Projet de fin d'année

Application CherckerSolver

DUBUS Ernest

07/04/19

Table des matières

Chapitre 1

Introduction

1.1 Description du projet

Pour mon projet de fin d'année (PFE), j'ai voulu réaliser un projet sur un ton ludique. Appréciant beaucoup les jeux de plateau, je me suis dit que pouvoir recevoir des conseils en temps réel serait un concept original. C'est pourquoi j'ai décidé de développer CherckerSolver : une application d'aide au jeu de dames. Bien qu'il existe déjà des applications pour donner le meilleur coups possible, le fait de remplir manuellement la position de toutes les pièces rend la tâche relativement fastidieuse. Ainsi l'idée m'est venue de pouvoir faire cette tâche juste avec une seule photo, et mon concept porteur est né.

J'ai longtemps hésité avec le jeu de plateau que j'allais utilisé pour mon projet, et mon choix c'est porté sur le jeu de dames. Il a été motivé par sa ressemblance avec des jeux plus prestigieux comme les échecs ou le jeux de go mais qui est à la fois plus simple à représenter que les échecs et aussi plus simple que le jeu de go. Je me suis également fait la réflexion que si j'obtenais de bons résultats et que le type de projet me plaît, je pourrais toujours changer de jeux.

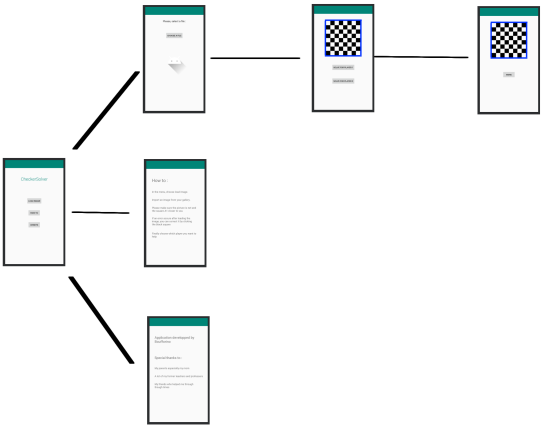
1.2 Objectifs personnels

En plus de vouloir réaliser un projet relativement ludique, j'ai également voulu réaliser certains objectifs personnels. J'ai voulu que mon sujet possède une partie de image processing car c'est une partie du programme que je souhaitais approfondir. De plus, je pense que le monde de la programmation mobile va continuer à s'étendre et je souhaitais approfondir mes connaissances dans la matière. Enfin, mon dernier objectif que je me suis fixé était de réaliser un réseau de neurones pour ensuite l'importer dans mon application mobile.

Chapitre 2

Android

Réalisant une application à but non commercial mais personnel, mais je n'ai pas réalisé une étude de marché pour savoir qu'elle cible choisir. Je voulais être en en mesure de l'utiliser et de pour réaliser des tests facilement. C'est pourquoi je me suis tourné vers une application Android en natif. La plus value de réaliser une application sur iOS est relativement limité, surtout que souhaitais utilisé TensorFlow Lite pour utiliser mon réseau de neurones sur mon application. Au final, tous les appareils Android possédant une version supérieure à 6.0 peut faire tourner mon programme.



Pour l'architecture de mon application est très simple. Au démarrage, l'utilisateur est envoyé eu menu d'accueil. Il a le choix entre une page de remerciement, une page contenant le manuel ainsi que des conseils utilisations pour obtenir des résultats optimaux ainsi que la branche principale où s'effectue le traitement de mon programme. La première de cette branche permet d'ouvrir la galerie et d'importer une image depuis celle-ci. Le bouton permettant de continuer, n'appairait que si une image est chargée pour éviter les erreurs. Vient ensuite une page de confirmation pour vérifier si il n'y a pas d'erreur. L'utilisateur peut choisir le joueur qui a besoin d'un conseil. Enfin la dernière page contient juste le meilleur coups possible calculé ainsi que un raccourci vers le menu principal.

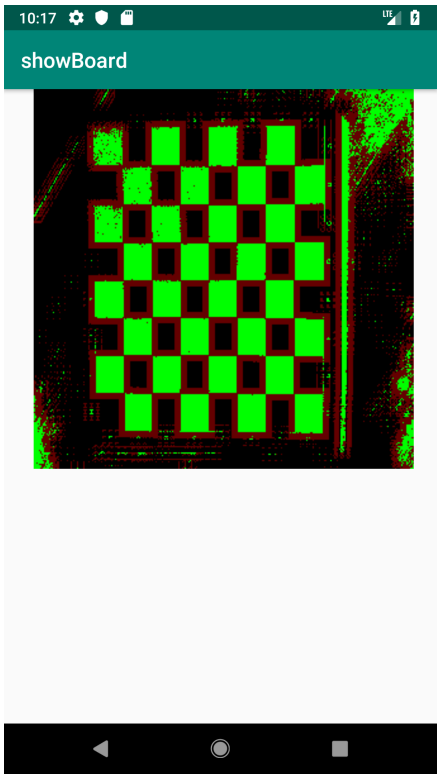
Chapitre 3

Image processing

Mon travail a été composé de trois parties. La première était de réaliser le front sur l'application Android. La deuxième est de faire le traitement l'image dans le but de reconnaître le plateau de jeu de dames et d'extraire la position des pièces . Enfin dans la dernière partie je vais expliquer les moyens que j'ai mis en place pour trouver le meilleur coups possible. Dans ce chapitre je vais vous expliquer les différents traitements que j'ai réalisé pour obtenir la position des pièces.

3.1 Pre-processing

Dans le but d'effectuer des traitements plus compliqués, il est important de préparer et nettoyer l'image. La première étape est de redimensionner l'image pour à la fois réduire la complexité et uniformiser les traitements. Ensuite vient un traitement simple pour rendre l'image en noir et blanc. Quand l'image est en binaire, il devient plus simple d'identifier le damier. Étant donné que les cases noires peut posséder des jetons qui peuvent modifier la couleur, c'est pourquoi dans un premier temps je me suis concentré sur trouver la position des case blanches pour en déduire la position des cases noires. De ce fait, j'ai fait l'option de réduire l'espace blanc en plus de réduire le bruit.



Algorithm 1 Réduire le bruit

```
i ← 0
j ← 0
for i < largeurImage do
    for j < largeurImage do
        if valeurPixel(i)(j) ==blanc then
            réduireMarge(i)(j)
        end if
    end for
end for
```

Bien que cet algorithme est simple, il permet de réduire les pixels blancs isolés. Il permet aussi qu'en moyenne les zones blanches soient un peu plus petites. En prenant les points sur les diagonales plus petits que sur les côtés, il est possible de réaliser une légère rotation des zones blanches pour essayer d'avoir des lignes plus verticales pour nous aider dans la suite.

3.2 Obtention du damier

Une fois que l'image a été préparé, on doit essayé de récupérer la position des cases du damier. Pour ce faire j'utilise une méthode en deux temps. Dans un premier temps j'effectue un algorithme récursif pour pour pouvoir récupérer

toutes les zones blanches indépendantes. On peut y effectuer des traitements pour supprimer toutes les zones qui paraissent incohérentes. (surface minime,

Une fois on possession de toutes coordonnées des zones blanches, il faut déterminer les quelles font parties du damier.

Comme l'illustre cette image, chaque coordonnée x et y de chaque case est partagée par 3 autres cases. C'est sur ce principe que je me suis basé pour réaliser ma reconnaissance du damier

Algorithm 2 Réduire le bruit

```
i ← 0
j ← 0
for all i in listeZoneBlanche do
  if auMoins4(i,liteZoneBlanche then
    caseBlanche.add(listeZoneBlanche(i))
  end if
end for
```

Avant d'aller chercher la case A1, je vérifie que je trouve suffisamment de case avec quatre coordonnées en commun.

3.3 Récupération des pièces

A l'aide des coordonnées des cases blanches je peux identifier où ce trouve la case A1 (qui est noire). A partir de celle ci, je récupère la position de toutes les autres cases grâce à la taille de la case d'origine. Il ne reste plus qu'à différentier les cases noires vides avec les cases contenant les pièces des joueurs.

Pour cela, je fais une étude des pixels moyens. J'attribue à la moyenne la plus basse la valeur noire puis j'attribue le reste au joueur respectif.

Une autre méthode que j'ai essayé était d'utiliser que les pixels d'une valeur maximum ou minimum mais malheureusement

Chapitre 4

Résolution de la partie

La dernière étape de mon application consiste à retourner le meilleur coups possible pour la position de pièce donnée. Ma première idée était de réaliser des simulations de Monte Carlo à l'intérieur de l'application.

Par la suite, j'ai décidé de réalisé un réseau de neurones en reinforcement learning. Pour ce faire j'ai reproduis le jeu de dame sur en python. Puis j'ai effectué des simulations de Monte Carlo UCB

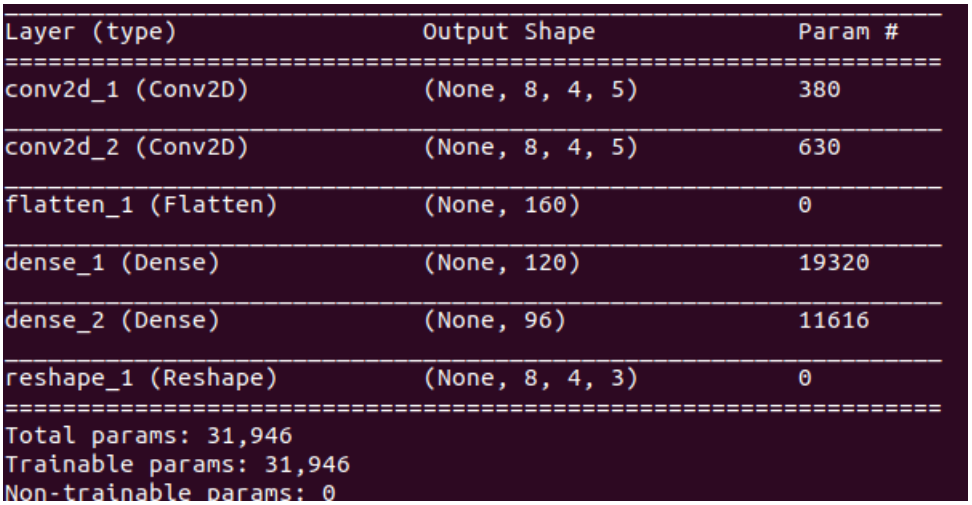
Algorithm 3 Monte Calo UCB

for i in range(10) **do**
  for j in listCoup **do**
    if partieSImulée() **then**
    resultat $\leftarrow resultat + 1$
    end if
  scoreJ $\leftarrow resultat/nbCpJj + 0.4 * (\sqrt{log(nbPartie)/nbCpJ})$
  end for
end for
for z in range(1000) **do**
  sortBy(scoreJ)
  if partieSImulée() **then**
  resultat $\leftarrow resultat + 1$
  end if
resultat $\leftarrow$ $resultat + 1 scoreJ \leftarrow resultat/nbCpJj + 0.4 * (\sqrt{log(nbPartie)/nbCpJ})$
end for

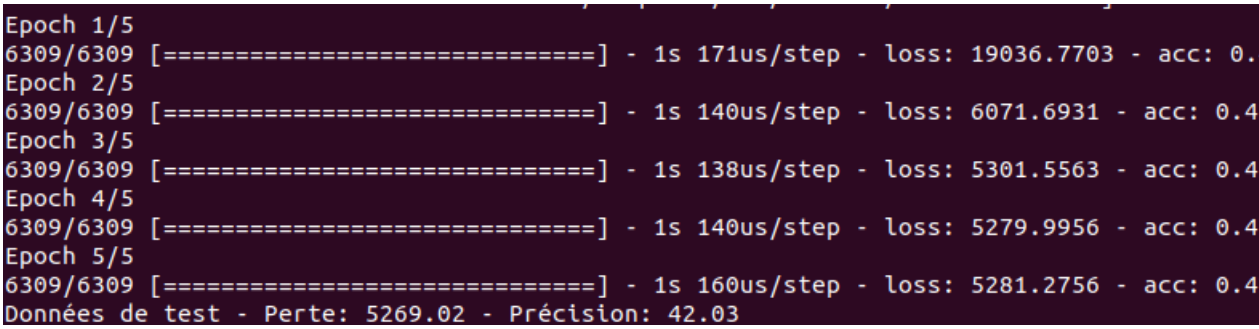
On retrouve :
— resultat : le nombre de partie gagné en faisant le coups j
— nbCpJj : le nombre de fois que le coups j a été effectué
— nbPartie : le nombre total de partie simulée
Cette version permet de se focaliser sur les branches qui sont prometteuses tout en laissant place à l'exploration.

Maintenant que j'ai un des meilleurs coups possible, j'ai utilisé le format .bpm pour enregistrer la position des pièces avant et après que le coups soit joué en utilisant une couleur par joueur et le noir pour les cases vides. J'ai simulé au total 1000 parties, et j'ai obtenu le meilleur coups possible pour plus

de 14 000 coups. J'ai ensuite séparé les images pour me faire un jeu de test . Enfin j'ai créer un réseau de neurones qui permet à partir d'une image de retourner une image réponse. Malheureusement, mon réseau ne fournit pas des résultats suffisants pour justifier son intégration dans l'application.



Les différentes couches de mon réseau de neurones



Résultat de l'entrainement de mon réseau de neurones

Chapitre 5

Conclusion

Au final je suis satisfait de l'application que j'ai réalisé. J'ai réalisé une majorité des objectifs que je me suis fixé. La partie front end sur Android m'a permis de me familiarisé sur le développement mobile. Ensuite j'ai pu manipuler des images et extraire les informations importants. Enfin, j'ai pu avancer sur le développement d'un réseau de neurones même si cela n'a pas pu fonctionner.

C'est un projet que j'aimerais sans doute continuer sur mon temps libre.