


Atelier d'Intelligence Artificielle (I-ISIA-202)

Système IA embarqué pour maisons et villes intelligentes

Edge AI System for Smart Cities and Smart Homes


***** Pistes & environnement de développement sur la ressource Edge « Jetson Xavier » *****

1. Entraînement de modèles sur ressources Cloud :

- Nous vous proposons d'entraîner vos modèles IA sur ressources Cloud « Colab, Colab Pro ou Pro+, vast.ai) à partir de vos PCs personnels que vous pouvez apporter avec vous durant le Workshop.

2. Environnement de travail sur la carte Jetson Xavier :

- Comme convenu, chaque groupe aura accès à une carte Jetson Xavier installée et configurée avec l'environnement « **deep_tf27_torch111** » utilisant les librairies présentes dans « **agx-environment.txt** » (Tensrflow 2.7 et torch 1.10)
- Pour activer cet environnement, il suffit de lancer le terminal et taper cette commande :

source deep_tf27_torch111/bin/activate

3. Projet de démarrage :

- Un programme de démarrage (Fig. 1) sera fourni dans le dossier « **home/HackIA22_Input** » qu'il faudra compléter avec vos modèles, algorithmes et optimisations. Vous pouvez le lancer après avoir activé l'environnement avec la commande : **python3 EdgeAI_Smart.py**
- Vous pouvez aussi partir de vos propres solutions si vous souhaitez.

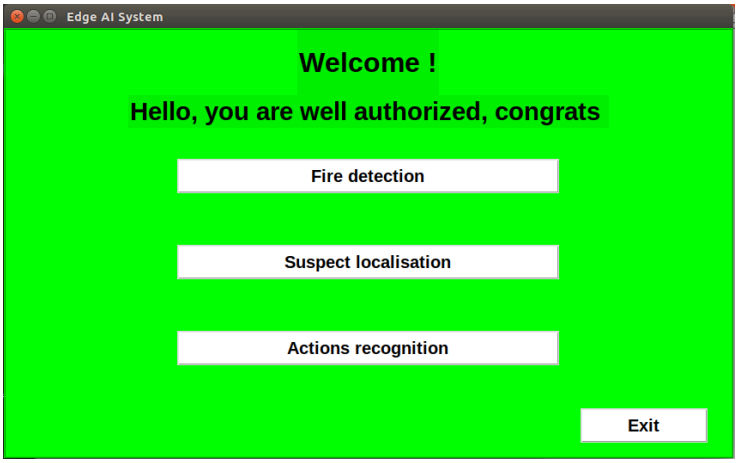

Figure 1: programme et interface de démarrage (obtenue avec authentification faciale)

4. Modèles de classification :

- **Modèle :** Vous pouvez entraîner vos modèles avec une version récente de tensorflow (2.7 ou 2.8) pour générer des modèles compatibles avec la version tf installée sur la carte « Jetson Xavier ».
- **Données de test :** seront partagées durant le Workshop.

5. Modèles de détection d'objets :

- En raison de sa précision et rapidité, nous vous suggérons d'utiliser Yolo V5 pour les modèles de détection d'objets en exploitant vos solutions utilisées pour la localisation de clés durant le défi 1.
- **Modèle :** lien Github Yolo V5 : **https://github.com/ultralytics/yolov5**
- **Données d'entraînement :** en plus des données partagées via le protocole, vous pouvez annoter des données réelles (si vous souhaitez) à l'aide de « **Roboflow** » très efficace. Voici un exemple de son utilisation pour Yolo V5 : **https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/**
- **Données de test :** seront partagées durant le Workshop.

6. Modèles de reconnaissance d'actions :

- **Modèle :** en raison de son efficacité, nous vous suggérons d'utiliser **mmaction2**
- **Données :** en raison de l'intensité de calcul de ce modèle, nous vous suggérons de prendre en compte 4 ou 5 classes d'actions uniquement (actions en lien avec le thème choisi « Smart Cities »)
- **Données de test :** à proposer par vos soins en fonction des classes d'actions choisies.

7. Explicabilité de modèles :

a. **Explicabilité de modèles Tensorflow : https://github.com/sicara/tf-explain**
b. **Explicabilité de modèles PyTorch : https://github.com/jacobgil/pytorch-grad-cam**

8. Eventuelles pistes de compression et optimisation de modèles :

En plus des méthodes d'optimisation proposées dans l'énoncé, nous partageons avec vous des pointeurs vers des frameworks NVIDIA d'optimisation de modèles : **NVIDIA TensorRT et NVIDIA TAO**

Tutoriel TAO : https://docs.nvidia.com/tao/taotoolkit/text/deepstream_tao_integration.html

Attention, ces frameworks **ne sont pas installés sur la carte (ceci reste optionnel bien évidemment)**.

9. Quelques conseils pratiques :

- **Note 1 :** pour l'interface de la figure 1, vous avez le libre choix d'adapter, modifier et incrémenter l'interface en fonction de choix de modèles et options d'améliorations.
- **Note 2 :** pour la reconnaissance faciale, il y a aussi une version PyTorch que vous pouvez utiliser si vous souhaitez : **https://github.com/timesler/facenet-pytorch**