LET'S SHOP!

L'objectif de ce projet est :

- ➢ Entraîner un modèle de reconnaissance d'actions à partir du jeu de données **MERL shopping Dataset.**
- ➢ Déployer le modèle entraîné afin de recevoir une vidéo et de la classifier.

Ce qui est important :

- La qualité de la réflexion : L'objectif n'est pas d'avoir le modèle le plus performant mais de pointer plusieurs problématiques liées à la classification des actions et d'apporter sa vision critique.
  Pour la partie (III- Déploiement), pensez performances et scalabilité.
- La capacité à franchir les obstacles qui se présenteront.
- Utilisez le langage **Python.**
- Documentez votre travail.
- Toute question ou remarque sont les bienvenues.

Ce qui est moins important:

- Pour la réalisation de ce projet, vous pouvez utiliser les outils, les librairies, bibliothèques et frameworks de votre choix.
- Pendant l'entretien, vous pouvez servir votre application localement ou fournir une url publique.

I- MERL shopping Dataset :

Le jeu de donnée suivant est à télécharger : *'https://www.merl.com/pub/tmarks/MERL_Shopping_Dataset/'*

Le jeu de données MERL contient 106 vidéos, chacune d'entre elles étant une séquence d'environ 2 minutes chacune d'entre elles étant une séquence d'environ 2 minutes.
Chaque vidéo contient plusieurs instances des 5 actions suivantes :

1. **Reach to shelf**
2. **Retract from shelf**
3. **Hand in shelf**
4. **Inspect product**
5. **Inspect Shelf**

Pour la réalisation de ces données, *41 personnes* y ont participé. *SubjectID = 1,….,41*
Chaque personne a participé à 3 vidéos au maximum. *SessionID = 1,…,3*

Chacune des 106 vidéos a comme nom de fichier : *'xx_yy_crop.mp4'* avec *xx le SubjectID* et yy *la SessionID*.

Les étiquettes pour chaque vidéo 'xx_yy_crop.mp4' sont des fichiers présentés sous la forme de **matrice Matllab 'xx_yy_label.mat'**
La matrice Matlab est appelée **'tlabs'**.

- **'tlabs{1}'** est un tableau de taille Kx2 qui liste la frame du début et la frame de la fin des K segments où l'action appartient à la classe 1.
- **'tlabs{2}'** est un tableau de taille Kx2 qui liste la frame du début et la frame de la fin des K segments où l'action appartient à la classe 2.
- **'tlabs{3}'** est un tableau de taille Kx2 qui liste la frame du début et la frame de la fin des K segments où l'action appartient à la classe 3.
- **'tlabs{4}'** est un tableau de taille Kx2 qui liste la frame du début et la frame de la fin des K segments où l'action appartient à la classe 4.
- **'tlabs{5}'** est un tableau de taille Kx2 qui liste la frame du début et la frame de la fin des K segments où l'action appartient à la classe 5.

II- Classification d'action : Modèle d'apprentissage profond :

Une fois le jeu de données inspecté. **L'objectif est de proposer un modèle permettant de classifier ces différentes actions.**
Une métrique permettant de juger les performances du modèle est à aussi à proposer.

****

Hint : 'Dans la littérature, plusieurs architectures de modèles permettent de répondre à la problématique de classification de vidéos.

'https://arxiv.org/abs/1705.07750' : Ce modèle I3D se base sur une architecture à deux entrées pour répondre à cette problématique.
La première entrée est la donnée RGB, soit les frames de la vidéo.
La deuxième entrée est le flot optique, il s'agit du mouvement apparent des objets, surfaces et contours d'une scène visuelle.
La combinaison de ces deux entrées et leur passage par plusieurs blocs de convolution 3D permet d'effectuer la classification d'une vidéo.

*'https://arxiv.org/abs/1812.03982' : Les auteurs de ce papier de recherche sont partis de l'observation suivante : les images des scènes vidéo contiennent généralement deux parties distinctes : des zones statiques dans l'image qui ne changent pas du tout ou changent lentement, et des zones dynamiques qui indiquent quelque chose d'important qui est en train de se passer. Par exemple, la vidéo d'un avion qui décolle comprend un aéroport relativement statique et un objet dynamique (l'avion) qui se déplace rapidement dans la scène. Dans un scénario de tous les jours où deux personnes se rencontrent, la poignée de main est généralement rapide et dynamique alors que le reste de la scène est statique. Ainsi, **SlowFast** utilise un CNN lent et haute définition (voie rapide) pour analyser le contenu statique d'une vidéo tout en faisant tourner en parallèle un CNN rapide et basse définition (voie lente) dont le but est d'analyser le contenu dynamique d'une vidéo. Cette technique est partiellement inspirée du ganglion rétinien des primates, dans lequel 80% des cellules (cellules P) fonctionnent à basse fréquence temporelle et reconnaissent les détails fins, et ~20% des cellules (cellules M) fonctionnent à haute fréquence temporelle et sont sensibles aux changements rapides. De même, dans SlowFast le coût de calcul de la voie lente est 4x plus grand que celui de la voie rapide.*

'https://www.merl.com/publications/docs/TR2016-080.pdf' : Ce modèle se base sur une architecture à quatre flux en entrée. Deux de ces flux traitent les

informations de mouvement (similaires au flux optique), un flux pour la région centrée sur la personne (recadrée) et un pour l'ensemble de l'image. Les deux autres flux traitent les informations sur l'image, un pour la région centrée sur la personne et un pour l'ensemble de l'image. Chaque flux est traité par un réseau modifié qui produit un vecteur caractéristique représentant le flux. Les quatre vecteurs de caractéristiques sont concaténés ensemble et projetés dans un vecteur de caractéristiques de dimension inférieure.

Le vecteur de caractéristiques de dimension inférieure produit par le réseau est ensuite introduit dans un réseau bidirectionnel de mémoire à long terme (LSTM). La classification est faite en sortie de la cellule (LSTM).

****

Une fois le réseau de neurones **proposé et entraîné sur la dataset MERL shopping**. Nous pouvons juger ses performances en analysant la matrice de confusion obtenue sur un jeu de données de test.

D'autres métriques peuvent être pertinentes pour conclure quant à la pertinence du modèle.

III- Déploiement

L'objectif de cette partie est de déployer notre modèle entraîné dans un environnement de production.
Pour cela il faudra mettre en place une **API REST** avec l'outil de votre choix qui délivrera au client **la classe prédite** par le modèle ainsi que **la probabilité de cette classification** à partir d'une vidéo reçue par un fichier **json** ou par **un chemin** vers la donnée sur la machine.