Licence 3 informatique
2023-2024

Rapport de stage

Projet de fin de licence

Université d'Angers

Encadré par :

Sommaire

01

Présentation du projet

1.1 Présentation du jeu

Le Sudoku est un jeu de logique très populaire basé sur une grille de 9x9 cases, divisée en 9 régions de 3x3 cases. L'objectif est de remplir la grille avec des chiffres de 1 à 9, de telle sorte que chaque chiffre n'apparaisse qu'une seule fois dans chaque ligne, colonne et région.Règles du Sudoku

- La grille est composée de 9 lignes, 9 colonnes et 9 régions de 3x3 cases.
- Certaines cases sont pré-remplies avec des chiffres au début du jeu.
- Le but est de remplir les cases vides avec des chiffres de 1 à 9, sans répéter aucun chiffre dans une même ligne, colonne ou région.

1.2 Notre projet

Dans ce projet, nous explorons l'application des techniques de machine learning pour automatiser la résolution des grilles de Sudoku. Notre objectif est de développer un modèle capable de résoudre des Sudoku de manière rapide et précise, surpassant les méthodes traditionnelles et offrant une solution élégante à ce problème classique.

Fonctionnement

1

Importation ou capture photo d'une image de Sudoku

Reconnaissance

Reconnaissance des chiffres écrits dans l'image de sudoku.

Construction

Construction d'une matrice en utilisant les chiffres reconnus par le modèle de reconnaissance et en mettant les cases vides à 0

Transmission

Transmission de la matrice au modèle de résolution

Résolution

Résolution de la grille avec un réseau de neurones qui prend en entrée la grille incomplète et retourne une grille complète

Écriture

Écriture des prédictions retournées par le modèle de résolution sur notre image de départ.

Affichage

Affichege de l'image du sudo

Le schéma ci-dessus illustre la décomposition du problème de résolution de Sudoku en sous-problèmes indépendants. Cette approche modulaire a permis de développer le code de manière plus efficace et structurée, tout en préservant une cohérence globale.

Ainsi qu'on a développer une interface utilisateur intuitive permettant aux utilisateurs de soumettre des grilles de Sudoku et de recevoir des solutions en temps réel.

02

Méthodes de travail

2.1 IDE

Visual studio code

2.2 Dépendances

Nous avons choisi GitHub comme système de versionnage, car il permet :

- Un suivi des modifications et retour en arrière.
- Une collaboration simultanée et fusion de code.
- Une résolution des conflits et une gestion des problèmes.
- Une communication et documentation centralisées

Grâce à cet outil, nous avons pu éviter des bugs dans notre application.

2.3 Organisation

Concernant l'organisation du projet, nous avions des réunions avec M. Goudet chaque semaine pour lui exposer nos idées et l'avancement de notre code. Nous prenions en compte ses propositions et ses méthodes, ce qui nous a permis de développer cette application. Nous organisions notre travail en tenant des réunions sur Discord ainsi qu'à la bibliothèque pour mieux progresser.
Depuis le départ, nous connaissions les étapes essentielles à suivre pour réussir ce projet que nous avons validé avec M.Goudet. Alors, nous avons pu construire un plan à suivre et nous nous sommes divisés les taches équitablement en se basant sur les points forts de chacun.

02 Méthodes de travail

Dépendances

- **Customtkinter :** Utilisé pour créer une interface utilisateur moderne et personnalisable.
- **OpenCV :** Utilisé pour la capture vidéo et le traitement d'images.
- **PIL (Python Imaging Library) :** Utilisé pour la manipulation et l'affichage d'images.
- **torch et keras :** Utilisés pour charger et exécuter les modèles de machine learning.
- **TensorFlow :** Une bibliothèque de et de machine learning utilisée pour développer et entraîner des modèles d'apprentissage profond, cette bibliothéque a été développer par Google.
- **Keras (via TensorFlow) :** Une API de haut niveau intégrée à TensorFlow pour construire et entraîner des modèles de réseaux de neurones.
- **NumPy :** Utilisé pour les calculs numériques et les opérations sur des tableaux multidimensionnels.

- **NumPy :** Utilisé pour les calculs numériques et les opérations sur des tableaux multidimensionnels.

- **Random :** Fournit des fonctions pour générer des nombres pseudo-aléatoires.
- **Matplotlib** : Utilisé pour créer des visualisations graphiques telles que des graphiques, des histogrammes, etc.
- **Pandas** : Une bibliothèque puissante pour la manipulation et l'analyse de données. Elle fournit des structures de données flexibles et des outils pour manipuler des tableaux de données (DataFrames), ce qui facilite le nettoyage, la transformation, et l'analyse des données.
- **Pytorch :** Bibliothéque équivalent a Tensorflow, mais plus fléxible développé par Facebook.
- **scikit-learn:** offre des outils pour les taches de machine learning

03

Analyse du projet

3.1 Traitement d'image

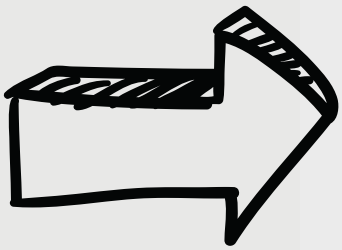
3.1.1 Chargement et Prétraitement de l'Image

· Le prétraitement de l'image de Sudoku commence par le chargement de l'image en utilisant la bibliothèque OpenCV. Cette étape initiale consiste à lire l'image en couleur, ce qui permet de préserver toutes les informations visuelles nécessaires pour les étapes suivantes.

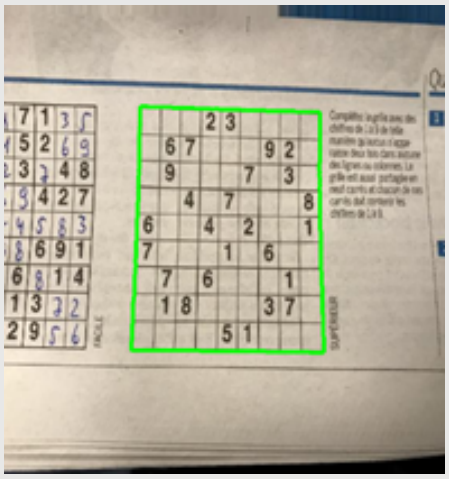
· Une fois l'image chargée, elle est convertie en niveaux de gris. Cette conversion est cruciale car elle simplifie les traitements ultérieurs en éliminant les informations de couleur et en se concentrant uniquement sur l'intensité lumineuse des pixels. Travailler en niveaux de gris permet de réduire la complexité computationnelle et d'améliorer la précision des étapes suivantes.

· Pour réduire le bruit et les variations locales présentes dans l'image, un flou gaussien est appliqué. Le flou gaussien aide à lisser l'image, rendant la détection de contours plus robuste en atténuant les détails inutiles et les petites variations. Cela prépare l'image pour une binarisation plus efficace.

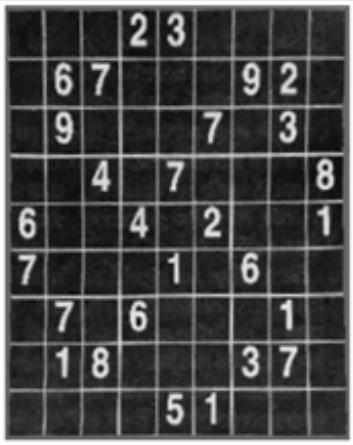
· La transformation de l'image en une version binaire, noire et blanche, est réalisée à l'aide d'un seuillage adaptatif. Ce type de seuillage est particulièrement utile pour les images de Sudoku, car il s'ajuste aux variations d'éclairage présentes dans l'image. En adaptant le seuil localement, le seuillage adaptatif produit une binarisation plus cohérente, même dans des conditions de luminosité inégales.



3.1.2 Détection et Transformation de la grille

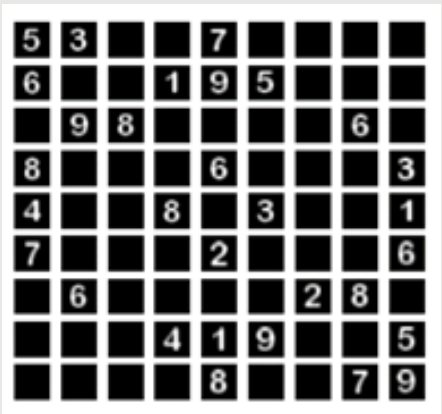


· Tout d'abord, les contours de l'image binarisée sont détectés, identifiant ainsi toutes les formes présentes. Le contour correspondant à la grille de Sudoku est supposé être le plus grand. En parcourant les contours détectés, nous calculons l'aire de chaque contour et sélectionnons celui avec la plus grande aire. Ce contour est ensuite simplifié en un polygone à quatre côtés, représentant les quatre coins de la grille.



· Les quatre coins de la grille sont ordonnés pour former un rectangle. Une transformation en perspective est appliquée, redressant l'image de sorte que la grille de Sudoku apparaisse plane et alignée, indépendamment des distorsions présentes dans l'image originale. Cette transformation utilise une matrice de transformation calculée à partir des coins de la grille.

3.1.3 Segmentation des Cellules



· Avec l'image redressée, la grille de Sudoku est divisée en 81 cellules individuelles en segmentant l'image en neuf lignes et neuf colonnes. Chaque cellule est extraite en découpant la grille en sous-images égales. Ces cellules sont ensuite traitées pour recentrer les chiffres et ajuster leur taille à 28x28 pixels, préparant ainsi chaque cellule pour la reconnaissance des chiffres.

3.2 reconnaissance des chiffres

Nous avons développé un modèle de réseau neuronal convolutif (CNN) pour la reconnaissance de chiffres manuscrits. Le modèle est basé sur l'architecture classique CNN avec quelques modifications mineures. Il se compose de deux couches convolutives, suivies de couches de pooling, de dropout et de deux couches entièrement connectées.
Ce modèle est inspiré de celui que nous avons réalisé lors du TP de reconnaissance des Kanjis, car les deux modèles sont conçus pour effectuer des tâches de classification.

3.2.1 modèle

- La couche convolutive 1 utilise 32 filtres de taille 5 x 5 et applique une fonction d'activation ReLU. La couche de pooling suivante réduit la dimension spatiale de l'image d'entrée par un facteur de 2.

- La couche convolutive 2 utilise 64 filtres de taille 5 x 5 et applique également une fonction d'activation ReLU. Une couche de dropout est ensuite appliquée pour éviter le surapprentissage. La couche de pooling suivante réduit à nouveau la dimension spatiale de l'image par un facteur de 2.

- L'aplatissement permet de transformer l'image tridimensionnelle en un vecteur unidimensionnel.

- La couche entièrement connectée 1 utilise 128 neurones et applique une fonction d'activation ReLU.

- La couche entièrement connectée 2 utilise 10 neurones, correspondant aux 10 classes de chiffres (0 à 9). La fonction d'activation finale est une fonction softmax logarithmique, qui normalise les sorties pour qu'elles représentent des probabilités de chaque classe.

3.2.2 entrainement et évaluation

L'entraînement et l'évaluation du modèle de reconnaissance de chiffres manuscrits suivent l'approche utilisée dans le TP de reconnaissance des Kanjis.
la seul différences est le dataset qui est celui de MNIST sur le quelle on effectue l'entrainement et la validation.

Camera

Tout comme le traitement d'image statique, elle commence par détecter les contours dans l'image capturée, identifiant le plus grand contour qui est supposé être la grille de Sudoku. Ce contour est ensuite transformé en un polygone à quatre côtés, et une transformation de perspective est appliquée pour redresser la grille. Toutefois, travailler avec une caméra en temps réel introduit des défis supplémentaires non présents dans le traitement d'images statiques. La qualité et l'éclairage de l'image peuvent varier considérablement, influençant la facilité avec laquelle les contours sont détectés et la précision de l'extraction des cellules. De plus, le mouvement continu de la caméra ou des changements subits dans l'environnement peuvent rendre le processus plus complexe et moins prévisible que le traitement d'une image fixe préalablement capturée. Ces facteurs ajoutent une couche de complexité et nécessitent des ajustements en temps réel pour assurer que la grille de Sudoku soit correctement capturée et traitée pour la reconnaissance des chiffres. Dans le cadre de notre projet, la prédiction précise des chiffres est essentielle. Toutefois, nous avons parfois rencontré des erreurs lors de cette étape cruciale.

3.3.Difficultés Rencontrées :

Prétraitement de l'Image

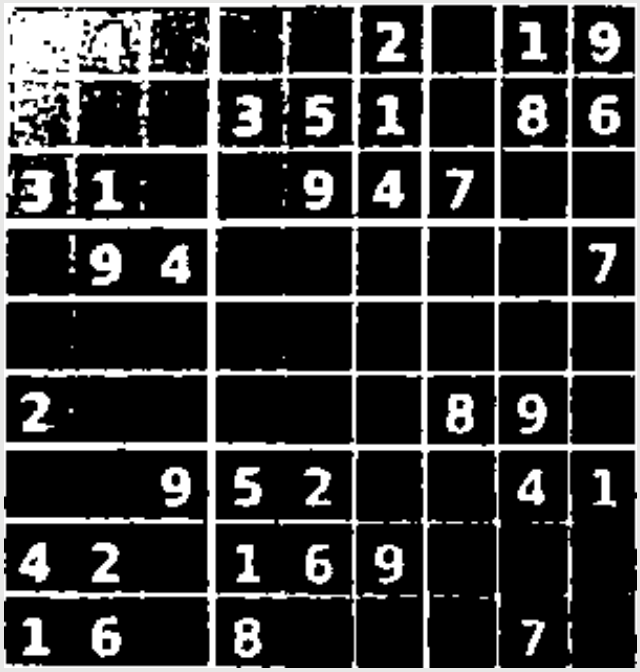
L'une des principales difficultés rencontrées lors de ce projet a été d'assurer une binarisation correcte de l'image de Sudoku pour toutes les conditions d'éclairage et types d'images. Le choix des paramètres pour le seuillage adaptatif a nécessité de nombreux ajustements et essais. Trouver les paramètres optimaux pour chaque image était complexe et crucial, car une mauvaise binarisation pouvait compromettre toutes les étapes suivantes.

Détection de la Grille

La détection précise des contours de la grille de Sudoku a également posé des défis. Cette étape peut échouer si l'image est trop bruitée ou si le contraste entre la grille et l'arrière-plan est insuffisant. De plus, les contours peuvent ne pas être détectés correctement si la grille est déformée ou si les lignes de la grille ne sont pas droites, rendant la transformation en perspective difficile et imprécise.

Segmentation des Cellules

La segmentation des cellules en 81 parties égales doit être très précise. Toutefois, cela peut être problématique si la grille de Sudoku dans l'image est déformée ou si les lignes ne sont pas parfaitement droites. Cette imprécision peut entraîner des cellules mal segmentées, ce qui complique la reconnaissance des chiffres. Un code général qui fonctionne pour n'importe quelle photo est difficile à écrire, car chaque image peut présenter des défis uniques.



Reconnaissance des Chiffres

La reconnaissance des chiffres avec le modèle de réseau de neurones dépend fortement de la qualité des images des cellules. Les chiffres doivent être bien centrés et clairs pour que le modèle puisse les identifier correctement. Des chiffres mal centrés ou de mauvaise qualité peuvent entraîner des erreurs de prédiction. De plus, même la qualité de l'image globale peut impacter les résultats : des pixels en excès ou un faible contraste peuvent perturber le modèle.

3.4 Résolution de sudoku

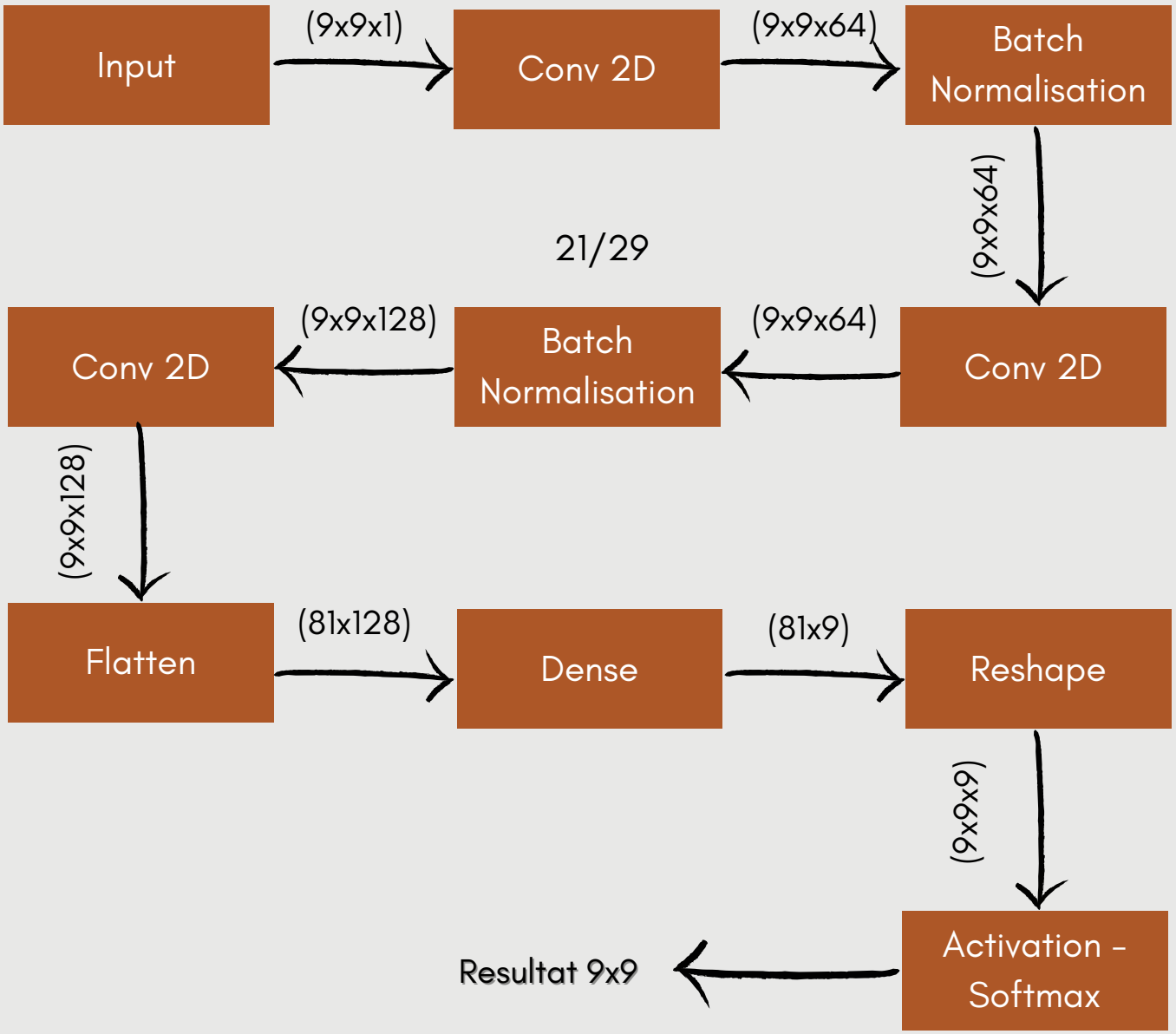
3.4.1 Donnée utilisé

Après de nombreuses recherches, nous avons trouvé un ensemble de données de 1 million de sudokus sur Kaggle. Ce jeu de données est sous forme de fichier CSV contenant deux colonnes : la première pour les grilles non résolues et la seconde pour les solutions.

La qualité, la diversité et la quantité d'un ensemble de données sont des éléments essentiels, et ce jeu de données répondait parfaitement à ces exigences.

3.4.2 Modèle de résolution de sudoku

La résolution de sudoku est un problème de satisfaction de contraintes. Cependant, nous avons choisi d'utiliser un réseau neuronal convolutif (CNN) pour le résoudre, bien que les CNN soient généralement plus adaptés aux problèmes de reconnaissance d'images.
Notre approche consiste à entraîner le CNN sur un ensemble de données très volumineux, ce qui lui permettra de reconnaître des motifs grâce aux couches de convolution. Une fois le modèle entraîné, il sera capable d'imiter le comportement d'un solveur de sudoku classique.



Explication du modéle :

1. **Couches convolutives:**

- `**keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same')**` (x2)
  - Extrait des caractéristiques spatiales à partir de la grille de Sudoku avec 64 filtres de taille 3x3.
  - Applique une activation ReLU pour introduire une non-linéarité.
  - Utilise un "same padding" pour maintenir les dimensions spatiales de l'entrée.
- `**keras.layers.Conv2D(128, kernel_size=(1, 1), activation='relu', padding='same')**`
  - Extrait des caractéristiques supplémentaires avec 128 filtres de taille 1x1.

2. Couches de normalisation par lots:

- `**keras.layers.BatchNormalization()**` **(x2)**
  - Normalise les activations des couches convolutives précédentes, améliorant la stabilité et la convergence de l'apprentissage.

3. Couche d'aplatissement:

- `**keras.layers.Flatten()**`
  - Transforme la sortie convolutive 3D en un vecteur 1D, préparant les données pour la couche dense.

4. Couche dense:

- `**keras.layers.Dense(81 * 9)**`
  - Prédit la valeur la plus probable pour chaque cellule, avec 81 * 9 unités (une pour chaque valeur possible dans chaque cellule).

5. Couche de remodelage:

- `**keras.layers.Reshape((-1, 9))**`
  - Remodelle la sortie de la couche dense en une grille 2D, représentant le Sudoku résolu.

6. Activation softmax:

- `**keras.layers.Activation('softmax')**`
  - Normalise les sorties en probabilités entre 0 et 1 pour chaque valeur de cellule.

3.4.3 entrainement et évaluation du modèle

3.4.3.1 entraînement du modèle

À partir du jeu de données de 1 million de sudokus, nous avons sélectionné 200 000 grilles, dont 80% sont utilisées pour l'entraînement du modèle. L'apprentissage automatique avec TensorFlow facilite grandement l'entraînement du modèle, car il suffit d'utiliser la méthode fit. Cette méthode permet d'ajuster les paramètres du modèle (poids et biais) en utilisant un ensemble de données d'entraînement et une fonction de perte.

```
model.fit(x_train, y_train, batch_size=32, epochs=2)
```

Voici une description du fonctionnement de la méthode fit :

1. Initialisation

- Le processus commence par l'initialisation des poids et des biais du modèle avec des valeurs aléatoires.

2. Boucle d'entraînement

Mélange des données : Les données d'entraînement (x_train et y_train) sont mélangées aléatoirement pour éviter que l'ordre des exemples n'affecte l'apprentissage.

3. Traitement par lot :

Le jeu de données d'entraînement est divisé en lots de taille batch_size.
Pour chaque lot :

- Le modèle prédit les sorties pour les entrées du lot.
- La fonction de perte calcule l'écart entre les prédictions du modèle et les étiquettes de sortie attendues (y_train).
- Le gradient de la fonction de perte par rapport aux poids et aux biais du modèle est calculé.

Les poids et les biais du modèle sont mis à jour en utilisant l'algorithme d'optimisation choisi (par exemple, Adam) et le gradient calculé.
Cette boucle d'entraînement par lots est répétée pour un nombre d'époques spécifié (epochs).

4. Sélection des hyperparamètres

La sélection des hyperparamètres a été réalisée en évaluant les performances du modèle sur un ensemble de données de validation distinct de l'ensemble de données d'entraînement. En choisissant les valeurs qui maximisaient la précision sur l'ensemble de validation, nous avons pu garantir que le modèle était capable de généraliser efficacement à de nouvelles données et d'obtenir des performances optimales sur l'ensemble de test.

3.4.3.2 – évaluation du modèle

Pour l'évaluation du modèle, j'ai utilisé les 20% restants des 200 000 sudokus sélectionnés, soit 40 000 grilles. Ces sudokus n'ont pas été utilisés pendant l'entraînement. Pour effectuer l'évaluation, j'ai créé une fonction qui vérifie si un sudoku est correctement résolu en respectant les contraintes du jeu. Ensuite, j'ai résolu chaque sudoku de l'ensemble de test et appliqué la fonction d'évaluation sur le résultat.

Le modèle a obtenu un taux de réussite de 95 % lors de l'évaluation, démontrant ainsi sa capacité à résoudre correctement la plupart des puzzles.

Voici les résultats obtenus :
Test 1:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

```
[[5 3 4 6 7 8 9 1 2]
 [6 7 2 1 9 5 3 4 8]
 [1 9 8 3 4 2 5 6 7]
 [8 5 2 7 6 1 4 9 3]
 [4 6 9 8 5 3 7 2 1]
 [7 1 3 9 2 4 8 5 6]
 [9 6 1 5 3 7 2 8 4]
 [2 8 7 4 1 9 6 3 5]
 [3 4 5 2 8 6 1 7 9]]
```

Test 2:
On a généré une grille de sudoku au hasard sous forme d'une matrice qu'on a donnée au modèle de résolution

```
[[6 0 0 0 0 9 0 0 4]
 [0 8 9 5 0 0 0 7 5]
 [5 0 0 0 5 0 3 0 9]
 [8 3 7 0 0 0 1 0 5]
 [0 2 0 0 0 0 0 5 0]
 [9 0 1 0 0 0 8 4 2]
 [2 0 5 0 7 0 0 0 8]
 [3 1 0 0 0 5 9 2 0]
 [7 0 0 3 0 0 0 0 1]]
```

```
[[6 7 3 4 1 9 2 8 4]
 [1 8 9 5 3 2 6 7 5]
 [5 4 2 6 5 7 3 1 9]
 [8 3 7 2 6 4 1 9 5]
 [4 2 6 8 9 1 7 5 6]
 [9 5 1 7 6 3 8 4 2]
 [2 1 5 9 7 6 4 3 8]
 [3 6 8 1 4 5 9 2 7]
 [7 9 4 3 2 8 5 6 1]]
```

3.4.5 Remarques :

Dans le cadre de ce projet, j'ai développé deux modèles de résolution de Sudoku utilisant différentes bibliothèques d'apprentissage automatique : PyTorch et TensorFlow. Le modèle TensorFlow a surpassé de manière significative le modèle PyTorch en termes de performance d'évaluation, atteignant un taux de réussite de 95%, comme indiqué précédemment, contre seulement 50% pour PyTorch.

Cette différence de performance notable suggère que l'architecture et l'implémentation du modèle TensorFlow sont mieux adaptées à la résolution de Sudoku que celles du modèle PyTorch. Il est possible que l'utilisation de fonctionnalités spécifiques à TensorFlow, telles que les couches convolutives ou les techniques de normalisation, ait contribué à l'amélioration des performances.

Cette différence s'explique par l'utilisation de l'API Keras de TensorFlow, qui facilite largement la tâche, notamment grâce à des méthodes comme Keras.fit dont nous avons parlé précédemment.

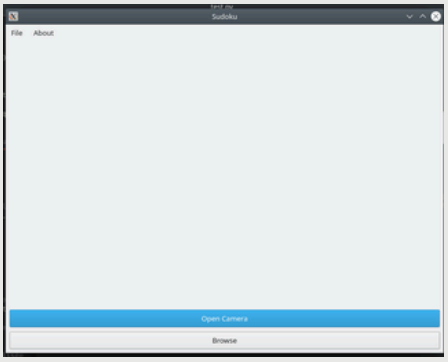
04

interface graphique

4.1 Introduction

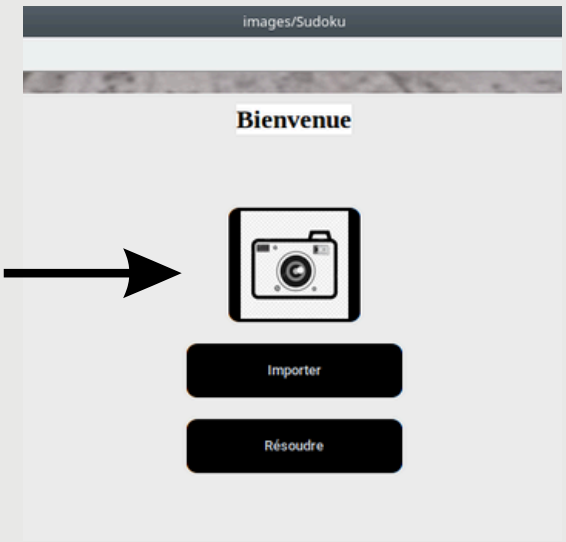
Avant de développer l'interface graphique actuelle, nous avons expérimenté avec plusieurs technologies différentes, notamment Tkinter et PyQt5. Initialement, notre application utilisait une interface classique créée avec ces outils. Bien que fonctionnelle, cette interface n'était pas du tout moderne et ne répondait pas aux attentes esthétiques et ergonomiques des utilisateurs contemporains.

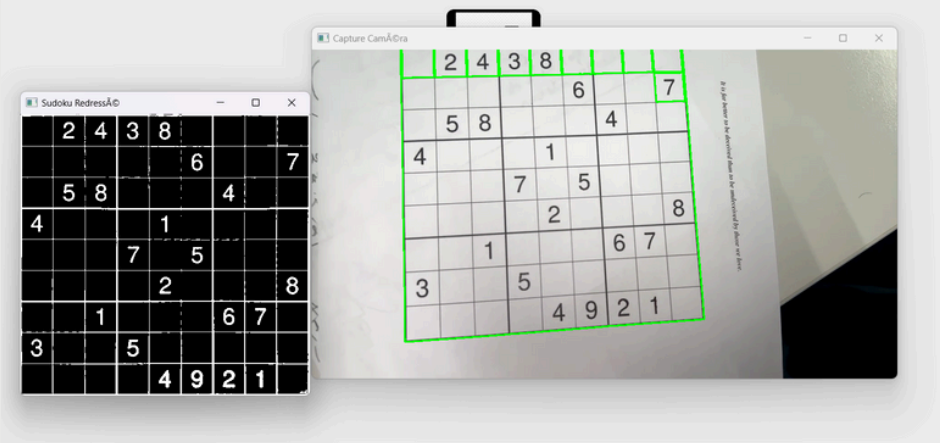
En recherchant une solution plus moderne et attrayante, nous avons décidé d'utiliser `customtkinter`, une bibliothèque qui permet de créer des interfaces graphiques plus élégantes et modernes tout en conservant la simplicité d'utilisation de Tkinter. Cette transition nous a permis d'améliorer l'expérience utilisateur en offrant une interface visuellement plaisante et plus intuitive, tout en maintenant la robustesse et la fiabilité de notre application.



4.2 Fonctionnement

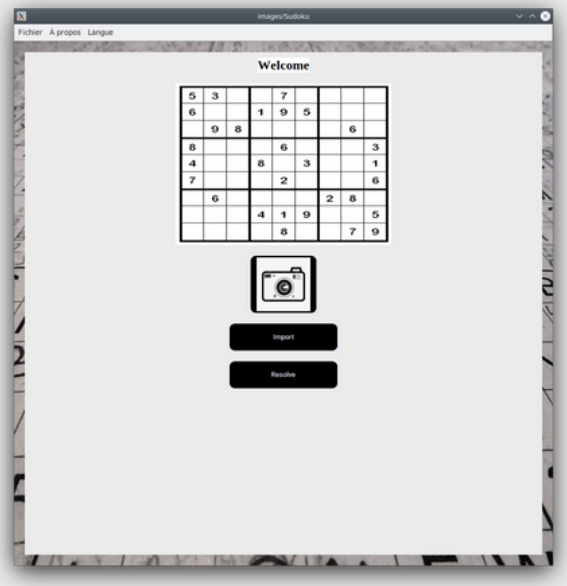
1. Bouton "Open Camera"
   - Fonctionnement : Ce bouton démarre la capture vidéo en temps réel depuis la caméra. Lorsqu'il est cliqué, la méthode open_camera est appelée.
   - Détails : La méthode open_camera désactive le bouton "Browse" pour éviter les conflits et appelle update_frame pour commencer la capture et l'affichage des images de la caméra dans l'interface.





04 Interface graphique

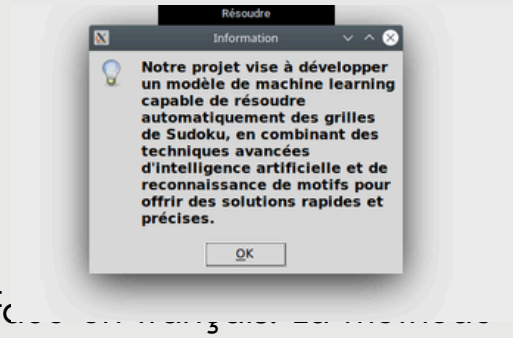
- Bouton "Importer"
  - Fonctionnement : Ce bouton ouvre une boîte de dialogue permettant à l'utilisateur de sélectionner un fichier image à importer. Lorsqu'il est cliqué, la méthode browse_file est appelée.
  - Détails : La méthode browse_file ouvre une boîte de dialogue pour choisir un fichier image (formats supportés : PNG, JPG, BMP). Si un fichier est sélectionné, il est affiché dans l'interface. Si la capture vidéo est en cours, elle est arrêtée.
- Bouton "Résoudre"
  - Fonctionnement : Ce bouton déclenche le processus de résolution du Sudoku en utilisant l'image importée. Lorsqu'il est cliqué, la méthode resultat est appelée.
  - Détails : La méthode resultat vérifie d'abord si un fichier image a été importé. Ensuite, elle utilise des modèles de reconnaissance et de résolution de Sudoku pour prédire et afficher les solutions sur l'image importée.



04 Interface graphique

Menu

- Menu "Fichier"
  - Option "Ouvrir"
    - Fonctionnement : Permet d'importer une image en ouvrant une boîte de dialogue. La méthode browse_file est appelée.
    - Détails : Identique au bouton "Importer", ouvre une boîte de dialogue pour sélectionner et afficher une image.
  - Option "Quitter"
    - Fonctionnement : Ferme l'application. La méthode quit de l'application est appelée.
    - Détails : Lorsque cette option est sélectionnée, l'application se ferme immédiatement.
- Menu "À propos"
  - Option "Info"
    - Fonctionnement : Affiche une boîte de message contenant des informations sur le projet. La méthode show_info est appelée.
    - Détails : La méthode show_info affiche une boîte de dialogue avec un message expliquant l'objectif du projet.



- Menu "Langue"
  - Option "Français"
    - Fonctionnement : Change la langue de l'interface en français. La méthode change_language est appelée avec l'argument 'fr'.
    - Détails : La méthode change_language met à jour les textes des widgets de l'interface pour les afficher en français.
  - Option "English"
    - Fonctionnement : Change la langue de l'interface en anglais. La méthode change_language est appelée avec l'argument 'en'.
    - Détails : La méthode change_language met à jour les textes des widgets de l'interface pour les afficher en anglais.

4.3 Interactions

1 .Interaction avec le Module de Reconnaissance

reconnaisance2

Le module de reconnaissance contient le modèle entraîné pour identifier les chiffres dans une grille de Sudoku. La fonction résultat dans le fichier principal utilise ce module pour obtenir les prédictions.

python

```
def resultat(self, chemin):
    if not chemin:
        messagebox.showerror("Erreur", "Veuillez importer une image.")
        return

    model_reconnaisance = model.Net()

model_reconnaisance.load_state_dict(torch.load('/home/etud/Bureau/projet_Sud
oku/reconnaisance2/mnist_model.pth'))
    model_reconnaisance.eval()

    grille, x, y = get_sudoku_predictions(chemin, model_reconnaisance)
    grille = np.array(grille).reshape(9, 9)
```

2.Interaction avec le Module de Résolution : resolution

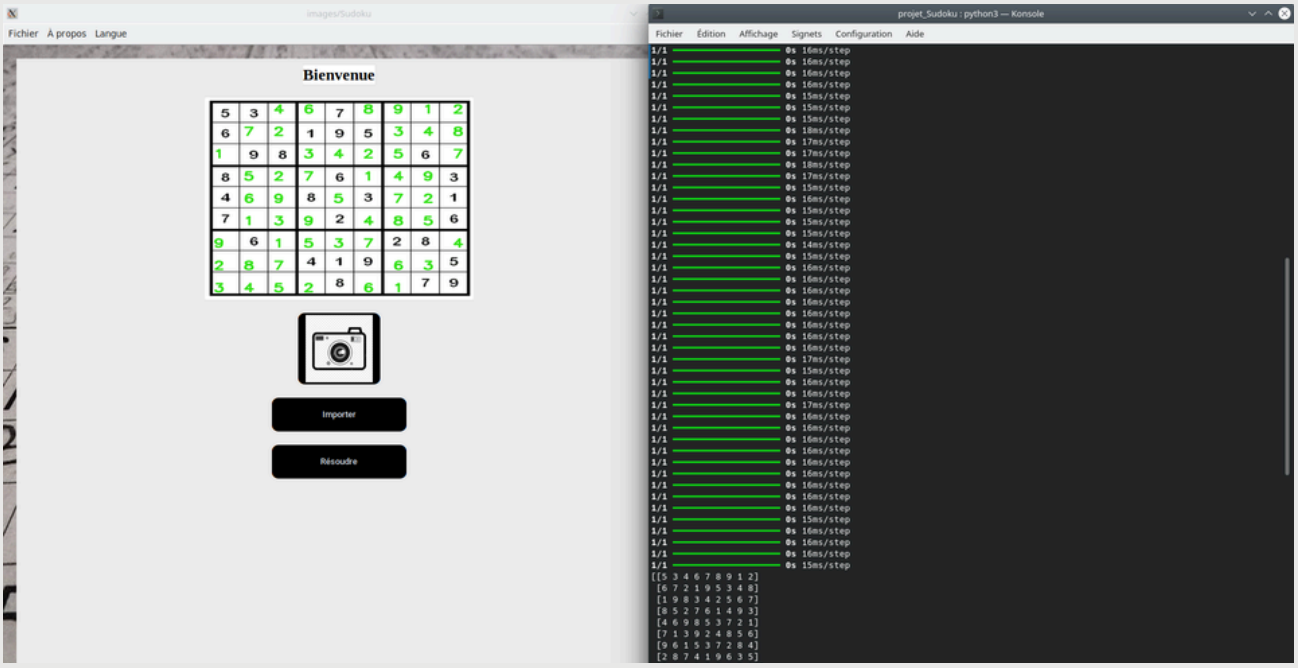
Le module de résolution utilise un modèle de machine learning pour résoudre les grilles de Sudoku prédictes par le module de reconnaissance.

```
model_resolution =
keras.models.load_model('/home/etud/Bureau/projet_Sudoku/resolution/sudoku_model.h5')
predi = outils.solve_sudoku(grille, model_resolution)
```

3.Interaction avec le Module d'Extraction : extraction

Le module d'extraction fournit les fonctions nécessaires pour extraire les chiffres et leurs coordonnées à partir de l'image de la grille de Sudoku.

```
grille, x, y = get_sudoku_predictions(chemin, model_reconnaisance)
draw_predictions_on_image(chemin, x, y, predi, result_image_path)
```



05

Exemple d'utilisation

1

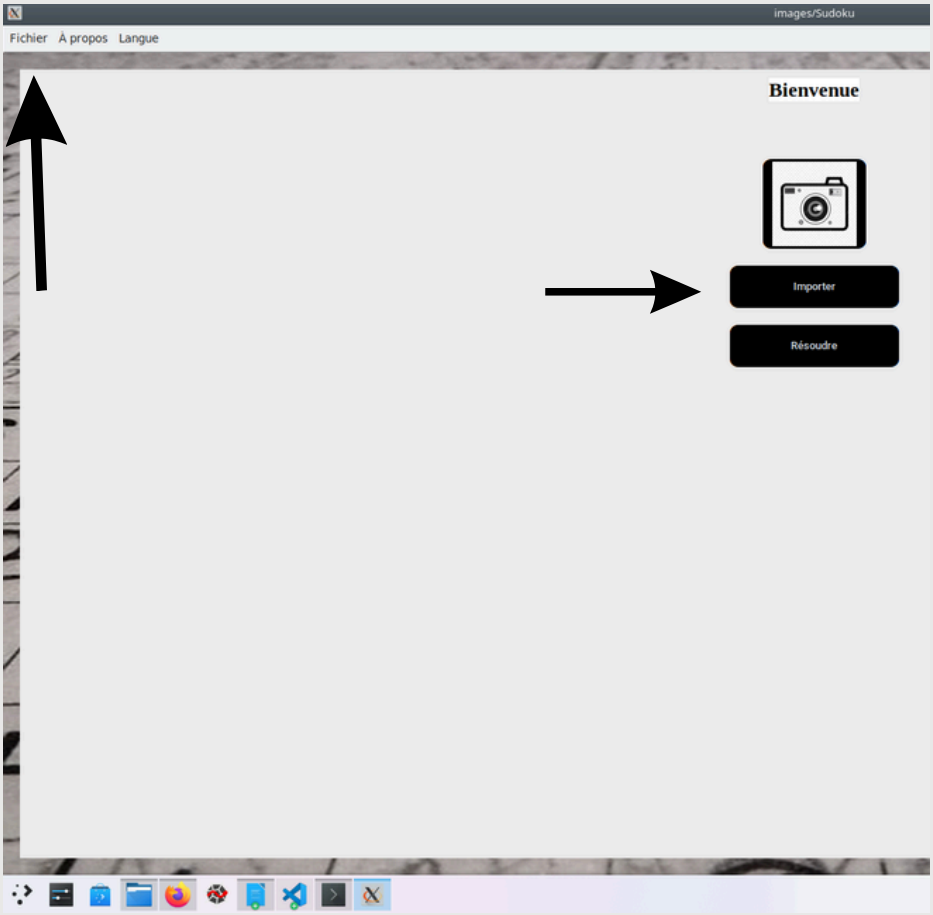
Exécutez le fichier `custom.py` en ligne de commande avec `python3 custom.py`. Assurez-vous d'être dans le bon répertoire.

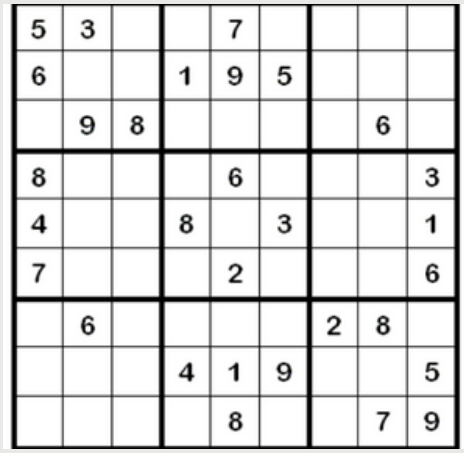
2

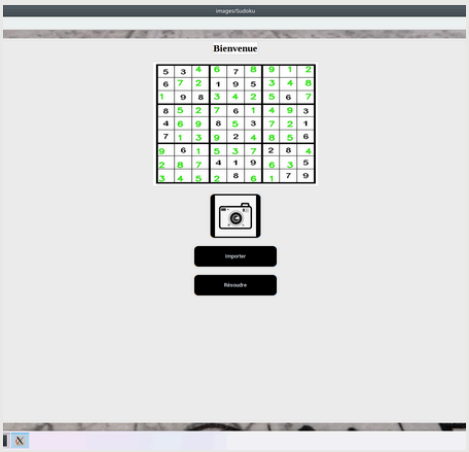
Sur la fenêtre qui s'ouvre, cliquez sur "Importer" ou "Fichier" dans le menu, puis sur "Ouvrir". Ensuite, choisissez une image d'un sudoku.

3

Cliquez sur "Résoudre" et patientez quelques secondes jusqu'à ce que le résultat s'affiche directement sur l'image importée.







06

Conclusion

Ce projet a représenté une exploration approfondie de l'application des techniques de machine learning pour automatiser la résolution des grilles de Sudoku. À travers ce travail, nous avons non seulement atteint notre objectif de développer un modèle performant capable de résoudre rapidement et précisément des Sudokus, mais nous avons également acquis des compétences précieuses en matière de manipulation des données et d'utilisation des algorithmes de machine learning.

Nous avons appris à mieux maîtriser différentes méthodes de machine learning, notamment les réseaux de neurones profonds et les techniques de reconnaissance de motifs. En parallèle, nous avons renforcé notre maîtrise de Python et des bibliothèques associées, ce qui nous sera bénéfique dans nos futures aventures professionnelles et académiques.

Ce projet a également été une excellente opportunité de travail collaboratif. Nous avons apprécié la dynamique de groupe et la possibilité de partager nos idées et nos compétences. Travailler avec notre professeur référent a été particulièrement enrichissant ; ses conseils et son expertise ont grandement contribué à la réussite de ce projet.

En conclusion, ce projet nous a non seulement permis de créer une solution élégante à un problème classique, mais il a également été une expérience d'apprentissage inestimable, enrichissant nos connaissances techniques et renforçant nos compétences en collaboration et en gestion de projet. Nous sommes très reconnaissants pour l'opportunité de mener à bien ce projet et pour le soutien continu de notre professeur.

07

Bibliographie

sources de dataset:
https://www.kaggle.com/datasets/rohanrao/sudoku (dataset utilisé dans la résolution)
https://www.kaggle.com/code/cdeotte/25-million-images-0-99757-mnist (dataset utilisé dans la reconnaissance des chiffres manuscrit)

sources d'information :
https://medium.com/analytics-vidhya/how-to-solve-sudoku-with-convolutional-neural-networks-cnn-c92be8830c52
https://cs230.stanford.edu/files_winter_2018/projects/6939771.pdf

autres sources :
https://sudoku.com/fr
https://fr.wikipedia.org/wiki/Sudoku

sites officiels pour les bibliothèques mentionnées :
1. **TensorFlow**
 - https://www.tensorflow.org/
2. **Keras**
 - https://github.com/keras-team/keras
3. **NumPy**
 - https://numpy.org/doc/
4. **OpenCV**
 - https://opencv.org/
5. **PyTorch**
 - https://pytorch.org/
6. **Matplotlib**
 - https://matplotlib.org/stable/contents.html
7. **Pandas**
 - https://pandas.pydata.org/pandas-docs/stable/
8. **Scikit-learn**
 - https://scikit-learn.org/stable/

FIN

Nous vous remercions sincèrement pour l'intérêt que vous avez manifesté en prenant le temps de lire notre rapport et d'évaluer notre projet. Votre attention et vos commentaires sont très précieux pour nous, et nous espérons que notre travail a su répondre à vos attentes et démontrer notre engagement et nos compétences.

ABOUAKIL Ibrahim
ARHARBI Hamza
MOUSSOUNI LITICIA