ENSEIRB-MATMECA

RAPPORT DU PROJET : LET'S SHOP !
ENTRETIEN TECHNIQUE

- Projet - LET'S SHOP -

Authors
Thibaut ROBINET

13 January 2022

Table des matières

Résumé

Let's shop, est un projet qui m'a été proposer pour un entretien technique d'empauche. Il vise a mettre en place une approche machine learning, pour predire des actions grâce à la vidéo surveillance. Il faut également mettre en place une API REST permettant de d'uploader et d'évaluer une vidéo. En effet, il est explicitement attendu que je documente mon projet. J'ai rédigé un 'readme' et j'ai ajouter une page de documentation de l'API pour l'aspect utilisation. Ce rapport existe afin de documenter les choix qui ont été fait lors de ce projet.

1 Introduction

Ce rapport vise à présenter les choix que j'ai effectué et à rassembler les problèmes auquels j'ai été confronté. Afin de répondre aux objectifs qui sont d'entrainer un model puis de le déployer dans les délais imparti, un grand nombre des choix ont été grandement influencé par le temps que j'avais a y consacré. En effet, j'ai eu une semaine pour effectué ce projet, pendant une période de charge de travail scolaire importante. Ce n'est pas une excuse mais un argument du temps va apparaître régulièrement dans la justification des choix qui ont été fait.

2 Le modèle

2.1 Choix du modèle

Après avoir lu les articles, j'en suis arrivé à la conclusion suivante : [3] est le plus ancien et semble est le plus facile à mettre en place. [1] est le plus récent, et j'ai bien aimé l'approche du modèle 2-streams.

Les modèles de 3DConv sont basés sur un imagenet pré-entraîné sont des inception-VI et j'ai trouvé un dépot github [5] avec une partie du code pour entraîner et évaluer ce modèle Inception-VI.

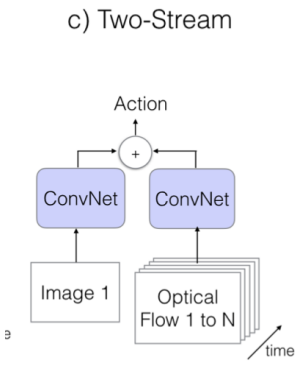

FIGURE 1 – Architecture Two-streams

2.2 Implémentation

Je suis donc parti de ce code comme point de départ. Je me suis d'abord attelé a le faire fonctionner sur un dataset contenant une seule image RGB. Les résultats n'étaient pas très bons . Et je ne pouvais pas faire beaucoup

d'epoch sur ma machine (trop long). Je me suis rendu compte qu'on pouvait gagner du temps en delocalisant le resize des images dans le pre-traitement. ET j'ai diminué le nombre de couche du model Inception-VI pour encore une fois gagner du temps.

2.3 Performances

Une fois, cette partie fonctionnel, j'ai modifié le model pour recevoir une 2^{me} input : le flux optique. Les résultats semblent nettement meilleur, même si je n'ai pas pu l'entrainé sur plus de 2 epoch, pour des question hardware. * Accuracy 37.500 et Loss 1.659 sur le dataset de test. Voici la matrice de confusion que j'ai obtenu sur le dataset de test :

On y voit clairement les defaults sur causé par le manque d'entrainement, seulent 2 classes sont réellement prédites :

```
[[0.33333333 0.         0.         0.         0.         0.66666667]
 [0.4        0.         0.         0.         0.         0.6       ]
 [0.4        0.         0.         0.         0.         0.6       ]
 [0.5        0.         0.         0.         0.         0.5       ]
 [0.58333333 0.         0.         0.         0.         0.41666667]
 [0.56666667 0.         0.         0.         0.         0.43333333]]
```

FIGURE 2 – Matrix de confusion sur le dataset de test

3 Le pré-traitement

J'ai lu en téléchargement le dataset, qu'ils on procéder sur des vidéos à 15 fps. Donc je commence le pré-traitement par une réduction des vidéos de 30 fps à 15 fps Ensuite, j'enregistre 1 frame sur 6 pour l'input 1. Enfin je converti les frames en nuances de gris pour pouvoir calculer le flux optique sur 6 frames. Ennfin j'enregistre les 3 flux en 2 images (dimension 6) que j'ai recentré autour de la valeur 128 pour ne pas avoir des pixel négatifs, pour constituer le dataset de l'input 2 du modèle.

4 Back-End

Un objectif du sujet est de prendre en compte la scalabilité. Cela m'a poussé à tout dockerisé le Back-End. En effet, grâce à une solution telle que Kubernetes, il serait possible de faire monté en charge le service. Je n'ai cependant pas eu le temps de monté de cluster Kubernetes.

4.1 Architecture

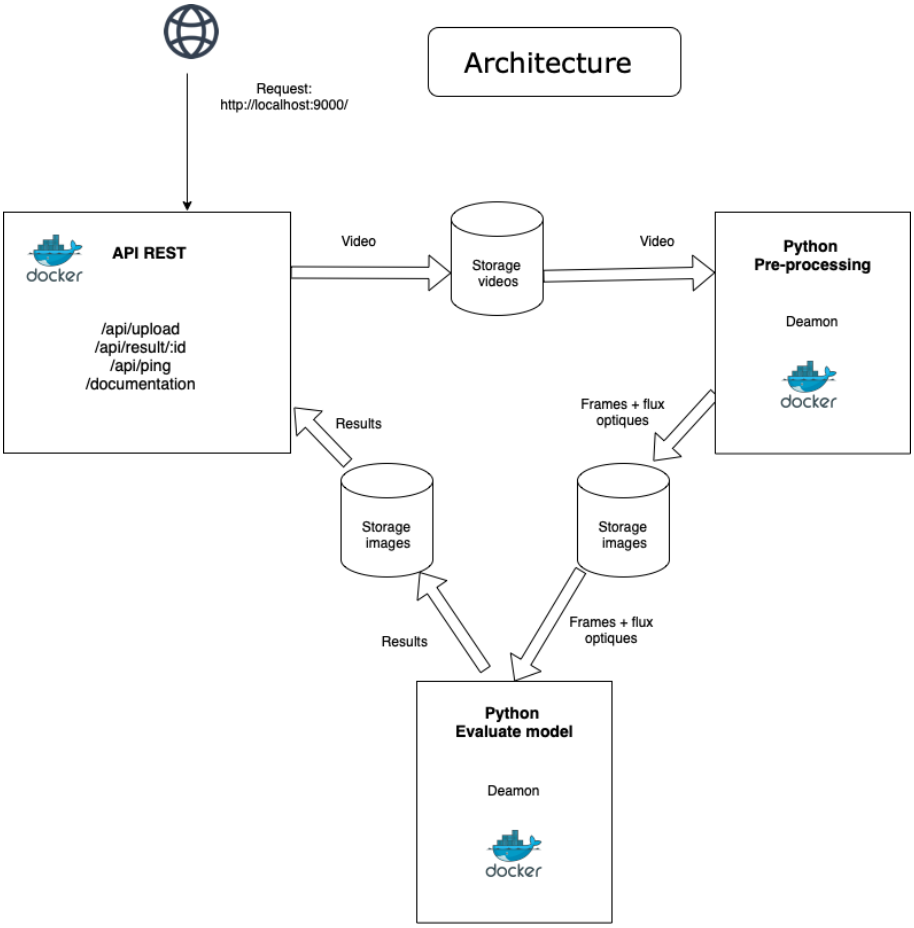

FIGURE 3 – Architecture Two-streams

4.2 API REST

J'ai fais le choix de faire une API en typescript + Express car j'en avait déja fait une au paravant. J'ai dockerisé cette API qui comporte 4 routes, qui sont documentés sur la page 'http ://localhost :9000/documentation'. Parmis les 4 routes, deux sont là pour pouvoir faire des pings sur l'api : '/api/ping' et '/api/version'. Les deux autres sont celles du service, '/api/upload' pour uploader notre vidéo que l'ou souhaite évaluer. Elle doit se trouver dans le champs 'video' des **form-data**. Et la dernière route '/api/result/ :id' qui permet de récupérer les résultats de l'évaluation par le model sous un format **json**. L'id en question nous est retourné par la route d'upload quand la video a été enregistrée.

4.3 Container Docker Python

Il y a 2 containers python, un pour faire le pré-traitement, et un pour évaluer les données après le pré-traitement grâce au model. J'ai envisagé de faire le pré-traitement dans un container matlab, mais j'ai été confronté a des problèmes d'OS. En effet, pour construire une telle image, il faut utilisé un OS Linux, j'ai donc du faire ce pré-traitement grâce au module Open-CV de python.

Au début, l'évaluation traitait un vidéo d'un seul coup. Mais j'ai eu des limitations de mémoire RAM. Pour éviter ce problème de ressource, j'ai du modifier légrèrement le fonctionnement.

5 Bilan

Voici un bilan de ce que j'ai pu faire par rapport à ce que je voulais faire :

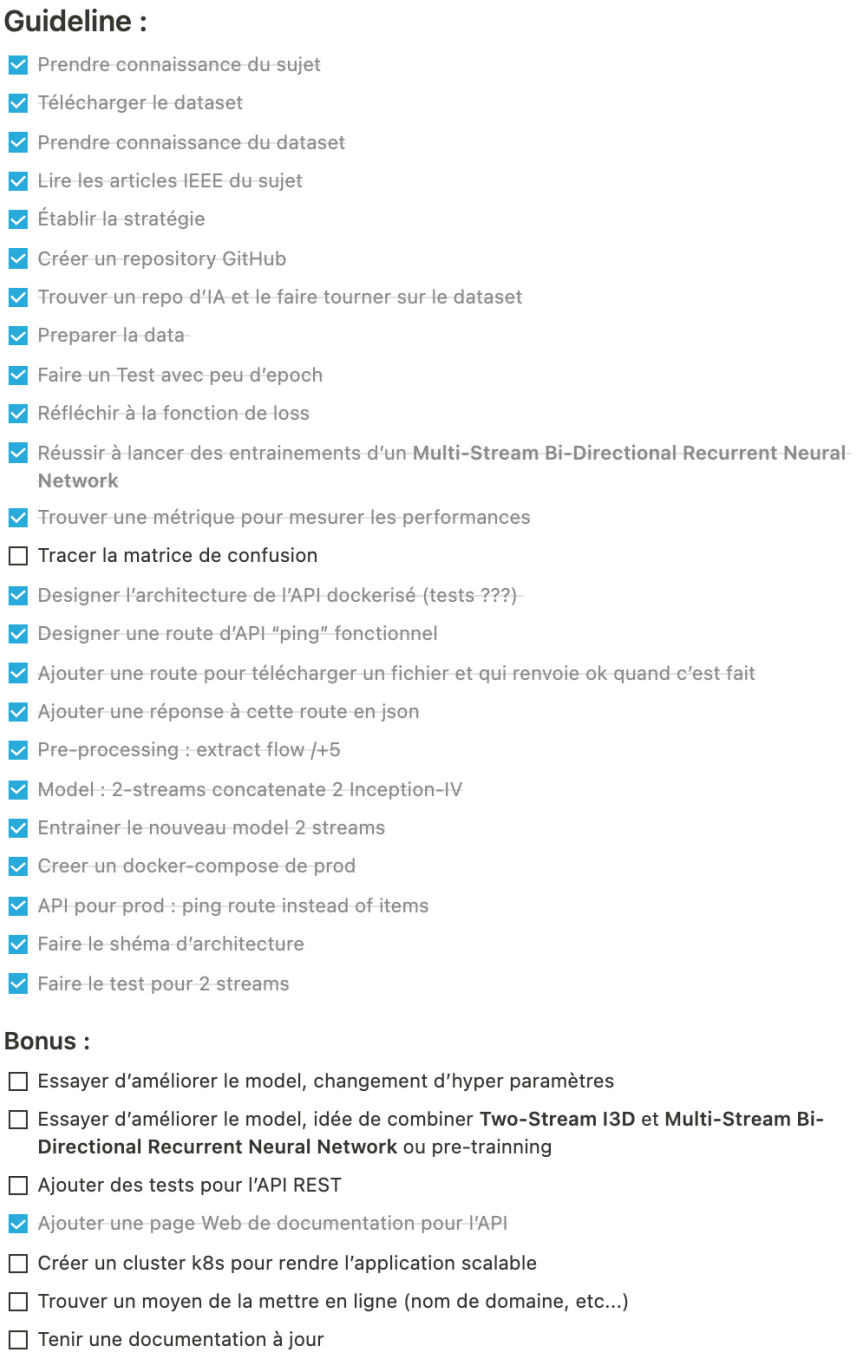

FIGURE 4 – Guideline

6 Conclusion

Pour conclure, j'ai réussi à mener à bien les différentes parties, pour arriver à remplir les objectifs. La durée du projet ainsi que l'incapacité Hardware d'effectuer un apprentissage profond, explique que les performances n'ont pas été priotisé.

Références

[1] Quo Vadis, Action Recognition ? A New Model and the Kinetics Dataset, Joao Carreira & Andrew Zisserman, 2018 : https ://arxiv.org/abs/1705.07750

[2] SlowFast Networks for Video Recognition, Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik et Kaiming He, Facebook AI Research (FAIR), 2019 : https ://arxiv.org/abs/1705.07750

[3] A Multi-Stream Bi-Directional Recurrent Neural Network for Fine-Grained Action Detection, Singh, Bharat ; Marks, Tim K. ; Jones, Michael J. ; Tuzel, C. Oncel ; Shao, Ming, 2016 : https ://www.merl.com/publications/docs/TR2016-080.pdf

[4] A Multi-Stream Bi-Directional Recurrent Neural Network for Fine-Grained Action Detection, Singh, Bharat ; Marks, Tim K. ; Jones, Michael J. ; Tuzel, C. Oncel ; Shao, Ming, 2016 : https ://www.merl.com/publications/docs/TR2016-080.pdf

[5] MERL shopping Dataset : https ://www.merl.com/pub/tmarks/MERL_Shopping_Dataset/

[6] Repo GitHub, model InceptionV4, 2018 : https ://github.com/naviocean/pytorch-inception :