FEN Chess Generator

EL JAAFARI
Nour el houda

13/02/2021

—

Image processing

—

Mme. MIKRAM Mounia

Résumé

Ce projet met en évidence les approches adoptées pour traiter une image d'un échiquier et identifier les différentes pièces des échecs en générant la notation FEN de chaque image d'échiquier. Pour la reconnaissance des pièces, la méthode introduit une approche inspiré d'un travail sur Kaggle consistant à utiliser un R-CNN pour former un classificateur robuste à travailler sur différents types de pièces d'échiquier.

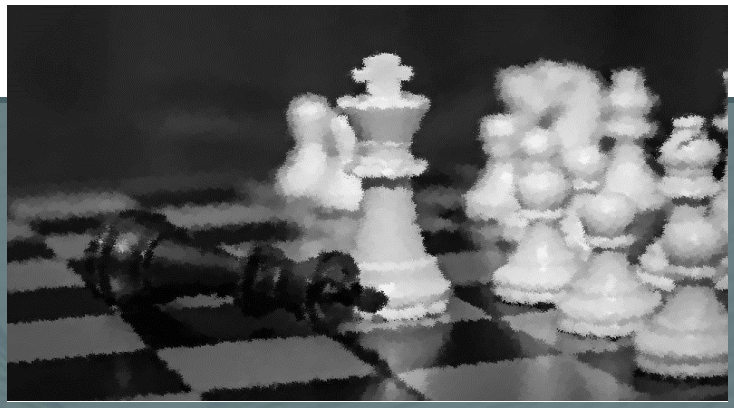


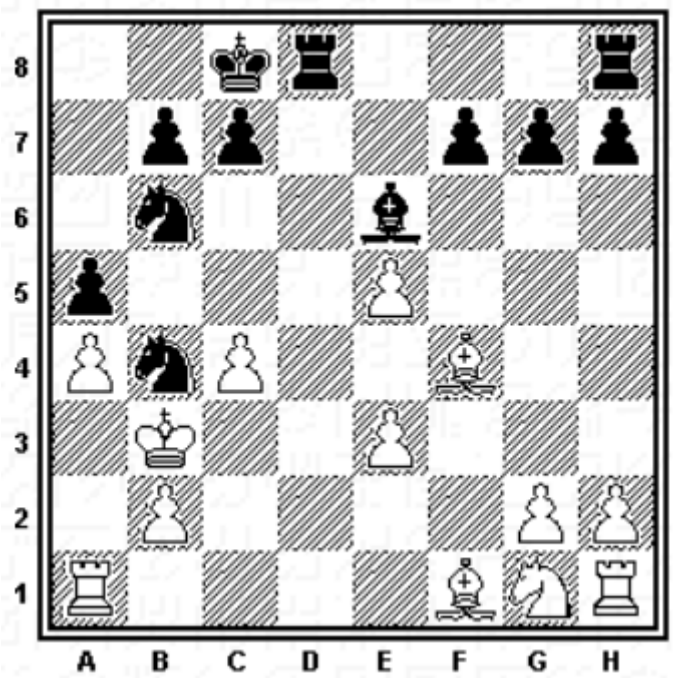
Table des matières

Introduction :

Le jeu d’échecs, ou les échecs, est un jeu de société opposant deux joueurs de part et d’autre d’un tablier appelé échiquier composé de soixante-quatre cases, 32 claires et 32 sombres nommées les cases blanches et les cases noires.

Les joueurs jouent à tour de rôle en déplaçant l'une de leurs seize pièces (ou deux pièces en cas de roque), claires pour le camp des blancs, sombres pour le camp des noirs.

FEN est l'abréviation de Forsyth-Edwards Notation. Le principe a été inventé par le journaliste écossais David Forsyth au 19ème siècle. Ce format permet de décrire la position des pièces lors d'une partie d'échecs.



Les échecs ont constitué l'un des premiers défis en matière d'intelligence artificielle. La génération de la notation FEN automatiquement à l'aide de l'image processing est un défi technologique important. Ce problème est d'un grand intérêt pour les organisateurs de tournois et les joueurs amateurs ou professionnels pour diffuser leurs parties over-the-board en ligne et les analyser.

Description du Dataset choisi :

Cette base de données contient 100k images de positions des pièces sur l'échiquier générées aléatoirement de 5 à 15 pièces.
Toutes les images de la base de données sont de taille 400x400 pixels.
Les pièces ont été générées avec la distribution de probabilité suivante :
- 30% pions
- 20% fous
- 20% cavaliers
- 20% tours
- 10% reines
- 2 rois sont assurés d'être sur l'échiquier.

La notation Forsyth-Edwards est utilisée pour labeliser les images mais avec des tirets au lieu des « / ».

Lien vers la base de données :
https://www.kaggle.com/koryakinp/chess-positions

Revue de littérature

1. Using Image Processing Techniques to Automate Chess Game Recording:

Les échecs sont un jeu de société populaire. L'enregistrement d'une partie d'échecs est important. Il permet aux joueurs de rejouer le jeu et d'améliorer leurs stratégies par des répétitions. Il permet également de partager les détails du jeu entre autres de manière compacte et sans ambiguïté. Différentes normes sont utilisées pour enregistrer une partie d'échecs. La notation la plus largement acceptée est la « notation algébrique ». Dans cette recherche, la faisabilité d'utiliser une caméra Web pour automatiser la procédure d'enregistrement et pour produire une notation algébrique est étudiée. En utilisant des techniques de traitement vidéo et d'image en temps réel, il a été démontré qu'il était possible d'automatiser la procédure d'enregistrement du jeu avec un haut niveau de précision. La texture du panneau et les conditions d'éclairage se sont avérées être les facteurs de précision les plus importants. Dans un environnement contrôlé, la précision de l'identification d'un mouvement correct était supérieure à 95%.

2. Convert a physical chessboard into a digital one:

Dans cet article, l'objectif était de construire un scanner d'échiquier qui convertit l'image d'un échiquier physique à n'importe quelle position (échecs) donnée en un échiquier numérique en générant la notation FEN qui sera utilisé pour créer l'échiquier numérique.

3. Chessboard and Chess Piece Recognition with the Support of Neural Networks:

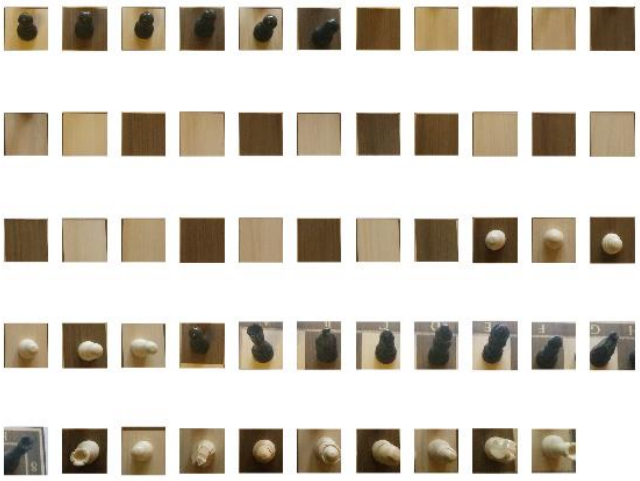
Dans ce travail l'objectif était de localiser et recadrer un échiquier dans une image, qui pourrait ensuite être utilisée pour créer un enregistrement numérique des positions des pièces d'échecs en utilisant la notation FEN, qui est la notation la plus couramment utilisée pour représenter les états des parties d'échecs. Dans un environnement contrôlé, la précision de l'identification d'une pièce dans son positionnement correct était supérieure à 90%.

4. Chess Vision: Chess Board and Piece Recognition:

Cet article détaille une méthode pour prendre une image d'un échiquier et produire une représentation reconstituée par ordinateur du tableau grâce à la reconnaissance du tableau et des pièces. Bien que les techniques de reconnaissance des échiquiers aient été minutieusement explorées dans le passé, en particulier en ce qui concerne l'étalonnage des échiquiers, les travaux précédents sur la reconnaissance des pièces se concentrent souvent sur des procédures de segmentation non robustes qui reposent fortement sur la couleur exacte des échiquiers et des pièces personnalisés. La méthode présentée dans cet article améliore les approches basées sur la segmentation des travaux antérieurs sur la reconnaissance de pièces en introduisant une nouvelle approche utilisant des classificateurs formés sur des descripteurs de caractéristiques, qui est plus robuste aux similitudes de couleur observées dans les échiquiers réels. Ce travail est important à la fois pour automatiser l'enregistrement des mouvements dans les parties d'échecs humains et pour améliorer la capacité de l'IA jouant aux échecs qui dépend de la vision.

5. Chess Recognition Using Computer Vision

Plutôt que d’entrainer un réseau à partir de zéro, nous utilisons l'apprentissage par transfert en commençant par un réseau préformé. Cette approche économise beaucoup de temps et d'efforts et fonctionne sur le principe du réglage du réseau à l'aide d'images nouvellement alimentées



Le modèle utilisé

Notions de bases :

- **Keras** :

Keras est une bibliothèque open source écrite en Python (sous licence MIT) basée principalement sur les travaux du développeur de Google François Chollet dans le cadre du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Une première version du logiciel multiplateforme a été publiée le 28 mars 2015. Le but de cette bibliothèque est de permettre la constitution rapide de réseaux neuronaux. Dans ce cadre, Keras ne fonctionne pas comme un framework propre mais comme une interface de programmation applicative (API) pour l'accès et la programmation de différents frameworks d'apprentissage automatique. Theano, Microsoft Cognitive Toolkit (anciennement CNTK) et TensorFlow font notamment partie des frameworks pris en charge par Keras.

Depuis la sortie de TensorFlow 1.4, Keras fait partie intégrante de l'API de base de TensorFlow. Toutefois, la bibliothèque est toujours développée comme un logiciel indépendant, car l'approche initiale visant à l'utiliser comme une interface pour différents frameworks est toujours d'actualité.

- **Tensorflow** :

Développé par les chercheurs de Google, TensorFlow est un outil open source d'apprentissage automatique (machine learning), d'apprentissage profond et d'analyses statistiques et prédictives. A l'instar de plateformes similaires, il vise à rationaliser le développement et l'exécution d'applications analytiques avancées destinées aux data-scientists, statisticiens et modélisateurs prédictifs.

Le notebook du projet :

On commence par importer les bibliothèques nécessaires :

```
[ ]  import matplotlib.pyplot as plt
     import pandas as pd
     import numpy as np
     import os
     import cv2
     import keras
     import seaborn as sns
     from keras.models import Sequential
     from keras.layers import Dense
     from random import sample
     from keras.models import load_model
     from time import time
     from sklearn.metrics import classification_report
     from keras.layers import Dense, Dropout, Activation, Flatten
     from keras.layers import Conv2D, MaxPooling2D
     from sklearn.model_selection import KFold
```

On charge la base de données :

```
[ ]  file=os.listdir('/content/drive/MyDrive/dataset/train')
     test=os.listdir('/content/drive/MyDrive/dataset/test/')
```

Définir une fonction de transformation entre les labels et les nombres :

- Utiliser 0 pour représenter le fou blanc ;
- Utiliser 12 pour un bloc vide.

```
[ ]  def tran(t):
         T={'B':0,'b':1,'K':2,'k':3,'Q':4,'q':5,'R':6,'r':7,'P':8,'p':9,'N':10,'n':11,'F'
         return T[t]

     def tran_t(t):
         T={0:'B',1:'b',2:'K',3:'k',4:'Q',5:'q',6:'R',7:'r',8:'P',9:'p',10:'N',11:'n'}
         return T[t]
```

Définir une fonction qui génère la notation FEN :

```
[ ]  def make_name(y):
         str1=''
         count=0
         for n in range(64):
             if n!=0 and n%8==0:
                 if count!=0:
                     str1+=str(count)
                 count=0
                 str1+='-'
             if y[n]==12:
                 count+=1
             else:
                 if count!=0:
                     str1+=str(count)
                 count=0
                 str1+=tran_t(y[n])
             if n==63 and count!=0:
                 str1+=str(count)
         return str1+'.jpeg'
```

Une fonction pour diviser l'image en 64 petits blocs :

```
[ ]  def read_picture(name,location='train'):
         img = cv2.imread('/content/drive/MyDrive/dataset/'+location+'/'+name,cv2.IMREAD_GRAYSCALE)
         name_t=name.split('.')[0] #Convert FEN code to labels
         po=name_t.split('-')
         index=np.zeros((8,8))

         for n in range(8):
             temp=[]
             for t in po[n]:
                 if t>='1' and t<='8':
                     for num in range(int(t)):
                         temp.append(int(12))
                 elif t>='A' and t<='z':
                     temp.append(tran(t))
             index[n,:]=np.intc(temp)

         size=50 #Divide the picture into 64 pieces
         index2=np.zeros((64,size**2))
         for i in range(8):
             for j in range(8):
                 index2[i*8+j,:]=np.array(img)[i*size:(i+1)*size,j*size:(j+1)*size].reshape(1,size**2)

         y_1=np.zeros((64,13))
         y_1[range(64),index.reshape(1,-1)[0].astype('int64')]=1

         return index2,y_1
```

Réaliser le modèle CNN en utilisant Keras :

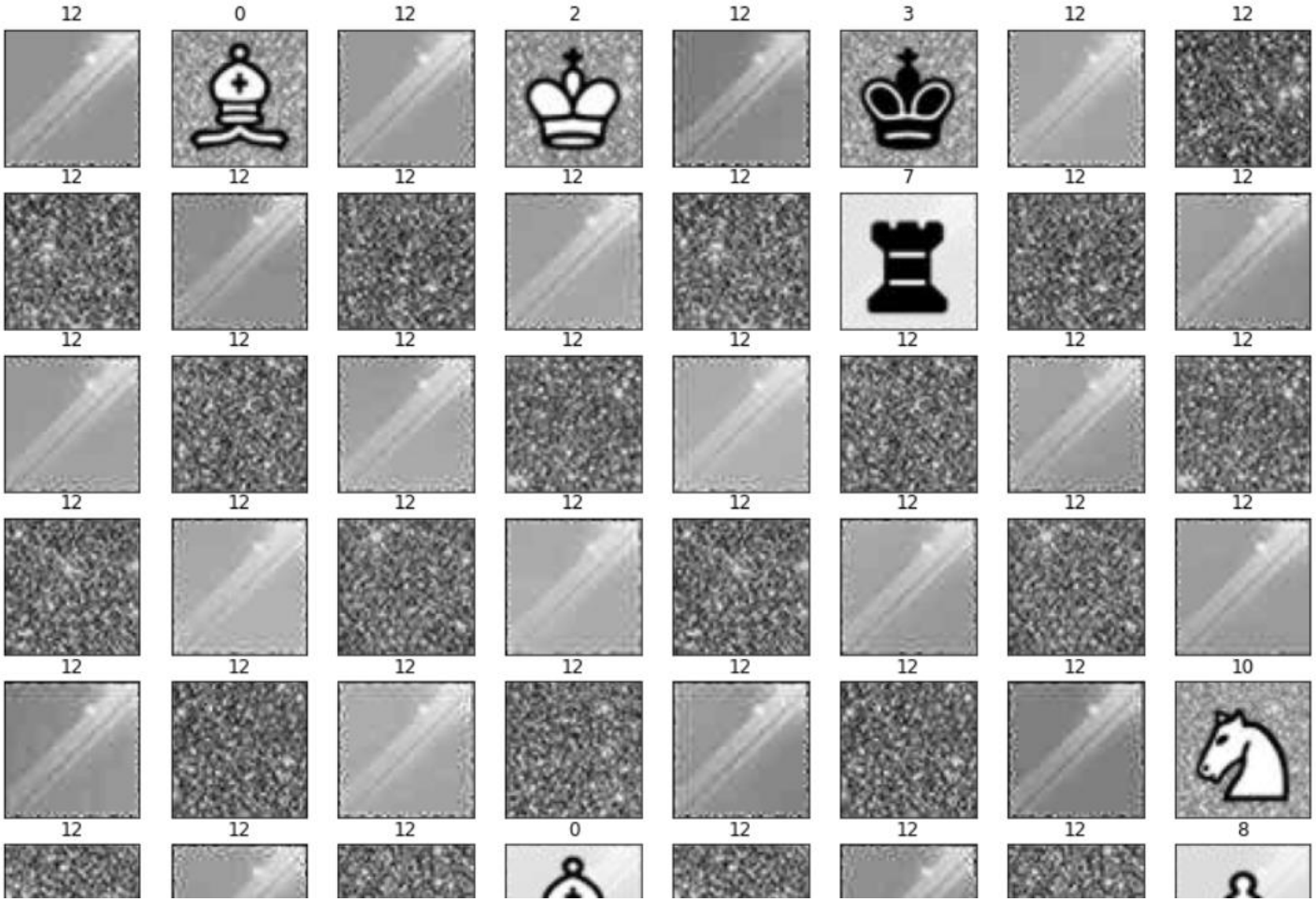
```
[ ]  def cnn_build():
         cnn = Sequential()
         cnn.add(Conv2D(32, (3, 3), padding='same',input_shape=(50,50,1)))
         cnn.add(Activation('relu'))
         cnn.add(Conv2D(32, (3, 3)))
         cnn.add(Activation('relu'))
         cnn.add(MaxPooling2D(pool_size=(2, 2)))
         cnn.add(Dropout(0.25))
         cnn.add(Conv2D(64, (3, 3), padding='same'))
         cnn.add(Activation('relu'))
         cnn.add(Conv2D(64, (3, 3)))
         cnn.add(Activation('relu'))
         cnn.add(MaxPooling2D(pool_size=(2, 2)))
         cnn.add(Dropout(0.25))
         cnn.add(Flatten())
         cnn.add(Dense(512))
         cnn.add(Activation('relu'))
         cnn.add(Dropout(0.5))
         cnn.add(Dense(13))
         cnn.add(Activation('softmax'))
         rms = keras.optimizers.RMSprop(lr=0.00005, decay=1e-6)
         cnn.compile(loss='categorical_crossentropy',optimizer=rms,metrics=['accuracy'])
         return cnn
```

Tester à partir d'un exemple d'une image : `1B1K1k2-5r2-8-8-7N-3B3P-P2p1q2-5Q2.jpeg`

```
#Example of reading a certain
name='1B1K1k2-5r2-8-8-7N-3B3P-P2p1q2-5Q2.jpeg'
img = cv2.imread('/content/drive/MyDrive/dataset/train/'+name,cv2.IMREAD_GRAYSCALE)

plt.imshow(img.reshape(400,400),cmap=plt.cm.gray)
plt.xticks(())
plt.yticks(())

index2,y_1=read_picture(name,location='train')

plt.figure(figsize=(15, 15))
plt.suptitle("1B1K1k2-5r2-8-8-7N-3B3P-P2p1q2-5Q2", size=16)
for i in range(64):
    plt.subplot(8, 8, i+1)
    plt.imshow(index2[i,:].reshape(50,50),cmap=plt.cm.gray)
    plt.title(np.argmax(y_1,axis=1).reshape(1,-1)[0][i])
    plt.xticks(())
    plt.yticks(())
```

L'output :

1B1K1k2-5r2-8-8-7N-3B3P-P2p1q2-5Q2



Calculer l'accuracy du modèle et le temps écoulé :

```
Time=[]
size=50
loss=[]
acc=[]
start=time()

cnn=cnn_build()

for ep in range(60):

    chess=np.zeros((1,size**2))
    label=np.zeros((1,13))

    for name in sample(file,1):  #Randomly select one picture

        index2,y_1=read_picture(name,location='train')

        X_noempty=index2[y_1[:,12]==0] #Balance the number of empty blocks and the pieces
        y_noempty=y_1[y_1[:,12]==0]
        X_empty=index2[y_1[:,12]==1][0:len(y_noempty)//2,:]
        y_empty=y_1[y_1[:,12]==1][0:len(y_noempty)//2,:]

        chess=np.concatenate((chess,X_noempty,X_empty), axis=0)
        chess=chess/255 #Do the normalization
        label=np.concatenate((label,y_noempty,y_empty))

    # Train CNN
    train_history=cnn.fit(chess[1:,].reshape(len(chess)-1,50,50,1), label[1:,], epochs=10, batch_size=10,verbose=False)
    #acc.append(train_history.history['acc'][-1])
    #loss.append(train_history.history['loss'][-1])
```

```
#Test on test set
y_res=[]
y_res_class=np.zeros((1,1))
y_ture=np.zeros((1,1))
for name in test:

    chess2,y_1=read_picture(name,location='test')
    chess2=chess2/255
    y_pre=np.argmax(cnn.predict(chess2.reshape(len(chess2),50,50,1)),axis=1)
    y_res.append(make_name(y_pre))

    y_res_class=np.concatenate((y_res_class,y_pre[:,np.newaxis]))
    y_ture=np.concatenate((y_ture,np.argmax(y_1,axis=1)[:,np.newaxis]),axis=0)

Classifi_R=classification_report(y_ture[1:,], y_res_class[1:,],output_dict=True)
A=np.mean(np.array(y_res)==np.array(test))


Time.append(time()-start)

print('Accuracy:',A)
print('Time used:',Time[0])
```

```
[ ]  print(classification_report(y_ture[1:,], y_res_class[1:,],output_dict=False))
```

	precision	recall	f1-score	support
0.0	0.73	0.92	0.82	600
1.0	1.00	0.50	0.67	601
2.0	0.80	0.89	0.84	680
3.0	0.70	0.81	0.75	680
4.0	0.13	0.86	0.22	281
5.0	1.00	0.24	0.39	307
6.0	0.84	0.65	0.73	618
7.0	0.74	0.84	0.79	618
8.0	0.94	0.75	0.84	678
9.0	0.77	0.89	0.82	619
10.0	0.83	0.88	0.85	595
11.0	0.99	0.60	0.75	590
12.0	1.00	0.97	0.98	36653
accuracy			0.93	43520
macro avg	0.81	0.75	0.73	43520
weighted avg	0.97	0.93	0.94	43520