédiction classe : -1

Prédiction précision: digit not recognized

-

Entraînement



-

Conclusion:

Ce projet de détection de chiffres en utilisant des réseaux de neurones pour la reconnaissance optique des caractères (OCR) a été une expérience enrichissante. L'organisation du projet en utilisant une interface de communication distincte entre le frontend et le backend a permis de séparer les responsabilités et de faciliter la communication entre les composants. Le prétraitement de l'image, en particulier le centrage du contenu du chiffre, a été une étape clé pour améliorer les performances de détection. Ce projet a mis en évidence le potentiel des réseaux de neurones dans la résolution de problèmes complexes tels que la reconnaissance des caractères manuscrits, et il offre des perspectives intéressantes pour de futures améliorations et développements dans ce domaine.