TP-Intégrer une IA dans une application

Contexte :

Une entreprise Alpha (nom imaginaire) qui recycle des déchets plastiques pour en faire des filament d'impression 3D, loue des machines à d'autres entreprises qui permettent de trier leurs déchets. Par exemple les déchets plastiques produits par la pause déjeuner de leurs employés, bouteilles, couverts, gobelets, etc.

Dans la production de filament d'impression 3D, il est important de trier les déchets plastiques avant de les recycler car le point de fusion de chaque plastique est différent.

Alpha a conçu cette machine avec une interface où l'utilisateur sélectionne le type de déchet qu'il entre dans la machine. Malheureusement elle s'est rendue compte que les utilisateurs faisaient souvent des erreurs, entraînant des mélanges de plastiques. Ce qui réduit la qualité du filament.

L'entreprise souhaite donc rajouter une solution à base d'IA qui permettrait de résoudre le problème: En disposant un appareil photo dans le réceptacle de la machine, on pourrait prendre une photo du déchet qui permettra à un modèle de classification d'image de catégoriser automatiquement le déchet pour ensuite proposer le résultat à l'utilisateur qui pourra corriger si besoin.
Un classifieur d'images est un service d'IA qui applique des étiquettes ou labels (représentant des classes) à des images sur la base de leurs caractéristiques visuelles.

Azure propose dans son service cognitive services un module appelé Custom Vision.
https://docs.microsoft.com/fr-fr/azure/cognitive-services/what-are-cognitive-services

Custom Vision utilise un algorithme de Machine Learning pour appliquer des étiquettes à des images. On lui envoie des groupes d'images présentant les caractéristiques en question, pour chaque classe. On étiquette les images au moment de la soumission. L'algorithme s'entraîne ensuite avec ces données et calcule sa propre précision en se testant lui-même à l'aide de ces mêmes images. Une fois l'algorithme entraîné, on peut le tester, le réentraîner et éventuellement l'utiliser pour classer les nouvelles images en fonction des besoins de notre application. On peut également exporter le modèle en question pour l'utiliser en mode hors connexion.

TP:

1) Collecter au minimum 5 images par classe de déchets sur Google images par exemple:
   classe 1 = gobelet
   classe 2 = bouteille
   classe 3 = couvert

exemples d'images:




2) Créer un classifieur avec le site web Custom Vision (suivez la documentation suivante) : https://docs.microsoft.com/fr-fr/azure/cognitive-services/custom-vision-service/getting-started-build-a-classifier

3) Entraîner l'outil pour faire la classification en suivant la documentation.

4) Publier votre modèle entraîné, et récupérer l'url et la clé de prédiction pour pouvoir l'intégrer à l'application : https://docs.microsoft.com/fr-fr/azure/cognitive-services/custom-vision-service/use-prediction-api

L'application :

Dans le dossier vous trouverez un fichier triof.tar.gz, qui contient le code de l'interface utilisateur de la machine. Vous n'aurez pas à coder l'application dans ce TP, tout est déjà fait pour vous.

L'application est développée grâce au framework Flask de Python (pour plus de renseignement sur Flask, c'est par ici : https://flask.palletsprojects.com/en/1.1.x)

Vous trouverez dans le dossier triof:

- un fichier triof_app.py, c'est le fichier main qui vous permettra de lancer l'application.
- un dossier static qui contient les feuille de style, les images, …, qui font "l'esthétique" de l'application
- un dossier templates qui contient les fichiers html pour chaque page de l'application.
- un dossier caméra qui contient des images de déchets. En effet l'application que vous allez lancer est une simulation de la machine réelle, pour simuler la prise d'une photo une fonction choisi au hasard une photo dans ce dossier et la soumet à la prédiction.

- et enfin celui qui nous intéresse pour intégrer notre modèle de prédiction, le dossier src et à l'interieur le fichier utils.py qui contient toutes les fonctions utiles au fonctionnement de l'interface.

1) Récupérez ce fichier et dézippez le.

2) Ouvrez le fichier utils.py et copiez votre url et votre clé à l'endroit adéquat.

3) Lancer l'application.

   Les prérequis pour que l'appli tourne correctement:
   - Python 3.6, 3.7 ou 3.8
   - scipy 1.4.1
   - numpy 1.16.2
   - Flask 1.0.3

   Pour lancer une application flask, ouvrez un terminal, naviguez jusqu'à votre dossier triof, et exécuter triof_app.py. Le prompt vous renvoi le port local sur lequel l'app tourne cpiez le dans votre navigateur et testez.