OCR Sudoku Solver
Cahier des charges

Table des matières

I. Le cadre

Le projet est à réaliser en groupe de quatre personnes (et uniquement quatre). Il s'étend du mois de septembre au mois de décembre.

Ce document présente les objectifs du projet, les différentes parties attendues, le planning de réalisation et ses différentes contraintes.

II. Les dates clés

- **Vendredi 17 septembre 2021** : date limite de constitution des groupes ;
- **Semaine du 25 octobre 2021** : première soutenance (éventuellement en fin de journée) ;
- **Semaine du 6 décembre 2021** : soutenance finale (éventuellement en fin de journée).

III. Le sujet

1. Description générale

L'objectif de ce projet est de réaliser un logiciel de type OCR (*Optical Character Recognition*) qui résout une grille de sudoku.

Votre application prendra donc en entrée une image représentant une grille de sudoku et affichera en sortie la grille résolue.

Dans sa version définitive, votre application devra proposer une interface graphique permettant de charger une image dans un format standard, de la visualiser, de corriger certains de ses défauts, et enfin d'afficher la grille complètement remplie et résolue. La grille résolue devra également pouvoir être sauvegardée.

Votre application devra aussi posséder un aspect apprentissage, qui pourra être séparé de la partie principale, et qui permettra d'entraîner votre réseau de neurones, puis de sauvegarder et de recharger le résultat de cet apprentissage.

2. Description détaillée

2.1. Principales étapes

Le traitement effectué par votre application sera approximativement le suivant :
- Chargement d'une image ;
- Suppression des couleurs (niveau de gris, puis noir et blanc) ;
- Prétraitement ;
- Détection de la grille ;
- Détection des cases de la grille ;
- Récupération des chiffres présents dans les cases ;

- Reconnaissance de caractères (ici les chiffres) ;
- Reconstruction de la grille ;
- Résolution de la grille ;
- Affichage de la grille résolue ;
- Sauvegarde de la grille résolue.

On notera qu'aucune information de couleur n'est nécessaire pour la reconnaissance de caractères et qu'il est même fortement conseillé de travailler en noir et blanc.

2.2. Découpage de l'image

Le découpage de l'image est nécessaire pour envoyer à la reconnaissance les morceaux d'images correspondants à des chiffres. Il faudra donc identifier la position des cases de la grille afin d'extraire les chiffres sous forme d'image et aussi de déterminer les cases vides.

2.3. Reconnaissance de caractères (réseau de neurones)

Cette partie nécessite une phase d'apprentissage pendant laquelle votre réseau de neurones va apprendre à reconnaître les différents caractères.

Un réseau de neurones est un outil permettant d'apprendre une fonction (possiblement non linéaire) à l'aide d'exemples. Dans la phase d'apprentissage du réseau, vous allez fournir des entrées déjà identifiées (ici des blocs d'images) auxquelles votre réseau répondra. La marge d'erreur de ces réponses sera alors utilisée pour corriger les poids internes du réseau. Avec un certain nombre d'exemples, votre réseau finira, dans la majorité des cas, par fournir les bonnes réponses. La sortie usuelle pour ce genre de problème est une probabilité par symbole du jeu de caractères. Le symbole avec la plus forte probabilité sera retenu comme le symbole reconnu.

Pour implémenter votre réseau de neurones, voici quelques termes à rechercher : **apprentissage supervisé, perceptron multicouche, rétropropagation, descente de gradient, AdaBoost.**

Le site http://neuralnetworksanddeeplearning.com (en anglais) fournit beaucoup d'informations théoriques et pratiques qui pourraient vous être utiles (notamment pour la partie paramétrage de votre réseau située dans le chapitre 3 du site).

Il existe de nombreuses formes de réseaux de neurones, mais pour votre projet vous n'avez probablement pas besoin de plus d'une couche cachée. Par contre, il peut être judicieux de remplacer la fonction d'activation de la dernière couche par un *softmax* plutôt que par la fonction sigmoïde habituelle. Vous trouverez toutes ces informations dans le lien fourni plus haut.

2.4. Prétraitement

L'objectif de cette partie est d'améliorer la qualité des images traitées. Cela permettra à votre application d'accepter en entrée des documents directement issus d'un scanner ou d'une photo. Voici les prétraitements que vous devez réaliser :

- Redressement manuel de l'image (rotation de l'image à partir d'un angle saisi par l'utilisateur) ;
- Redressement automatique de l'image (détection d'angle de rotation puis rotation de l'image) ;
- Élimination des bruits parasites (grain de l'image, tâches, etc.) ;
- Renforcement des contrastes.

2.5. Résolution d'une grille de sudoku

Vous devrez implémenter un algorithme de résolution de sudoku en langage C. Pour tester cet algorithme, vous devrez concevoir un petit programme appelé **`solver`**. Ce sera une application qui s'exécutera en ligne de commande uniquement. Il prendra en entrée un nom de fichier dans un format décrit en annexe. Il résoudra le sudoku et générera un fichier contenant le résultat. Le nom du fichier de sortie sera le nom du fichier d'entrée avec l'ajout de l'extension « .result ». Le format du fichier de sortie sera le même que celui du fichier d'entrée.

Par exemple :

```
$ ls
grid_00 grid_01 grid_02 solver
$ ./solver grid_00
$ ls
grid_00 grid_00.result grid_01 grid_02 solver
```

2.6. Format d'affichage de la grille résolue

Le choix du format d'affichage de la grille résolue vous appartient. Toutefois, **la qualité de l'affichage sera prise en compte dans l'évaluation**.

Il est donc fortement conseillé d'afficher le résultat sous la forme d'une image plutôt que sous la forme de simples données textuelles. Dans le cas d'une image, elle devra pouvoir être sauvegardée dans un format d'image standard.

Il est également préférable que les cases remplies par l'algorithme (celles qui étaient initialement vides) s'affichent d'une couleur différente de celles initialement remplies.

IV. Les contraintes du projet

1. Les contraintes techniques

Vous devrez respecter les points suivants :

- Votre projet sera écrit en langage C ;
- Il devra compiler et s'exécuter dans l'environnement de travail qui vous est fourni par l'école (Linux) ;
- Lors des soutenances, les démonstrations se feront sur les machines de l'école ;
- Tous les logiciels installés sur les machines de l'école sont autorisés à être utilisés pour le projet ;
- Votre code devra compiler sans erreur avec au moins les options suivantes : -Wall -Wextra ;
- Les lignes de code ne devront pas dépasser 80 colonnes et ne devront pas contenir d'espaces inutiles en fin de ligne ;
- Tous les identifiants utilisés dans votre code ainsi que les commentaires devront être en anglais ;
- Votre dépôt devra contenir **uniquement** les fichiers précisés dans la partie *V.3 Le contenu du dépôt GIT* ;
- Votre dépôt ne devra pas contenir :
  - De fichiers exécutables ;
  - De code source extérieur (bibliothèques, autres projets, etc.) ;
  - Tout autre type de fichier non nécessaire au projet.

2. Gestion de versions et procédure de rendu

Pour le projet, vous disposerez d'un dépôt *git* utilisable par tous les membres d'un groupe. Il est fortement conseillé d'utiliser les fonctionnalités de gestion de version de *git*.

Pour chaque soutenance, un rendu de code vous sera demandé avant la soutenance. Ce rendu de code s'effectuera à l'aide de *git* : les examinateurs cloneront votre dépôt sur la branche principale (*master*) à la date et à l'heure prévues pour le rendu. **Seuls les fichiers présents sur votre dépôt au moment du clonage seront disponibles pendant la soutenance.**

V. Les livrables

1. Le plan de soutenance

Un plan de soutenance sera à fournir au début de chaque soutenance. Il sera au format papier et devra tenir sur le recto d'une feuille A4.

2. Le rapport en version papier

Tout comme le plan de soutenance, un rapport présentant votre projet devra être imprimé et rendu le jour de la soutenance.

Il devra, entre autres, présenter :
- Votre groupe ;
- La répartition des charges ;
- L'état d'avancement du projet ;
- Les aspects techniques.

3. Le contenu du dépôt GIT

3.1. Le rapport en version numérique

Une version numérique de votre rapport devra également être présente dans votre dépôt GIT. Elle devra être au format PDF et se situer à la racine de votre dépôt. Le fichier devra porter le nom suivant :
- **`rapport_1.pdf`** pour la première soutenance (15 pages minimum) ;
- **`rapport_2.pdf`** pour la soutenance finale (40 pages minimum).

Attention, la récupération des rapports sur le dépôt GIT sera automatisée. Il est donc important de respecter les noms de fichiers ci-dessus, faute de quoi votre rapport ne sera pas évalué.

3.2. Le code

Votre dépôt GIT devra contenir l'intégralité du code source de votre projet ainsi qu'un fichier **`Makefile`**. Ce dernier devra fournir au moins les règles *all* et *clean*.

3.3. Les fichiers README et AUTHORS

Votre dépôt GIT devra également contenir :
- Un fichier **`README`** décrivant de manière concise comment utiliser votre application. Ce fichier devra être au format texte et pouvoir être lu correctement dans un terminal avec une commande comme *cat* ou *less*. Il pourra éventuellement être au format *markdown* ;
- Un fichier **`AUTHORS`** contenant les logins des membres de votre groupe. Il devra contenir une ligne par membre du groupe sous la forme suivante : login (NOM Prénom).

3.4. Les images

Votre dépôt GIT devra aussi contenir quelques images qui pourront servir aux démonstrations de vos différentes parties. Attention, ne surchargez pas votre dépôt avec un grand nombre d'images à très haute résolution. Trois ou quatre images à une résolution correcte devraient suffire.

Il vous faudra également un jeu d'images pour l'apprentissage de votre réseau de neurones.

VI. Les soutenances

1. Structure générale

Pour chaque soutenance, vous devrez prévoir une introduction, une conclusion et une démonstration de toutes les fonctionnalités attendues. Étant donné que le sujet est commun à tous les groupes, il n'est pas utile de le présenter ni de paraphraser le présent document.

La première soutenance est une soutenance de suivi. L'objectif est de montrer votre état d'avancement. Vous devrez également expliquer le fonctionnement de votre réseau de neurones. Attention, là encore, il ne vous est pas demandé de faire un exposé sur les réseaux de neurones, mais bien de décrire le fonctionnement de celui que vous avez implémenté.

Pour la soutenance finale, vous devrez faire la démonstration d'un programme fini. Il est important que celui-ci soit homogène et permette d'effectuer la chaîne de traitements attendus. Il est plus important d'avoir un programme complet (et pas une collection de bouts de code) même si celui-ci ne fonctionne que sur un jeu réduit d'exemples.

2. Découpage du projet

Votre projet se découpera en plusieurs éléments. Un élément est considéré comme étant terminé si votre programme est capable d'effectuer la tâche attendue sur quelques exemples non triviaux. Bien évidemment, l'évaluation complète de l'élément dépendra de sa qualité et de sa robustesse face à des tests avancés.

Éléments devant être présentés et obligatoirement terminés pour la première soutenance :

- Chargement d'une image et suppression des couleurs ;
- Rotation manuelle de l'image ;
- Détection de la grille et de la position des cases ;
- Découpage de l'image (sauvegarde de chaque case sous la forme d'une image) ;
- Implémentation de l'algorithme de résolution d'un sudoku. Vous devrez implémenter cet algorithme dans le programme en ligne de commande **`solver`** (*cf. III.2.5 Résolution d'une grille de sudoku*) ;
- Une preuve de concept de votre réseau de neurones. Pour cette preuve, vous réaliserez un mini réseau capable d'apprendre la fonction OU EXCLUSIF.

Éléments devant être commencés pour la première soutenance :

- Sauvegarde et chargement des poids du réseau de neurones ;
- Jeu d'images pour l'apprentissage ;
- Manipulation de fichiers pour la sauvegarde des résultats.

Éléments devant être présentés et obligatoirement terminés pour la soutenance finale :

- Le prétraitement complet ;
- Le réseau de neurones complet et fonctionnel
  - Apprentissage ;
  - Reconnaissance des chiffres de la grille.
- La reconstruction de la grille ;
- La résolution de la grille ;
- L'affichage de la grille ;
- La sauvegarde du résultat ;
- Une interface graphique permettant d'utiliser tous ces éléments.

Éléments supplémentaires (bonus évalués uniquement si la partie obligatoire est faite) :

- Reconnaissance de chiffres manuscrits ;
- Résolution d'hexadoku (une grille 16x16 avec la notation hexadécimale) ;
- Site internet présentant votre projet ;
- Autres.

VII. Quelques conseils

Ce projet est conçu pour être réalisé en équipe de quatre personnes sur une période totale de trois mois. Certains éléments, comme le réseau de neurones, nécessitent des phases d'ajustement et parfois des phases de calcul pouvant prendre du temps. **Il est donc important de commencer à travailler dès la constitution de votre groupe, sous peine de ne pas disposer d'assez de temps pour finaliser votre projet.**

Il peut être également judicieux de concevoir des tests indépendants pour chaque partie. Par exemple, le réseau de neurones devra recevoir en entrée des images de caractères déjà découpées et redimensionnées. Afin de tester et d'entraîner votre réseau de neurones, il serait pertinent de disposer d'un jeu de tests où les images sont déjà des caractères seuls au bon format. Ainsi, même si votre découpage de l'image n'est pas prêt, vous pourrez faire avancer cette partie.

Lors de la première soutenance, nous vous demanderons de faire la démonstration des parties attendues. Pensez à préparer votre projet pour faire la démonstration de ces parties de manière simple et efficace (**visualisation intermédiaire**, exécutions indépendantes, etc.).

Il y a quelques écueils en C qui pourraient vous faire perdre beaucoup de temps. Vous aurez besoin de porter une attention particulière à de nombreux détails techniques. Prenez de bonnes habitudes pour éviter de perdre du temps. Notamment, pour tous vos problèmes de mémoire, utilisez un outil comme *valgrind* ou l'option *-fsanitize=address* de *gcc*.

VIII. Annexes

1. Format de fichier utilisé pour le programme `solver`

Considérons la grille ci-dessous :

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

Le format du fichier correspondant sera le suivant :

```
... ..4 58.
... 721 ..3
4.3 ... ...

21. .67 ..4
.7. ... 2..
63. .49 ..1

3.6 ... ...
... 158 ..6
... ..6 95.
```

Il s’agit d’un fichier texte avec les caractéristiques suivantes :

- Il comporte 11 lignes :
  - 9 lignes de 11 caractères ;
  - 2 lignes vides.
- Les cases vides de la grille sont remplacées par des points ;
- Horizontalement, il y a un espace entre chaque groupe de 9 cases ;
- Verticalement, il y a une ligne vide entre chaque groupe de 9 cases.

Par exemple :

```
$ ls
grid_00 solver
$ cat grid_00
... ..4 58.
... 721 ..3
4.3 ... ...

21. .67 ..4
.7. ... 2..
63. .49 ..1

3.6 ... ...
... 158 ..6
... ..6 95.
$ ./solver grid_00
$ ls
grid_00 grid_00.result solver
$ cat grid_00.result
127 634 589
589 721 643
463 985 127

218 567 394
974 813 265
635 249 871

356 492 718
792 158 436
841 376 952
```

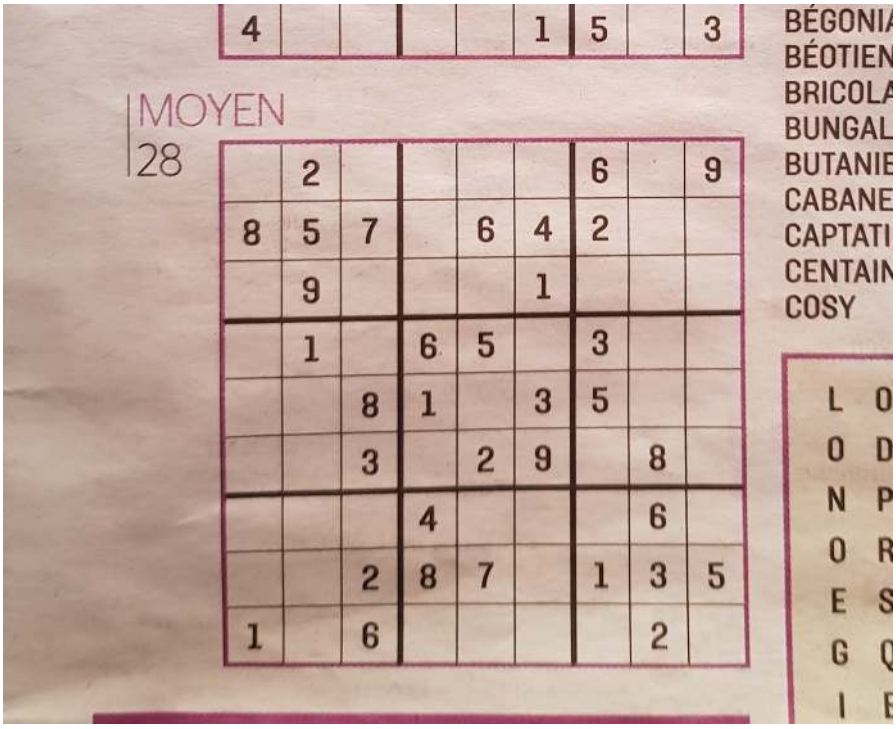
2. Exemples d'images

Vous trouverez ci-dessous quelques images qui pourront être utilisées pour tester votre code. La résolution de ces images est bien plus grande que ce qui s'affiche ci-dessous. Certains lecteurs PDF (comme Evince) vous permettent de sauvegarder ces images dans différents formats.

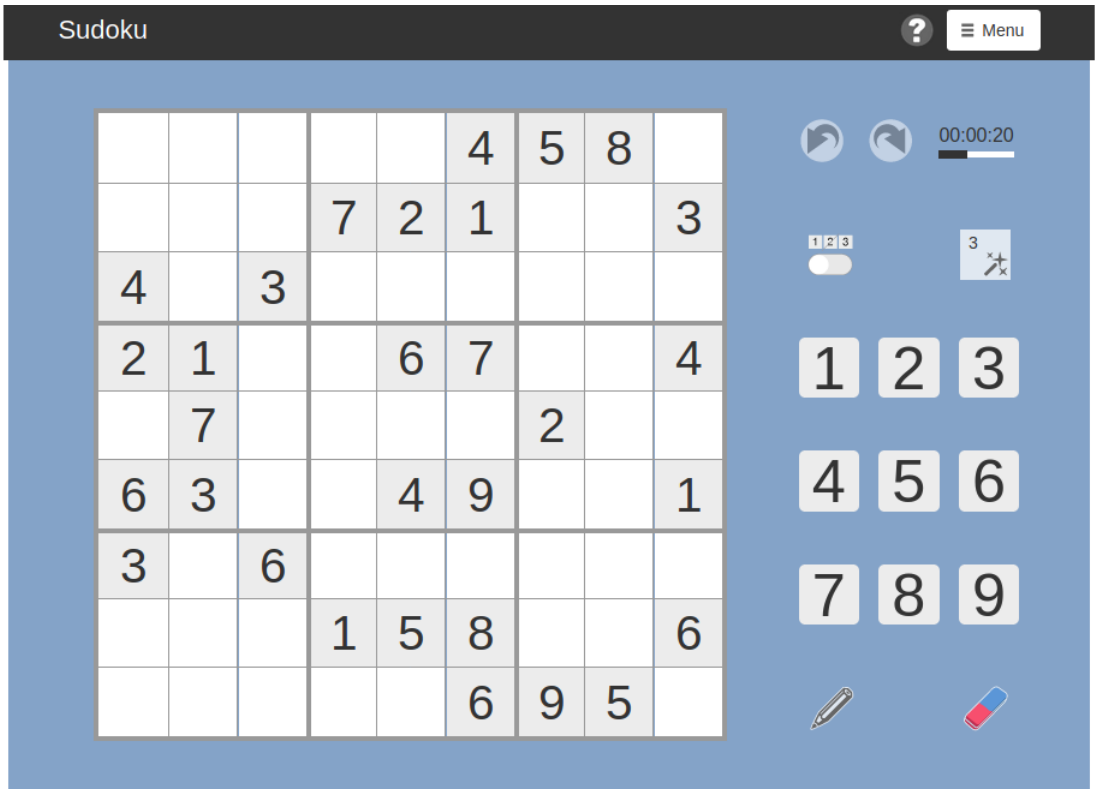
2.1. Image 1

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

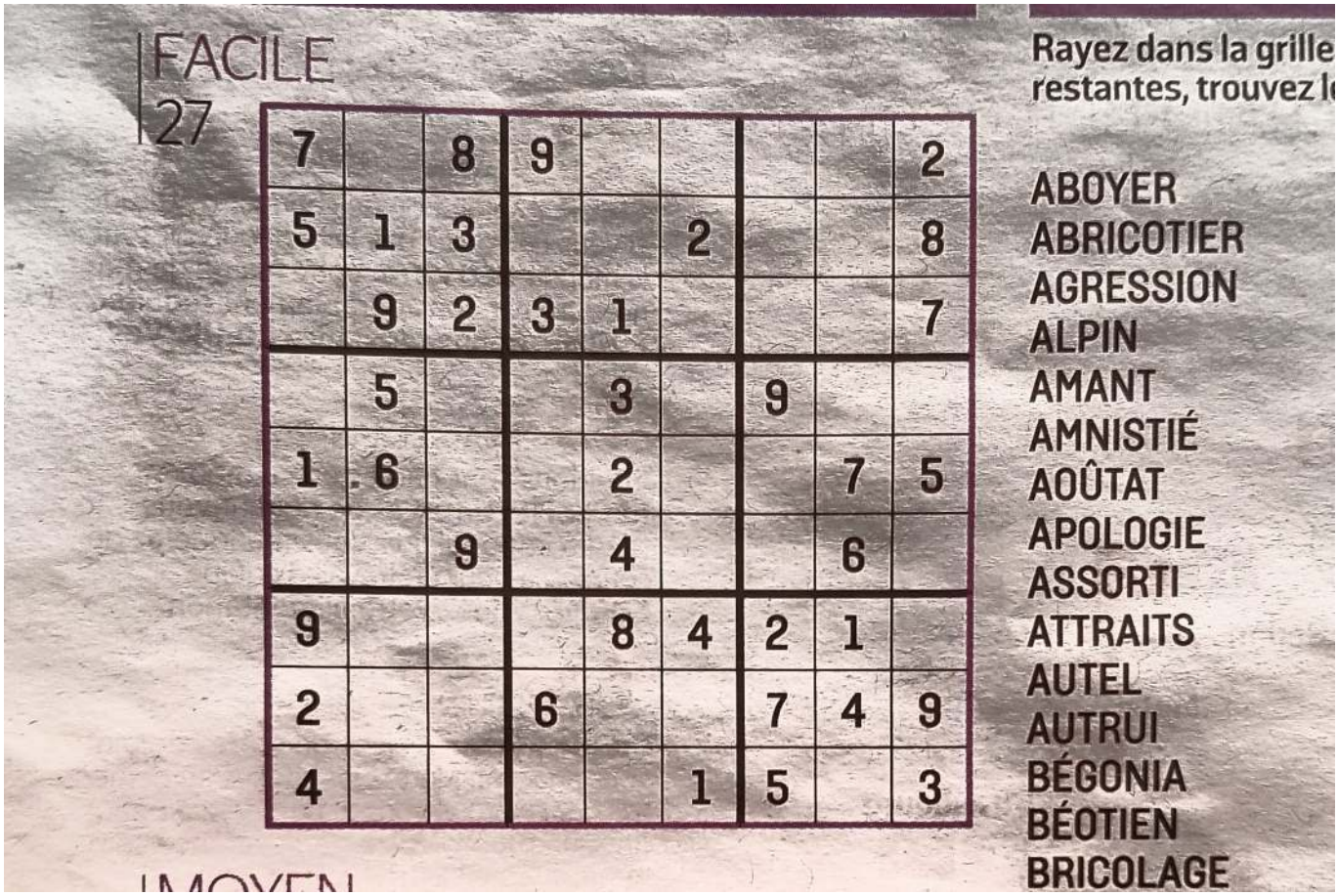
2.2. Image 2



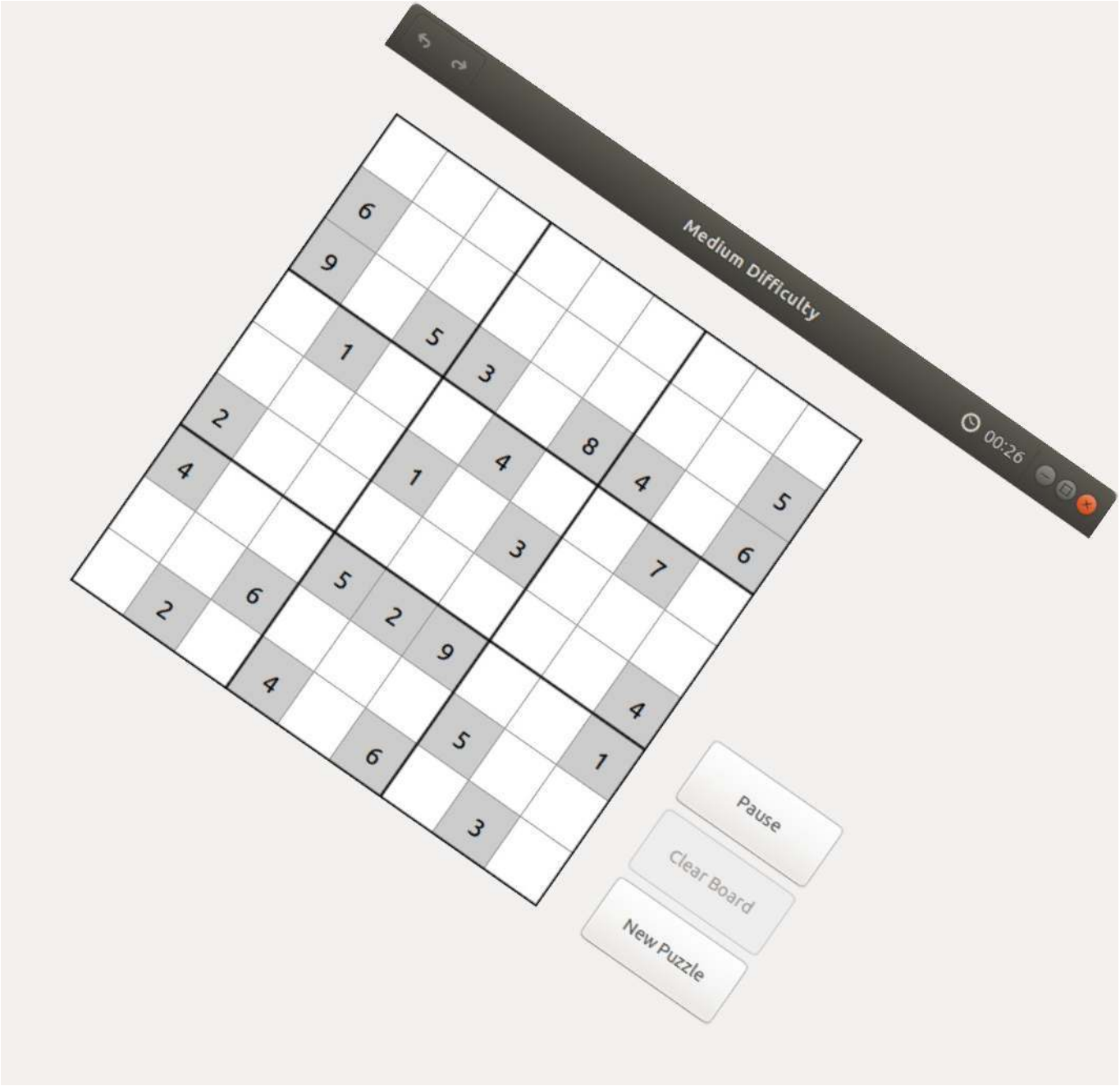
2.3. Image 3



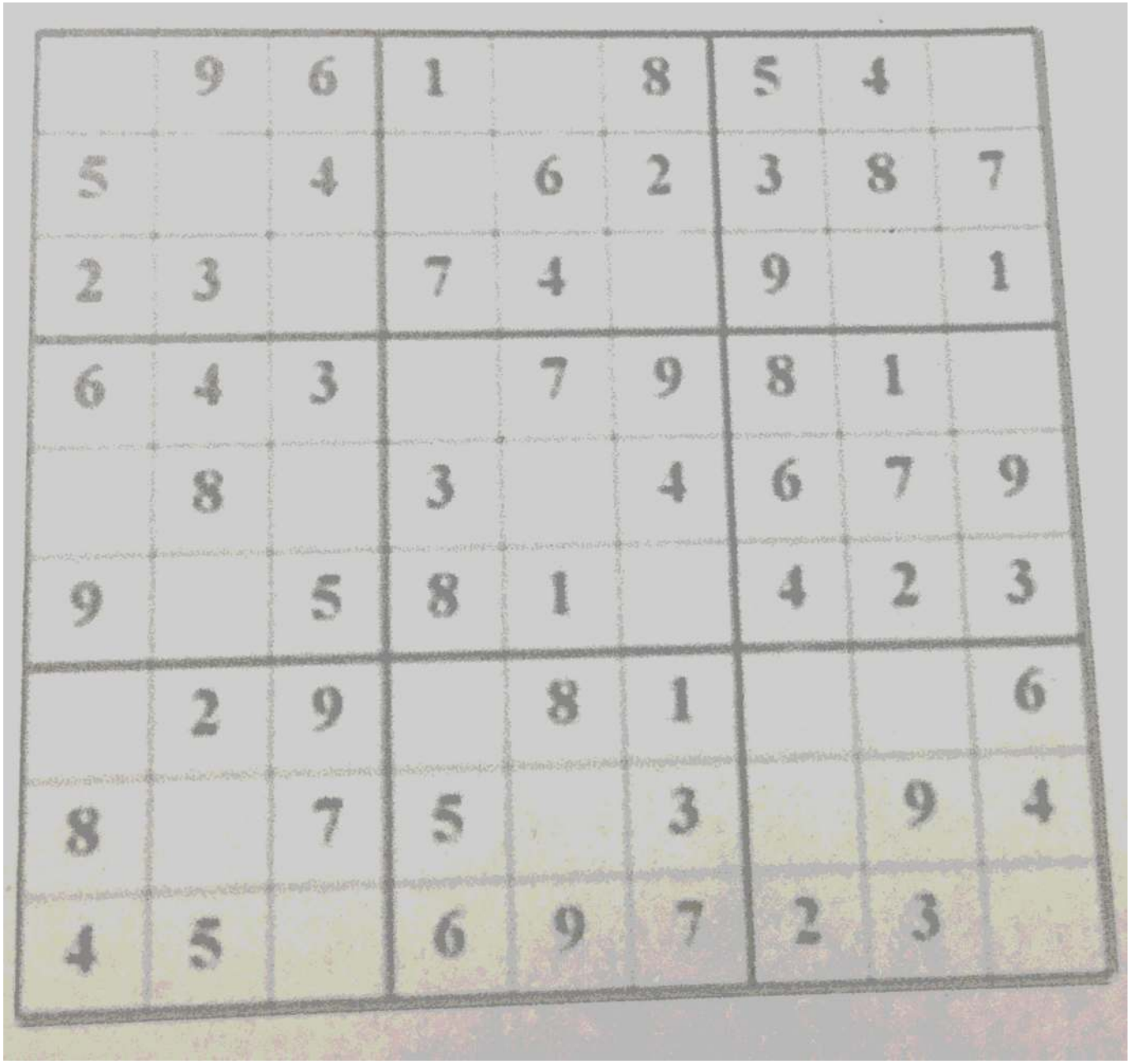
2.4. Image 4



2.5. Image 5



2.6. Image 6



2.7. Liens de téléchargement

Au besoin, ces images peuvent être téléchargées au format JPEG à l'aide des liens ci-dessous :

- http://www.debug-pro.com/epita/prog/s3/project/image_01.jpeg
- http://www.debug-pro.com/epita/prog/s3/project/image_02.jpeg
- http://www.debug-pro.com/epita/prog/s3/project/image_03.jpeg
- http://www.debug-pro.com/epita/prog/s3/project/image_04.jpeg
- http://www.debug-pro.com/epita/prog/s3/project/image_05.jpeg
- http://www.debug-pro.com/epita/prog/s3/project/image_06.jpeg