# Chess-Pieces-Detection

Ce projet vise à développer un modèle d'object detection capable d’identifier les différentes pièces d’échecs à partir d'images. Le modèle apprend à distinguer chaque type de pièce (roi, dame, etc.) afin de faciliter l’analyse de parties ou l'automatisation de la reconnaissance d'états de jeu.

## Technologies Utilisées

Python (version 3.8.6) \

TensorFlow (version 1.15) \

Keras (version 2.3.1) avec tensorflow.compat.v1 \

Mask R-CNN \

Skimage \

Scikit-learn

## Hyper Paramètres

Les principaux hyperparamètres utilisés dans l'entraînement sont les suivants :

BACKBONE : resnet101 \

IMAGES\_PER\_GPU : 1 \

NUM\_CLASSES : 7 \

STEPS\_PER\_EPOCH : 250 \

VALIDATION\_STEPS : 50 \

LEARNING\_RATE : 0.001 \

DETECTION\_MIN\_CONFIDENCE : 0.9 \

USE\_MINI\_MASK : False \

IMAGE\_MIN\_DIM : 800 \

IMAGE\_MAX\_DIM : 1024 \

RPN\_ANCHOR\_SCALES : (32, 64, 128, 256, 512) \

RPN\_ANCHOR\_RATIOS : [0.5, 1, 2] \

TRAIN\_ROIS\_PER\_IMAGE : 200 \

BATCH\_SIZE : 1

## Données

Le projet utilise un ensemble de données de pièces d'échecs comprenant :

Nombre total d'images : 581 \

Images d'entrainement : 461 \

Images de validation : 120 \

## Structure des Données

Les images et les annotations sont organisées de la manière suivante :

train/: Contient les images et annotations au format JSON pour l'entraînement. \

val/: Contient les images et annotations au format JSON pour la validation.

## Entrainement

Nous avons lancé notre entrainement avec 11 epochs de 250 pas chacune.

Notre entrainement nous affiche ces valeurs de pertes à la fin de la dernière epoch : \

<img width="627" alt="Capture d’écran 2025-04-28 à 11 43 11" src="https://github.com/user-attachments/assets/da6003d1-d22f-4380-b1a4-d62b91aba3da" />

## Visualisation avec données de validation

Après l'entrainement, nous avons lancé des prédictions sur nos données de validation. Avec un DETECTION\_MIN\_CONFIDENCE initialement paramétré à 0.9, les résultats n'étaient que très peu concluants. Après avoir testé plusieurs valeurs, nous avons retenu un DETECTION\_MIN\_CONFIDENCE à 0,6, qui nous renvoyait des résultats plus corrects sans être majoritairement erronés. Vous trouverez ci dessous la visualisation de la prédiction sur plusieurs pièces tirées au hasard dans nos données de validation :

<img width="213" alt="Capture d’écran 2025-04-28 à 11 45 03" src="https://github.com/user-attachments/assets/8b421bad-ec6c-41b6-9a29-3c2763edfb52" />

<img width="213" alt="Capture d’écran 2025-04-28 à 11 45 23" src="https://github.com/user-attachments/assets/c7b6163f-a6ef-4e24-97f0-40b609830644" />

<img width="185" alt="Capture d’écran 2025-04-28 à 11 45 40" src="https://github.com/user-attachments/assets/2450354e-6830-43a7-8fb5-eedf207619f1" />

<img width="143" alt="Capture d’écran 2025-04-28 à 11 46 05" src="https://github.com/user-attachments/assets/6f44e19b-3399-4eb6-8a77-4a7bf4e1b393" />

<img width="200" alt="Capture d’écran 2025-04-28 à 11 46 26" src="https://github.com/user-attachments/assets/5be40f75-d6ec-41f0-93e3-cd339cdea604" />

<img width="196" alt="Capture d’écran 2025-04-28 à 11 46 51" src="https://github.com/user-attachments/assets/d9c2dbf2-2cf5-4d39-8597-8cb0dd0f7033" />

Bien que la prédiction soit parfois éronnée ou inexistante à ce niveau de DETECTION\_MIN\_CONFIDENCE pour certaines images, la prédiction reste dans l'ensemble correcte.

## Résultats

Graphique de pertes

Nous tirons de notre entrainement les valeurs des pertes (loss) de chaque epoch exécutée, grâce auxquelles nous construisons le graphique de perte :

<img width="309" alt="Capture d’écran 2025-04-28 à 11 49 33" src="https://github.com/user-attachments/assets/9cdaad34-6137-403b-9c57-3df60cc678a8" />

<img width="491" alt="Capture d’écran 2025-04-28 à 11 50 02" src="https://github.com/user-attachments/assets/80fe0945-0654-4f24-96ff-bf834575b867" />

## Matrice de confusion

Nous tirons de notre modèle la matrice confusion suivante :

<img width="360" alt="Capture d’écran 2025-04-28 à 11 50 41" src="https://github.com/user-attachments/assets/e6a77349-d930-4c2a-b47b-71b3e7cf6325" />

On remarque que le modèle ne reconnaît pas les différentes pièces avec la même précision : le cavalier est la pièce la mieux reconnue et n'est confondu avec aucune autre pièce. Cela peut s'expliquer par la forme spécifique du cavalier, qui est relativement singulière et très reconnaissable. Le fou est la pièce la moins reconnaissable par notre modèle, souvent confondu avec le fond (background) ou les pions. Le fait que des pièces comme le pion ou le roi soient également confondues avec d'autres pièces peut s'expliquer par le fait que certaines classes de pièces partagent des caractéristiques similaires en termes de design.

## Conclusion et possibles améliorations

Nous pouvons conclure que notre modèle semble bien fonctionner pour certaines classes de pièces, mais fonctionne de manière médiocre pour d'autres. Plusieurs améliorations sont envisageables pour l'optimiser : entre autres, recueillir davantage de données d'entraînement, en particulier pour les classes où les confusions sont les plus fréquentes, et ajuster les hyperparamètres tels que le nombre d'epochs ou le nombre de pas. La réduction de DETECTION\_MIN\_CONFIDENCE lors de l'entraînement aurait également pu être envisagée, une valeur trop élevée ayant peut-être poussé notre modèle à ignorer certaines sélections.