

Rapport de soutenance 2

~~Eva Blum~~
Gustav Berloty
Julien Auzanneau
~~Jonas Attia~~

Avril 2022

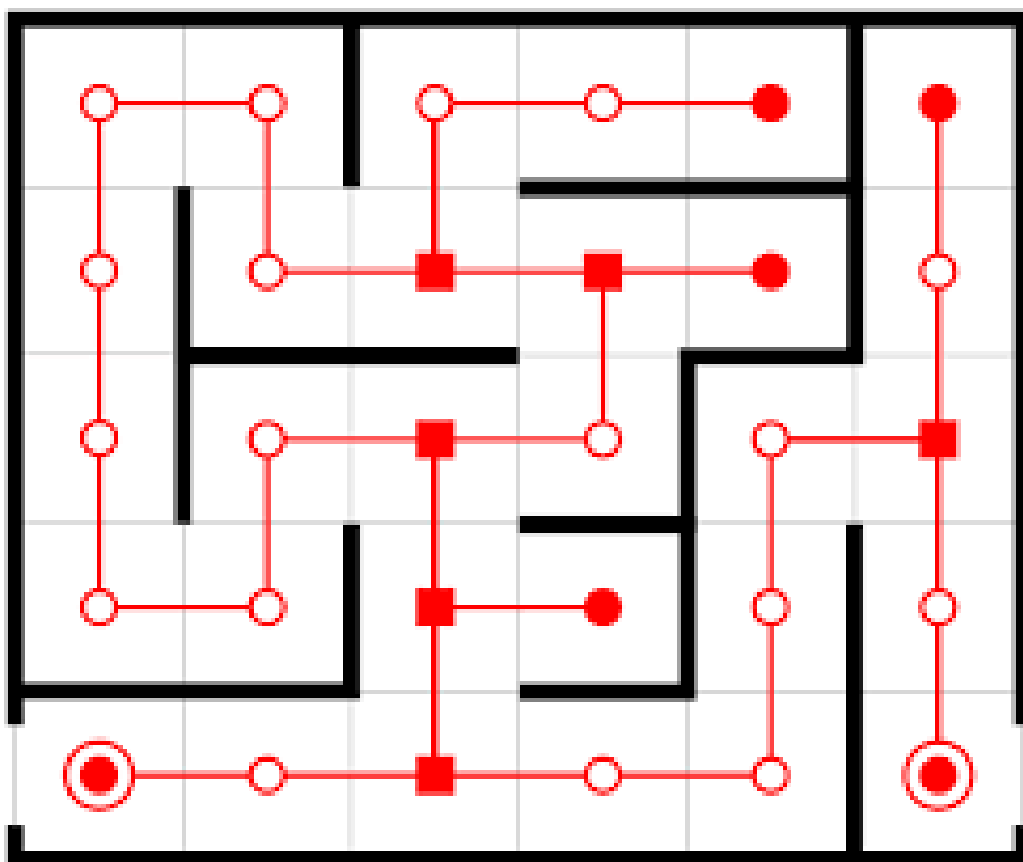


Table des matières

1	Introduction	3
2	Etat de l'art	4
3	Description des technologies	5
4	Pourquoi notre choix	6
5	Soutenance 1	7
5.1	Réseau de neurones	7
5.2	Pré-traitement	8
5.3	Interface	10
5.4	Graphes	11
5.5	Recherche de chemins	12
6	Soutenance 2	14
6.1	Réseau de neurones	14
6.2	Pré-traitement	14
6.3	Interface	15
6.4	Site web	17
7	Soutenance finale	18
7.1	Réseau de neurones	18
7.2	Pré-traitement	18
7.3	Interface	19
7.4	Site web	20
8	Répartition des taches et découpage du projet	21
9	Conclusion	23

1 Introduction

Durant le semestre précédent, le projet était un OCR sur un sudoku. Ainsi, cela nous a introduit au monde du traitement d'images ainsi que de la segmentation. Cela étant dit, la majeure partie et la plus complexe reposaient sur la conception d'un réseau de neurones qui relève de l'intelligence artificielle.

Pour ce semestre, nous avons envie d'aller plus loin dans le domaine de l'imagerie, notre but est de réaliser un logiciel de résolution de labyrinthe. A l'aide de la structure de données présentée en cours d'algorithmique : les graphes, il nous est possible d'effectuer une recherche de chemin au sein d'un graphe et ainsi de mettre en évidence le chemin reliant l'entrée et la sortie du labyrinthe.

L'intérêt de ce projet repose donc en grosse partie sur de l'algorithmie, contrairement au projet précédent reposant en majeure partie sur l'imagerie. En effet, nous avons déjà vu comment traiter une image correctement. A présent, grâce à nos nombreuses connaissances sur les graphes, nous implémenterons une manière de trouver, premièrement, un des chemins non nécessairement optimal, puis par la suite, avec l'optimisation de nos algorithmes, de trouver le plus court chemin du graphe à l'aide, par exemple, de l'algorithme de Dijkstra.

2 Etat de l'art

- Pour ce qui concerne les graphes, nos connaissances sur ce sujet sont très majoritairement celles qu'on a pu voir en cours d'algo (graphe orienté, non orienté, cyclique, connexe, complet, étiqueté, etc. parcours profondeur, largeur, etc.)

- Pour ce qui concerne, le réseau de neurones, il nous servira d'identifier les cloisons pour chaque case du labyrinthe. A priori, une couche d'entrée se décomposera en un nombre de neurones correspondant aux nombres de pixels de chaque image représentant une case du labyrinthe. Cependant, nous n'avons encore pas déterminé le nombre de couches cachées que comportera notre réseau de neurones. Enfin, la manière la plus judicieuse de représenter chaque neurone de la couche de sortie serait de déterminer toutes les combinaisons possibles de la présence d'une ou plusieurs cloisons sur la même case du labyrinthe et par la suite assigner à chacune de ces combinaisons possible un unique identifiant, probablement, sous la forme d'un numéro.

- Concernant l'interface graphique, à priori nous envisagerons d'utiliser une interface graphique propre au langage C, qui est GTK.

- Le site web : Eva a des connaissances en web qui nous serviront pour la réalisation du site web propre au projet. Gustav, de par ses projets personnels a aussi des connaissances en html/css, js.

- Le prétraitement, pourra se baser notamment sur ce qu'on a pu apprendre durant notre projet de S3 avec le projet de Sudoku. Il faudra néanmoins bien entendu l'adapter au labyrinthe donné.

- L'école, nous a initié à utiliser git et Discord, Gustav avait déjà pu utiliser ces deux outils pour des projets personnels.

3 Description des technologies

La première technologie, si on peut l'appeler technologie, est le langage C : c'est, comme exigé, en C que ce projet est prévu d'être développé. Ce langage implique, dans notre cas, l'utilisation de GCC (compilation du code C), d'un éditeur (ex : vim), et probablement aussi de GDB (débogage du programme C). Les ordinateurs sur lesquels pourra être testé le programme de ce projet sont ceux actuellement localisés en salle machine d'EPITA sur le campus Supbiotech Paris ; ils sont actuellement sous NixOS.

Pour la gestion de projet, il est prévu que nous utilisions git (nous utilisons le repo relatif mis-à-disposition par l'école) afin de pouvoir versionner notre code et afin de pouvoir le partager sans trop de "difficulté", Discord est un logiciel qui sera également probablement utilisé pour communiquer entre nous. Pour l'interface graphique, nous prévoyons l'utilisation du logiciel GTK.

Concernant le site web, nous utiliserons les technologies html/css.

4 Pourquoi notre choix

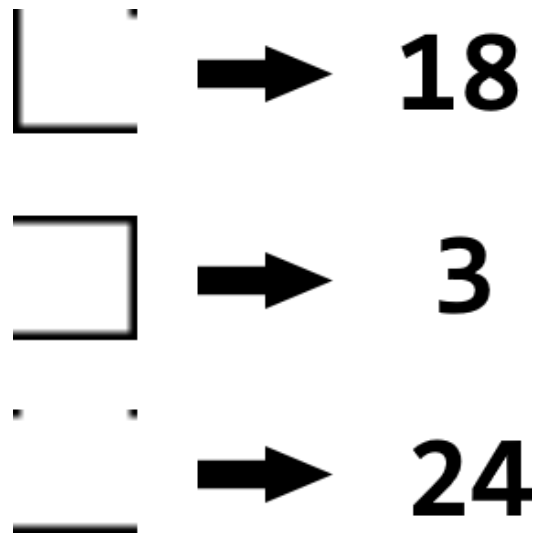
Lors du choix des technologies, le choix des graphes était évident. En effet, nous avons exploré cette structure de données en profondeur durant les cours d'algorithmique, ce qui nous procure une grande connaissance et compréhension de celle-ci. Le choix des graphes prend tout son sens une fois que l'on arrive à se représenter un labyrinthe comme un graphe dont les sommets et les arêtes représentent les chemins possibles. Une fois le labyrinthe représenté de la sorte, un simple parcours permet de trouver s'il existe le chemin reliant l'entrée et la sortie.

5 Soutenance 1

5.1 Réseau de neurones

Pour la première soutenance, nous avons déterminé qu'il était important de finaliser le réseau de neurones, tout en avançant en priorité sur la recherche du chemin et le pré-traitement de l'image. Ainsi, notre réseau de neurones est complété et renvoie dans un fichier la structure du labyrinthe sous forme de nombres correspondant aux différentes cases possibles.

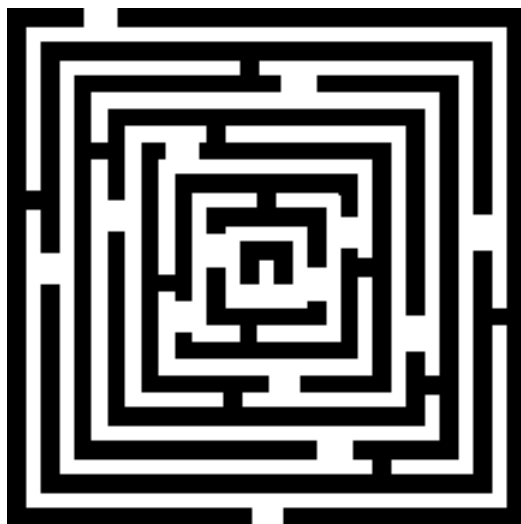
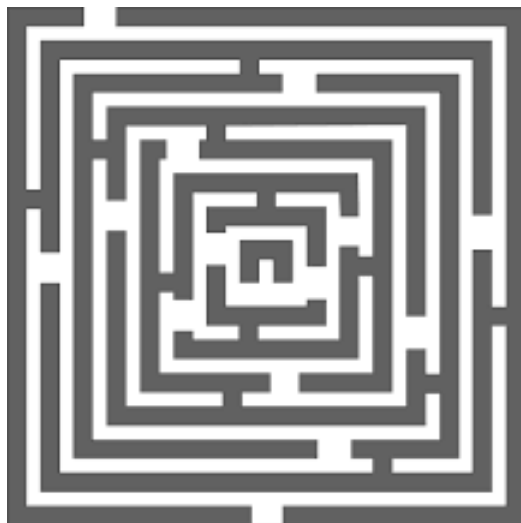
Le réseau fonctionne ainsi :



Chaque case est reconnue et est attribuée un nombre qui est ensuite utilisé pour reconnaître la case dans la recherche du chemin. Pour reconnaître les cases, on utilise un enum.

Pour la première soutenance, le pré-traitement de l'image consiste à rendre l'image en noir et blanc ou en nuances de gris puis de découper le labyrinthe en cases qui seront par la suite reconnues par le réseau de neurones. Cette détection de cases est très simple grâce à la propriété des cases et du labyrinthe d'être des carrés parfaits. Ainsi, une simple recherche de la couleur des pixels permet de déterminer la taille d'une case et de la bordure, ce qui nous permet de découper précisément.





5.3 Interface

L'interface n'étant pas notre priorité, nous avons décidé de nous concentrer principalement sur le reste. Par conséquent, nous n'avons pas d'interface interconnectée à présenter. Cependant, certains éléments du projet affichent leurs résultats dans des interfaces temporaires qui deviendront dans le futur une interface utilisateur complète pouvant accomplir toutes les fonctions proposées par notre projet.

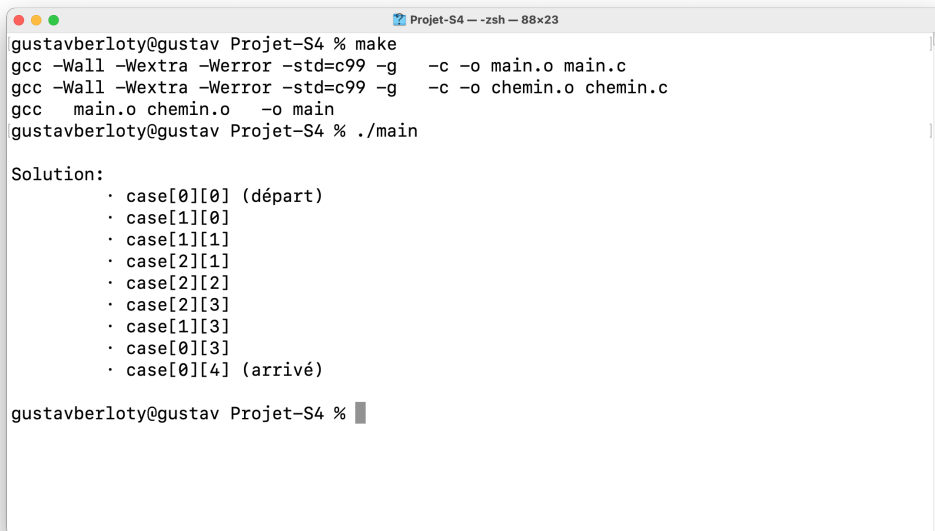
5.4 Graphes

Les graphes que nous utilisons pour représenter notre labyrinthe sont des graphes simplifiés. En effet, dans un labyrinthe, il n'y a que 4 directions possibles, ce qui nous permet d'implémenter une structure de graphe simplifiée ne possédant que 4 voisins, soit existants, soit non-existants. Il est possible de voir cette implémentation comme une liste chaînée, qui est liée non pas à 1 élément, mais jusqu'à 4 éléments.

5.5 Recherche de chemins

Nous avions prévu au départ d'implémenter une structure de graphe "classique", mais comme dit précédemment, nous utilisons finalement un graphe simplifié. Or, il s'avère que nous souhaitons utiliser un parcours profondeur tel que vu en cours d'algorithmique, pour trouver la solution du labyrinthe (un chemin amenant à l'arrivée depuis le point de départ), mais nous utilisons donc un parcours profondeur adapté aux graphe simple. On construit le chemin en "remontant" le parcours.

Voici un exemple d'exécution du programme :



```
gustavberloty@gustav Projet-S4 % make
gcc -Wall -Wextra -Werror -std=c99 -g -c -o main.o main.c
gcc -Wall -Wextra -Werror -std=c99 -g -c -o chemin.o chemin.c
gcc main.o chemin.o -o main
gustavberloty@gustav Projet-S4 % ./main

Solution:
  . case[0][0] (départ)
  . case[1][0]
  . case[1][1]
  . case[2][1]
  . case[2][2]
  . case[2][3]
  . case[1][3]
  . case[0][3]
  . case[0][4] (arrivé)

gustavberloty@gustav Projet-S4 %
```

pour le labyrinthe suivant :

A terminal window titled "Projet-S4 - zsh - 88x23" showing the command "cat maze.txt" and its output, which is a 10x10 grid of dots representing a maze. The grid is as follows:

```
. . . . .
. . . . .
.... . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
```

```
gustavberloty@gustav Projet-S4 %
```

Le labyrinthe ici représenté a été fait à la main pour tester l'algorithme de résolution.

6 Soutenance 2

6.1 Réseau de neurones

Pour la seconde soutenance, le réseau de neurones étant déjà terminé, nous n'avons pas réalisé de changements à son fonctionnement.

6.2 Pré-traitement

Pour la seconde soutenance, le pré-traitement actuel permet de répondre à toutes nos attentes afin de pouvoir appliquer notre algorithme de résolution à un labyrinthe. Nous avons donc décidé de se concentrer sur d'autres aspects du projet qui n'étaient pas encore à la hauteur de nos attentes pour ce projet.

6.3 Interface

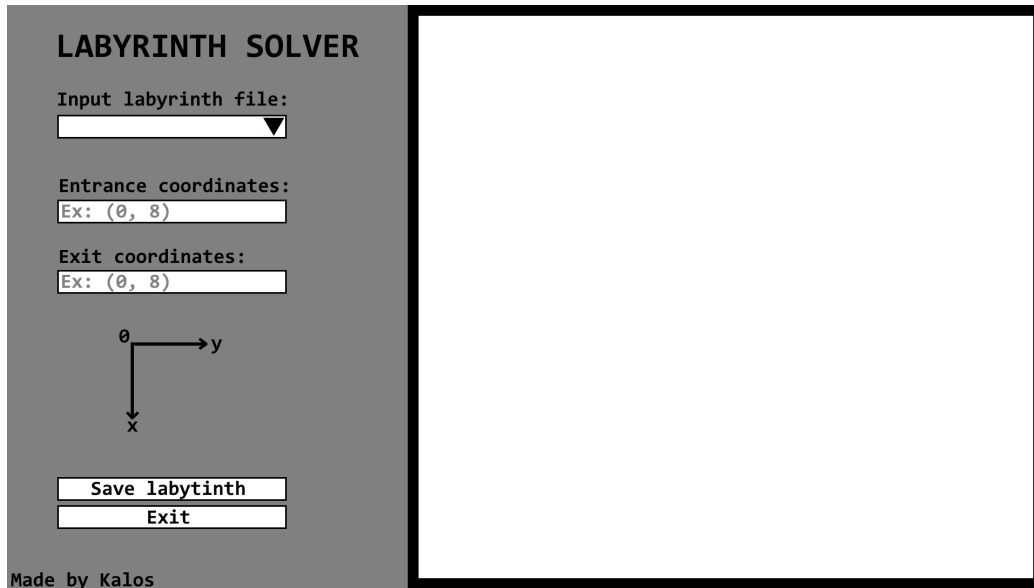
Il fut prévu que nous fassions une interface graphique pour rendre le programme plus ergonomique. GTK fut choisi pour développer l'interface.

Il s'est avéré néanmoins que l'installation de cette librairie n'a pas réussi sur 3h de temps, après avoir essayé de 3 manières différentes (apt-get, manuellement, homebrew), sur 2 systèmes d'exploitations différents (Ubuntu, macOS).

Face à une telle situation, Gustav ne veut rien avoir en commun avec GTK, ne veut pas l'utiliser ni y toucher. Cette librairie, n'est-ce pas une abomination à ses yeux ? "Il la hait" dit-il. Enfin.

C'est pourquoi, moi Gustav, je suis d'avis à cette étape du projet de ne pas faire d'interface graphique, mais plutôt une application en invité de commande. Je préfère avoir un programme qui fonctionne, un programme "béton", qui ne soit pas forcément esthétique, plutôt qu'un programme qui a été travaillé du point de vue graphique mais qui fonctionne à moitié, une fois sur trois.

C'est toutefois un peu dommage de ne pas faire d'interface graphique, Julien ayant fait une maquette tout à fait convenable :



Il est possible néanmoins que Julien veuille tester et voir par lui-même ce que vaut GTK, et ait donc en cela à coeur de développer cette interface.

Peut-être donc, malgré ce qui a été dit, qu'une interface "voit le jour" lors de la dernière soutenance ; si cela n'arrive pas, alors il est prévu que ça soit plutôt une application en invité de commande.

6.4 Site web

Le site web se veut assez minimaliste.

Sans détails de partout, de couleurs dans tous les sens.

On veut transmettre des informations pas dessiner un tableau.

Nous pensons avoir fournit avec celui-ci ce qui était demandé, à savoir :

- Une présentation du projet,
- Les liens sur les sites,
- Un download du rapport, du projet.

Seul manque peut-être un livrable light du projet.

Le site :

Projet S4

Présentation

Notre projet de S4 est un programme résolvant un labyrinthe donné ; associées au labyrinthe, une entrée et une sortie sont fournis au programme par l'utilisateur. Le programme renvoie le chemin, s'il existe, allant de l'entrée à la sortie. Le format du chemin retourné par le programme est prévu pour être *json*, afin d'être plus facilement manipulable, analysable. En entrée, les labyrinthes acceptés par le programmes suivent les règles suivantes :

1. Le labyrinthe doit être sous format *jpg* ou *png*.
2. L'image : ne contenir que le labyrinthe. Pas de parasites graphiques.
3. La qualité de l'image doit être suffisante pour que le réseau de neurones ne se trompe quant à la reconnaissance du labyrinthe (un seuil de qualité pourra éventuellement être donné par la suite).

Le fonctionnement du programme repose sur deux choses principales, un réseau de neurones et un algorithme de résolution de labyrinthe :

1. L'utilisateur fournit au programme un fichier correspondant à un labyrinthe, fichier accepté par le programme (cf. les critères à respecter ci-dessus).
2. Le programme analyse le fichier à l'aide du réseau de neurones de sorte à pouvoir indiquer à l'algorithme de résolution à quoi ressemble le labyrinthe, pour plus d'informations à ce sujet voir le rapport de soutenance.
3. L'algorithme de résolution de chemin utilise les informations données par le réseau de neurones pour trouver le chemin, s'il existe, entre l'entrée et la sortie données par l'utilisateur.
4. L'algorithme de résolution renvoie le chemin (une fois encore : s'il l'a trouvé).

Historique :

2ème soutenance (11/04/22) :

- site web complet
- avancée concernant l'expérience utilisateur

1ère soutenance (16/03/22) :

- réseau de neurone fonctionnel
- algorithme de recherche de chemin fonctionnel

Problème rencontrés :

• Librairie GTK :

Il fut prévu que nous fassions une interface graphique pour rendre le programme plus ergonomique. GTK fut choisit pour développer l'interface.

Il s'est avéré néanmoins que l'installation de cette librairie n'a pas réussi sur 3h de temps, après avoir essayé de 3 manières différentes (apt-get, manuellement, homebrew), sur 2 systèmes d'exploitations différents (Ubuntu, macOS).

Face à une telle situation, Gustav ne veut rien avoir en commun avec GTK, ne veut pas l'utiliser ni y toucher. Cette librairie, n'est-ce pas une abomination à ses yeux ? "Il la hait" dit-il. Enfin.

C'est pourquoi, moi Gustav, je suis d'avis à cette étape du projet de ne pas faire d'interface graphique, mais plutôt une application en invité de commande. Je préfère avoir un programme qui fonctionne, un programme "béton", qui ne soit pas forcément esthétique, plutôt qu'un programme qui a été travaillé du point de vue graphique mais qui fonctionne à moitié, une fois sur trois.

- Synchronisation du réseau de neurones avec l'algorithme de recherche de chemin .

7 Soutenance finale

7.1 Réseau de neurones

Depuis la première soutenance, le réseau de neurones est déjà terminé, nous n'avons pas réalisé de changements à son fonctionnement. Des simples changements ont été effectués à la structure des fichiers.

7.2 Pré-traitement

Pour la dernière soutenance, le prétraitement nous permet de répondre à toutes nos attentes afin de pouvoir appliquer notre algorithme de résolution à un labyrinthe. Il nous permet de découper notre labyrinthe en cases, pour le traiter avec le reste du projet. Des simples changements ont été effectués à la structure des fichiers.

7.3 Interface

Pour la dernière soutenance et après beaucoup de difficultés matérielles, l'interface graphique utilisateur est maintenant terminée. Cette interface est donc à présent la seule partie visuelle que l'utilisateur peut voir. Elle permet à l'utilisateur d'importer son image de labyrinthe (à condition qu'elle respecte les contraintes) et de résoudre ce labyrinthe. Des pointillés rouges apparaîtront alors pour marquer le premier chemin trouvé reliant l'entrée et la sortie. Enfin l'utilisateur a le choix de sauvegarder ou non cette image de labyrinthe résolu.

7.4 Site web

Le site web se veut assez minimaliste.

Sans détails de partout, de couleurs dans tous les sens.

On veut transmettre des informations pas dessiner un tableau.

Nous pensons avoir fournit avec celui-ci ce qui était demandé, à savoir :

- Une présentation du projet,
- Les liens sur les sites,
- Un download du rapport, du projet.

Seul manque peut-être un livrable light du projet.

Le site :

Projet S4

Présentation

Notre projet de S4 est un programme résolvant un labyrinthe donné ; associées au labyrinthe, une entrée et une sortie sont fournis au programme par l'utilisateur. Le programme renvoie le chemin, s'il existe, allant de l'entrée à la sortie. Le format du chemin retourné par le programme est prévu pour être *json*, afin d'être plus facilement manipulable, analysable. En entrée, les labyrinthes acceptés par le programmes suivent les règles suivantes :

1. Le labyrinthe doit être sous format *jpg* ou *png*.
2. L'image : ne contenir que le labyrinthe. Pas de parasites graphiques.
3. La qualité de l'image doit être suffisante pour que le réseau de neurones ne se trompe quant à la reconnaissance du labyrinthe (un seuil de qualité pourra éventuellement être donné par la suite).

Le fonctionnement du programme repose sur deux choses principales, un réseau de neurones et un algorithme de résolution de labyrinthe :

1. L'utilisateur fournit au programme un fichier correspondant à un labyrinthe, fichier accepté par le programme (cf. les critères à respecter ci-dessus).
2. Le programme analyse le fichier à l'aide du réseau de neurones de sorte à pouvoir indiquer à l'algorithme de résolution à quoi ressemble le labyrinthe, pour plus d'informations à ce sujet voir le rapport de soutenance.
3. L'algorithme de résolution de chemin utilise les informations données par le réseau de neurones pour trouver le chemin, s'il existe, entre l'entrée et la sortie données par l'utilisateur.
4. L'algorithme de résolution renvoie le chemin (une fois encore : s'il l'a trouvé).

Historique :

2ème soutenance (11/04/22) :

- site web complet
- avancée concernant l'expérience utilisateur

1ère soutenance (16/03/22) :

- réseau de neurone fonctionnel
- algorithme de recherche de chemin fonctionnel

Problème rencontrés :

• Librairie GTK :

Il fut prévu que nous fassions une interface graphique pour rendre le programme plus ergonomique. GTK fut choisit pour développer l'interface.

Il s'est avéré néanmoins que l'installation de cette librairie n'a pas réussi sur 3h de temps, après avoir essayé de 3 manières différentes (apt-get, manuellement, homebrew), sur 2 systèmes d'exploitations différents (Ubuntu, macOS).

Face à une telle situation, Gustav ne veut rien avoir en commun avec GTK, ne veut pas l'utiliser ni y toucher. Cette librairie, n'est-ce pas une abomination à ses yeux ? "Il la hait" dit-il. Enfin.

C'est pourquoi, moi Gustav, je suis d'avis à cette étape du projet de ne pas faire d'interface graphique, mais plutôt une application en invité de commande. Je préfère avoir un programme qui fonctionne, un programme "béton", qui ne soit pas forcément esthétique, plutôt qu'un programme qui a été travaillé du point de vue graphique mais qui fonctionne à moitié, une fois sur trois.

- Synchronisation du réseau de neurones avec l'algorithme de recherche de chemin .

8 Répartition des taches et découpage du projet

Bien évidemment, il n'est pas impossible que la répartition change plus ou moins légèrement au fil du projet, car il est très difficile d'accorder des tâches sans même avoir commencé. La répartition de travail envisagée est la suivante :

	Julien	Gustav	Jonas	Eva
Pré-traitement	---			
Réseau de neurones		---		---
Graphes				---
Interface			---	

	Julien	Gustav	Jonas	Eva
Pré-traitement			---	---
Réseau de neurones			---	---
Graphes			---	---
Interface			---	---

Le départ de Jonas et Eva nous a obligé de revoir notre répartition. Nous avons finalement décidé de travailler en groupe sur chaque aspect du projet plutôt que de travailler seul sur des aspects distincts.

La progression estimée du travail est la suivante :

Avancée attendue	Soutenance 1	Soutenance 2	Soutenance 3
Pré-traitement	50%	75%	100%
Réseau de neurones	100%	100%	100%
Graphes	50%	100%	100%
Interface	10%	50%	100%

La progression réalisée du projet à l'heure de la seconde soutenance est la suivante :

Avancée obtenue	Soutenance 1	Soutenance 2	Soutenance 3
Pré-traitement	75%	100%	---
Réseau de neurones	100%	100%	---
Graphes	100%	100%	---
Interface	0%	50%	---

La progression réalisée du projet à l'heure de la soutenance finale est la suivante :

Avancee obtenue	Soutenance 1	Soutenance 2	Soutenance 3
Pre-traitement	75%	100%	100%
Reseau de neurones	100%	100%	100%
Graphes	100%	100%	100%
Interface	0%	50%	100%

9 Conclusion

Ce projet regroupe à la fois la partie programmation qu'on a pu voir en cours (notamment le langage C donc), et la partie algorithmique qu'on a pu, pareillement, voir en cours (notamment les graphes donc). Néanmoins, ce projet fait aussi appel à certaines connaissances qu'on a pu acquérir nous-mêmes (ex : réseau de neurones, interface graphique et développement web). Il s'est avéré que 2 personnes du groupe n'ont plus été investis dans le projet suite à leur semestre à l'international

Pour la première soutenance, notre avancée sur le projet correspond plutôt bien à l'avancée estimée, à l'exception de l'interface graphique globale qui n'est pour l'instant pas encore réalisée. Cependant, même avec la perte de la moitié des membres de notre groupe, le projet avance avec un bon rythme et nous sommes optimistes que nous le finirons dans les temps.

À l'heure de la seconde soutenance, nous sommes globalement satisfaits de l'avancée du projet. Il regroupe actuellement toutes les fonctionnalités que l'on priorise, c'est-à-dire la reconnaissance d'un labyrinthe carre, et sa résolution à l'aide des graphes. Cependant, à cause des problèmes techniques liés à GTK, nous n'avons que la partie graphique de l'interface et pas encore de partie fonctionnelle.

Pour conclure, l'ajustement de nos exigences dû au départ de la moitié du groupe a fait en sorte que le projet avance à un rythme très promettant et nous sommes satisfaits vis-à-vis du produit final.