Haute École d'Ingénierie et de Gestion du Canton de Vaud

Gestion de projet de Machine Learning

GML

Crapauduc - 2022

Authors:
Schaller Joris
D'Ancona Olivier
Logan Victoria
Akoumba Erica Ludivine
Wichoud Nicolas

December 19, 2022

Contents

Chapter 1

Introduction

Ceci est un acronyme k-nearest neighbor

Chapter 2

Context

2.1 Location

2.2 Cameras

Ceci est un acronyme k-nearest neighbor

Chapter 3

Data preparation

```
    start_val = 1000000
    start_date = '2009-01-01'
    end_date = '2011-12-31'
    symbols = ['SPY','XOM','GOOG','GLD']
    allocs = [0.4, 0.4, 0.1, 0.1] # @ beginning, 40% to SPY, 40% to
        XOM, etc
```

Chapter 4

Filtrage

Idée générale Le but de ce chapitre est de décrire les différentes méthodes de filtrage utilisées pour améliorer la qualité des données.

Problème Le dataset original est composé de 18 caméras regroupant plus de 800'000 images. Une bonne partie de ces images sont des faux positifs. Il est donc nécessaire de filtrer les images afin de ne garder que les images qui nous intéressent. Une première observation nous fait remarquer que les images uniquement constituées de feuilles n'ont jamais d'animaux. Ensuite, une deuxième lecture nous fait remarquer que les animaux se déplacent plus facilement par temps humide. Et finalement, nous constatons que les animaux sont nombreux certains jours. À partir de ces observations, nous avons élaboré 3 méthodes pour filtrer les images et ainsi augmenter notre probabilité de trouver des animaux pour constituer de nouveaux labels ou constituer un dataset de validation. Ces méthodes sont décrites dans les sections suivantes.

4.1 Analyse de la météo

4.2 Analyse du nombre d'animaux

4.3 Détecteur de planches

Nous avons développé un réseau de neurones convolutif à l'aide de la libraire PyTorch. Ce classificateur binaire, prédit ou non la présence de planche.

Dataset d'entraînement Nous avons extrait 600 images d'une même caméra et labellisé 359 non planches et 241 planches. Ensuite, nous avons développé un dataloader permettant d'intégrer nos labels et de charger des batchs de données directement dans la libraire PyTorch. Celui ci, utilise un pipeline d'entrée qui applique plusieurs transformations à l'image avant de pouvoir l'utiliser comme un tenseur.

Architecture du Détecteur Le détecteur est simplement constitué de 3 couches convolutives suivi de 2 couches entièrement connectées. Les channels d'entrée et de sortie des couches convolutives est de : 3 - 32, 32 - 64, 64 - 128. Le nombre de neurones des couches fully connected sont de 128 et 1 pour le neurone de sortie. La fonction de coût utilisé est

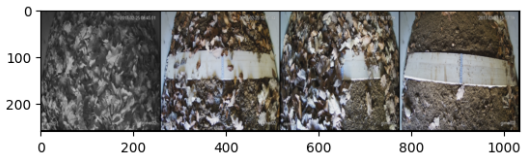

Figure 4.1: Exemple de données d'entraînement

la BCELoss et l'optimiseur est Adam. Le réseau est entraîné pendant 10 epochs avec un learning rate de 0.001 et un momentum de 0.9 sur 3 epochs.

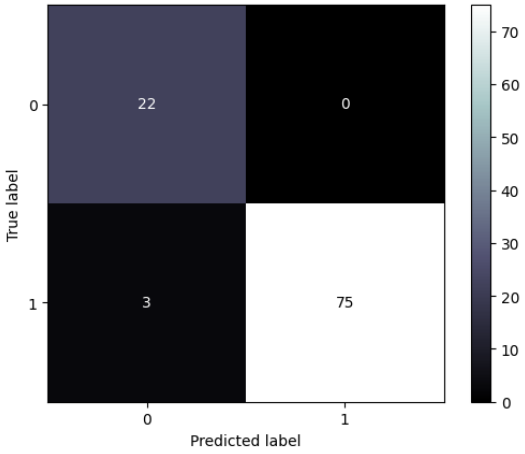

Figure 4.2: Matrice de confusion du détecteur de planche

Résultats Le détecteur de planche a une précision de 1 et un recall de 0.98 sur la détection de planche. En revanche, la précision sur la détection de non planche est de 0.88 et un recall de 1. Ce qui veut dire que notre filtre est un peu trop efficace et a tendance à se tromper pour détecter les images sans planche. Comme les résultats sont satisfaisant pour dégrossir le travail, nous n'avons pas passé de temps supplémentaire à optimiser le réseau afin qu'il sépare mieux les images dotés d'une planche ou non. Comme, nous traitons une grande quantité de données, l'erreur est acceptable. Lancé sur la quasi intégralité du dataset, le filtre a tourné pendant plus de 10h sur un ordinateur de bureau doté d'un processeur Ryzen9500X. Au final, le filtre a détecté 48910 images de non planches sur les 754543 images analysées.

Chapter 5

Models

5.1 Model1

description

5.1.1 Training

5.2 Model2

description

5.2.1 Training

Chapter 6

Evaluation

6.1 Section1

paragraph1

6.1.1 Features

Ceci est un acronyme k-nearest neighbor

Chapter 7

Deployment

7.1 Section1

paragraph1

7.1.1 Features

Chapter 8

Conclusion

Ceci est un acronyme k-nearest neighbor