Projet FitSize : Etude de l'IA

Florian CHAPPAZ, Valentin DE OLIVEIRA, Sami IFAKIREN, Clément NGUYEN

I. Introduction

Après le premier sprint, notre serveur est capable d'utiliser un script basé sur Pytorch et Ailia pour déterminer les keypoints d'un vêtement à partir d'une image. Ce script nous a été fourni par l'entreprise FitSize, et il est tiré d'un concours basé sur le dataset *DeepFashion2*. L'entreprise nous a donc demandé d'étudier les technologies utilisées, en particulier les points d'améliorations éventuels, afin de pouvoir rédiger une documentation pour les futurs développeurs reprenant notre travail. Ce travail sera également utile pour le cas où l'on souhaite ajouter de nouveau types de vêtements à détecter.

II. Technologies utilisées

Le framework de Machine Learning utilisé dans ce projet est PyTorch. Dans la documentation que nous a laissé le développeur du script, Mr. Muhammad Nouman Ahsan, il nous est également indiqué l'usage d'autres technologies que nous allons tâcher d'expliquer dans cette partie.

A. Ailia

Ailia est une solution cross-plateforme permettant d'effectuer des inférences à partir de modèles entrainés de manière performante (cette solution permet d'ailleurs l'utilisation de ressources GPU).
La solution Ailia SDK est utilisée pour faire des inférences sur des nouvelles entrées de vêtement.
Les informations sur Ailia SDK peuvent être retrouvées ici :
https://axinc.jp/en/solutions/ailia_sdk.html.
Les modèles supportés par Ailia sont décrits ici : https://github.com/axinc-ai/ailia-models.
En particulier, le modèle *FashionAI-key-points-detection* utilisé dans notre projet pour les 5 types de vêtements est disponible ici :
https://github.com/axinc-ai/ailia-models/tree/master/deep_fashion/fashionai-key-points-detection

Après consultation de ce repository, nous remarqué que notre script est fortement inspiré de celui donné en exemple :
https://github.com/axinc-ai/ailia-models/blob/master/deep_fashion/fashionai-key-points-detection/fashionai-key-points-detection.py

Nous notons aussi que nous n'avons pas forcément besoin de stocker les modèles sur notre serveur, et que nous pouvons les télécharger directement depuis l'url fournie.

B. YoloV7

YoloV7 est un modèle de détection d'objets en temps réel. Dans notre application, nous avons plusieurs modèles basés sur YoloV7, mais chacun étant entraîné pour un type de vêtement en particulier.
Lien du projet d'origine YoloV7 : https://github.com/WongKinYiu/yolov7.

C. OpenCV

OpenCV est un bibliothèque d'outils spécialisés dans la Computer Vision. Ce projet utilise notamment OpenCV pour calculer la correspondant pixel/cm à partir de l'échiquier 5x5cm placé sur l'image du vêtement. Néanmoins, OpenCV ne détecte pas automatiquement l'échiquier : c'est le détecteur YoloV7 qui s'en occupe.

D. DeepFashion2

DeepFashion2 est un dataset ouvert pour la détection de vêtement. C'est sur ce dataset que les modèles du projet ont été entrainés.
Lien du papier scientifique : https://arxiv.org/pdf/1901.07973.pdf.
Lien vers le projet : https://github.com/switchablenorms/DeepFashion2.

E. Autre référence

Une autre référence a été laissée par l'auteur de notre script. C'est une implémentation d'un papier de recherche sur les réseaux de neurones en cascades. Nous n'avons pas plus d'informations.
https://github.com/gathierry/FashionAI-KeyPointsDetectionOfApparel

III. Amélioration des performances

Dans le but d'améliorer les performances de calcul des keypoints, nous avons décider d'analyser les performances du script *run_no_det.py*, le script principal sur lequel nous nous appuyons.

L'ensemble des tests de ce document seront effectués sur une puce Apple Silicon M1.

Des points de débogage sont déjà présents, affichant sur la sortie standard une série de chiffres nous indiquant le fil de l'exécution dans le code.
Nous décidons d'ajouter à ces points de débogages des informations sur le temps en secondes depuis le passage au point précédent.

On récupère ensuite la sortie standard dans un fichier nommé *trash* (fichier ainsi nommé car nous n'avions au départ pas besoin de l'ensemble de la sortie standard).
Voici les informations obtenues après une exécution, et après suppression des lignes dont l'exécution était de l'ordre de la milliseconde :

```
DBG6000::: exec: 0.22164225578308105 sec
DBG13000::: exec: 0.15275001525878906 sec
DBG17000::: exec: 0.016477108001708984 sec
DBG18000::: exec: 0.1780698299407959 sec
DBG220000::: exec: 0.003947019577026367 sec
DBG230000::: exec: 1.6712150573730469 sec
DBG240000::: exec: 1.0062079429626465 sec
 The image with the result is saved in: runs/detections334/tmp690c11f2-beca-43e2-bd03-889487fd3800.jpg
Done. (2.892s)
```

Figure 1 : Extraits intéressants de la sortie standard après exécution du script run_no_det.py

Nous constatons dans notre script 2 étapes qui sont particulièrement coûteuses en temps :

- Entre le point DBG22000 et DBG23000
- Entre le point DBG23000 et DBG24000

Regardons maintenant la partie du code correspondant dans le script.

```
# inference feedforward
print("DBG220000:::" + exec_time())
output = net.predict({'img': imgl})
print("DBG230000:::" + exec_time())
output_flip = net.predict({'img': imgl_flip})
print("DBG240000:::" + exec_time())
```

Figure 2 : Partie du script coûteuse en temps

Les deux étapes les plus coûteuses en temps sont donc les étapes d'inférence. Ces deux lignes font appel à la fonction *predict()* du SDK Ailia.

```
net = ailia.Net(model_path, weight_path, env_id=args.env_id)
```

A. Améliorations GPU

Une des premières pistes lorsque l'on cherche à améliorer les performances d'une intelligence artificielle est des passer les calculs sur les ressources GPU.
Nous avons tout d'abord vérifier dans notre script si Ailia était exécuté sur un environnement GPU ou non :

```
ailia.get_gpu_environment_id()
```

A notre grande surprise, la fonction renvoie un chiffre différent de -1, ce qui signifie qu'Ailia tourne sur notre environnement GPU. Nous n'avons pourtant pas modifié le script, et notre environnement est assez spécifique : une puce à architecture ARM avec GPU intégré. C'est là toute la force de Ailia, qui nous promet un environnement rapide cross-plateforme.
La promesse est tenue, et nous n'avons donc pas à basculer l'exécution sur GPU manuellement.

IV. Conclusion

Par manque de temps, la conclusion de ce rapport arrive de manière un peu précoce. Néanmoins, le travail qui a été effectué ici permet de mieux connaître la stack technologique utilisée. Nous savons désormais que la piste d'amélioration principale se situe sur l'entraînement des modèles. Nous savons aussi que les modèles fournis sont issus d'un repository GitHub existant. Si il peut être difficile d'améliorer nous même les modèles, nous pouvons cependant surveiller les évolutions de ce projet Open Source. Nous pouvons nous documenter un peu plus sur ces modèles et même entrer en contact avec le créateur afin d'envisager un travail commun, en particulier pour ajouter de nouveaux types de vêtements.