UMONS

Certificat d'université en

Intelligence Artificielle

07 et 08 Mars 2020

WORKSHOP RÉSIDENTIEL CERTIFICAT IA UMONS

Hôtel Utopia

Système Edge IA pour maisons et villes intelligentes
Edge AI System for smart homes and smart cities

Ingénieurs | Informaticiens | Doctorants

Coachs Techniques :

Pr. Sidi Ahmed Mahmoudi, UMONS

Ir. Mohammed. A. Belarbi, UMONS

Ir. Olivier Debauche, UMONS

Coachs Business :

Ir. Benoit Vermoortel, Accenture

Ir. Nathan Derave, Accenture

Membres de Jury :

Pr. Pierre Manneback, UMONS

Pr. Thierry Dutoit, UMONS

Ir. Terry Wyckle, Infrabel

Ir. Tarik Zedazi, Accenture

Ir. Vivian Delplace, NECKO Technologies

Nombre de participants : 25
Nombre de groupes : 05

Hôtel Utopia
Chaussée Brunehault 392,
7050 Masnuy-Sait-Jean

065 84 87 85

UMONS Université de Mons | POLYTECH MONS | Faculté des Sciences | numediart INSTITUT DE RECHERCHE POUR LES TECHNOLOGIES CRÉATIVES DE L'UMONS | inforTech UMONS RESEARCH INSTITUTE FOR INFORMATION TECHNOLOGY AND COMPUTER SCIENCE | accenture

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

Système IA embarqué pour maisons et villes intelligentes

Edge AI System for Smart Cities and Smart Homes

Table des matières


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

Système IA embarqué pour maisons et villes intelligentes

Edge AI System for Smart Cities and Smart Homes


1. Enoncé

L'objectif de ce workshop est de développer un système d'intelligence artificielle embarqué sur des ressources Edge (matériel proche des capteurs de collecte de données). Le système s'appuiera sur les techniques et modèles Deep Learning vues et développés durant les défis du certificat IA. Ces modèles seront combinés pour fournir un module « Edge AI » appliqué aux vidéos capturées en temps réel. Chaque groupe aura le choix entre deux possibilités : module **« Edge AI »** pour maisons intelligentes ou un module **« Edge AI »** pour villes intelligentes.

1.1. Développement d'un module « Edge AI » pour maisons intelligentes

Développer un support pour maisons intelligentes avec une application utilisant des techniques et modèles Deep Learning pour détecter et localiser différents objets à partir d'images capturées via la caméra. Ce module devra utiliser deux ou trois modèles :

1.1.1. Modèle « Face »

Le premier modèle servira comme système d'authentification faciale grâce à la reconnaissance de visages. Ce modèle permettra d'autoriser l'utilisateur à employer les autres applications (modèles IA pour maisons intelligente) si la personne est à la fois reconnue et autorisée (via son visage).

1.1.2. Modèle « Fire »

Le deuxième modèle (de classification) consiste à détecter les feux ou fumées (ex. détecter des cas d'oubli de feu de gazinière allumée sans raison, présence de fumée, etc.) à partir d'images capturées via la caméra. Vous pouvez utiliser vos modèles développés lors du défi 1 du certificat IA.

1.1.3. Modèle « Personal »

Le troisième modèle (de localisation) consiste à localiser les objets personnels (clés, trousseaux de clés, télécommande, smartphone, portefeuille, etc.). Là aussi, vous pouvez repartir du modèle de localisation développé durant le défi 1 avant de l'augmenter pour détecter et localiser plusieurs objets, un réentrainement est nécessaire avec plus de classes.

Chaque groupe peut augmenter cette solution par d'autres applications et modèles mais cela reste optionnel (comptage d'objets, suivi d'objets, envoi d'alertes et notifications, etc.).

1.2. Développement d'un module « Edge AI » pour villes intelligentes

Développer un support pour villes intelligentes avec une application utilisant des techniques et modèles Deep Learning pour détecter et localiser différents objets ou situations à partir d'images capturées via la caméra. Ce module devra aussi utiliser deux ou trois modèles :

1.2.1. Modèle « Face »

Le premier modèle servira comme système d'authentification faciale grâce à la reconnaissance de visages. Ce modèle permettra d'autoriser l'utilisateur à employer les autres applications (modèles IA pour villes intelligente) si la personne est à la fois reconnue et autorisée (via son visage).

1.2.2. Modèle « Fire »

Le deuxième modèle (de classification) consiste à détecter les feux ou fumées dans des forêts proches de la ville intelligente à partir d'images capturées via la caméra. Vous pouvez utiliser vos modèles développés lors du défi 1 du certificat IA.

1.2.3. Modèle « Suspect »

Ce modèle (de localisation ou classification) consiste à détecter la présence d'un ou plusieurs objets suspects ou non autorisés (ex. bâtons, sacs isolés, armes, etc.) dans une scène filmée par la caméra.

1.2.4. Facultatif : modèle « Mouvement »

Le troisième modèle consiste reconnaitre et classifier les mouvements (marche, marche rapide, chute, dispute, etc.) dans une scène filmée par la caméra. Pour réaliser ce modèle, il est conseillé d'utiliser des bases de données publiques présentant des séquences vidéo annotées avec différents types de mouvements.

Chaque groupe peut augmenter cette solution par d'autres applications et modèles mais cela reste optionnel.

2. Phases du Workshop

Afin de répondre à l'énoncé décrit ci-dessus et d'organiser un partage de tâches entre les membres de chaque groupe, nous vous proposons de suivre les étapes suivantes :

- **Partie 0 :** choix du projet
- **Partie 1 :** développement et entrainement des modèles de classification et de localisation
- **Partie 2 :** Développement du programme de test (inférence) à partir de séquences vidéo, solution embarquée sur la ressource Edge : Jetson Xavier
- **Partie 3 (facultative) :** Portage de solution sur la ressource Edge « low cost » : Jetson Nano.

Chaque groupe peut intégrer d'autres idées innovantes pour améliorer la méthode proposée ci-dessus à tout en gardant l'objectif principal du projet choisi.

2.1. Partie 0 : choix du projet

Chaque groupe devra donner son choix de projet parmi les deux possibilités :

- Module **« Edge AI »** pour maisons intelligentes
- Module **« Edge AI »** pour villes intelligentes

Choix à compléter ici pour le **Lundi 02/03/2020** après concertation dans chaque groupe.

2.2. Partie 1 : développement et entrainement des modèles

Le choix du projet vous permettra de sélectionner les modèles à développer. Les entrainements peuvent être réalisés avec les mêmes frameworks utilisés lors du défi 1 (Python, Tensorflow, Keras, OpenCV, etc.). En termes de ressources, vous pouvez utiliser Google Colab ou Floydhub. Chaque groupe disposera, si besoin, de 50 heures de calcul avec Floydhub. Les principaux modèles à développer et entrainer sont :

2.2.1. Modèle « Face »

Ce modèle permettra d'identifier et reconnaître les visages des personnes présentes dans la scène. Pour cela, nous vous recommandons d'utiliser l'architecture FaceNet permettant de détecter et reconnaître des personnes (Nom de la personne). Bien sûr, il faudra entrainer le modèle avec une base de visages personnalisée (exemple : visages des différents membres du groupe). Cette partie peut être réalisée en trois sous-étapes :

a. **Collecte et préparation de la base de visages :** pour simplifier la phase de collecte de données, nous vous proposons de travailler sur vos propres visages en sauvegardant une vidéo pour chaque membre du groupe. Ensuite, il faudra extraire les images de ces vidéos en utilisant le programme « video_to_image.py » disponible via ce lien. Pour lancer le programme, il faudra :

   - Se connecter à votre environnement de travail (PC personnel, Google Colab, Floydhub, etc.)
   - Via le terminal, aller sur le dossier FaceNet (cd FaceNet)
   - Lancer la commande suivante : **python video_to_image.py**

   Comme résultat vous devrez avoir cinq dossiers, dont chacun contient les images d'un membre du groupe. Il faudra copier ces sous-dossiers dans le dossier « PERSONS » récupérable en cliquant ici. Vous aurez donc 13 classes (8 classes représentant les membres du service IILIA et 5 classes représentant les membres de votre groupe). Veuillez garder la confidentialité de ces images.

b. **Extraction et localisation des visages :** cette étape consiste à extraire les visages à partir des images collectées plus haut (Fig. 1), une étape primordiale pour la reconnaissance des visages. L'extraction se fera via un algorithme de détection, fourni dans le projet facent. Pour lancer ce programme, il faudra lancer la commande (l'ensemble des étapes et données décrites ci-dessous sont mises en place dans ce notebook):

```
python src/align/align_dataset_mtcnn.py  PERSONS PERSONS_ALIGNED
```

- align_dataset_mtcnn.py : représente le programme de détection de visages
- PERSONS : représente le dossier contenant les photos de personnes (voir plus haut)
- PERSONS_ALIGNED : représente le résultat (visages détectés), nous vous conseillons de vérifier les images résultantes étant donné que cet algorithme n'est pas efficace à 100%. Ce dossier présente donc notre base d'apprentissage pour la reconnaissance de visages.

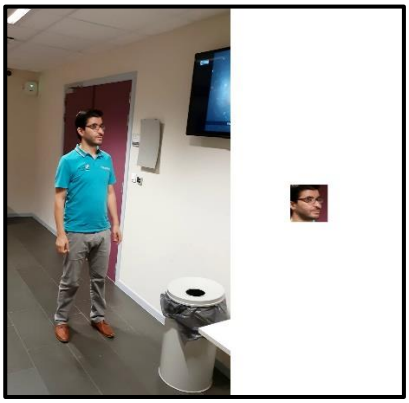

FIGURE 1 : EXEMPLE DE DÉTECTION ET LOCALISATION DE VISAGE

c. **Reconnaissance des visages :** une fois que la base de visages est bien préparée, nous pouvons utiliser l'algorithme de reconnaissance de visages en utilisant le modèle FaceNet, pré-entraîné par Google sur une base de 16 millions de visages. Vous êtes donc invités à utiliser la technique de « transfer learning » pour entraîner ce modèle avec votre base de visages. Pour lancer cet apprentissage, il faudra lancer la commande :

```
python src/classifier.py TRAIN FPMS_PERSONS_ALIGNED/ 20170511-185253.pb
model_checkpoints/my_classifier.pkl --batch_size 100
```

Le résultat de cet apprentissage est représenté par le modèle « **my_classifier.pkl** »

Si vous voulez lancer l'entrainement à nouveau, il faut lancer la commande suivante :

```
python src/train_softmax.py --logs_base_dir logs/facenet --models_base_dir . --data_dir
FPMS_PERSONS_ALIGNED/ --model_def models.inception_resnet_v1 --optimizer ADAM
```

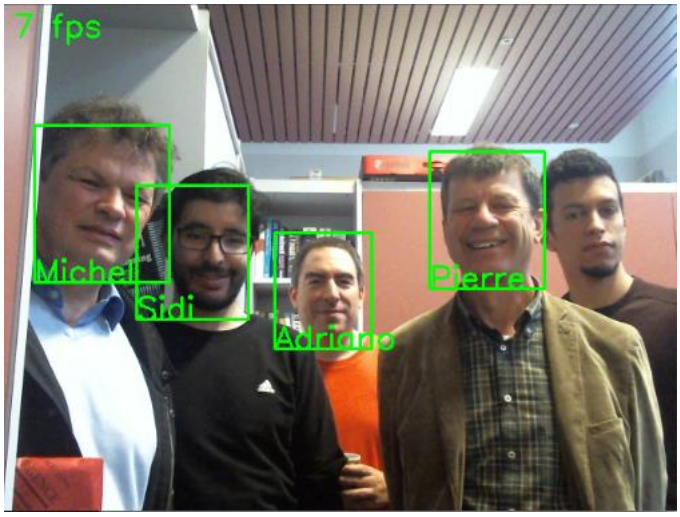

FIGURE 2 : EXEMPLE DE RECONNAISSANCE DE VISAGES

d. **Reconnaissance de visages en temps réel :** il faudra tester le modèle développé ci-dessus « **My_classifier.pkl** » avec les images provenant, de la webcam ou d'une vidéo enregistrée. Vous pouvez utiliser le programme « real_time_face_recognition.py » fourni avec le dossier « facenet/contributed ».

2.2.2. Modèle « Fire »

Il s'agit du modèle de classification développé lors du défi 1, vous pouvez utiliser vos modèles du défi 1.

2.2.3. Modèle « Personal »

Ce modèle à consiste à détecter et localiser les objets personnels (clés, télécommande, smartphone, portefeuille, etc.). Vous pouvez repartir du modèle de localisation développé durant le défi 1 en ajoutant :

- augmenter la base de données de clés (utilisée lors du défi1) par d'autres classes d'objets annotées (télécommande, smartphone, portefeuille, etc.). N'hésitez pas à utiliser des bases de données publiques ou des modèles déjà pré-entrainés.
- ré-entrainer le modèle de localisation avec la nouvelle base de données

2.2.4. Modèle « Suspect »

De la même manière avec laquelle vous avez travaillé lors du défi 1, vous pouvez développer un modèle (de classification ou localisation) permettant de détecter la présence d'un ou plusieurs objets suspects dans une image. Il faudra donc utiliser la technique du « transfer learning » pour exploiter un modèle, pré-entrainé avec la base ImageNet, et l'adapter par rapport à votre base de données, le but étant d'identifier la présence d'un objet suspect. Pour démarrer cette étape, nous vous fournissons un code de classification (similaire au défi1) ainsi qu'une base de données d'objets suspects via sur ce lien. Bien évidemment, vous êtes invités à modifier/augmenter la base de données et modèles en fonction de vos choix d'objets à détecter.

2.2.5. Modèle « Mouvement » (Facultatif)

Ce modèle a pour objectif de reconnaitre des mouvements (exp. marche, marche rapide, fuite, chute, dispute, etc.) à partir de séquences vidéo. Etant donnée qu'une vidéo est toujours représentée par une succession d'images, les modèles de classification d'images pré-entrainés pourront être utilisés pour reconnaitre les mouvements. Toutefois, ces modèles ne prennent pas en compte l'information temporelle ce qui entraine des pertes de précisions. Dans ce contexte, il existe d'autres types de réseaux de neurones profonds (réseaux de neurones récurrents, réseaux de neurones convolutionnels 3D, etc.) qui peuvent combiner les informations spatiales, temporelles (ainsi que les caractéristiques de mouvements) pour générer des modèles capables d'identifier les mouvements avec une précision satisfaisante. Pour démarrer cette étape, nous vous invitons à consulter les bases de vidéos publiques (Fig. 3) ainsi que des projets de classifications de vidéos partagées en ligne.

a. Bases de vidéos annotées :

- **UCF-101 :** 13320 clips, 27 heures de vidéo, 101 classes
- **Kinetics :** 306425 clips, 400 à 700 classes
- **HMDB51 :** 6766 clips et 51 classes
- **Etc.**

BD	Année	Nb. actions	Nb. clips/ action	Clips	Vidéos
HMDB51	2011	51	min. 102	6766	3312
UCF101	2012	101	min. 101	13320	2500
Thumos ('15)	2013	101	/	/	> 20700
Sports-1M	2014	487	1000-3000	1000000	1000000
ActivityNet	2015	200	env. 141	28108	19994
Youtube-8M	2016	4800	/	8264650	8264650
Kinetics	2017	400	min. 400	306245	306245
HACS	2018	200	/	1550000	504000

FIGURE 3 : COMPARAISON ENTRE LES BASES DE VIDÉOS

b. Projets de classification de vidéos :

- Approche Two-Stream
- Approche LRCN « Long-term Recurrent Convolutional Network »
- Approche Temporal 3D ConvNets
- Etc.

2.3. Partie 2 : portage de solution sur la ressource Edge : Jetson Xavier

Après validation des modèles, il faudra les porter vers la ressource Edge Nividia Jetson Xavier afin de développer une solution embraquée et appliquée aux vidéos capturées via la caméra connectée à la carte (via un port USB). L'idée serait de combiner les modèles pour fournir un module Edge AI pour maisons ou villes intelligentes en fonction de votre choix. Les figures 4 et 5 illustrent les programmes à développer et compléter (Edge AI systems for Smart Homes and Smart Cities).

```
#        ******************         EDGA AI module For Smart Homes         ******************         #
# face : model de reconnaissance facile
# fire : model de détéction de feu, fumée, etc.
# personal : modèle de détection ou localisation d'objets personnels
# movement : modèle de reconnaissance de mouvements/actions (facultatif)
cap = cv2.VideoCapture(0)    # Capture de la vidéo en temps réel
    while(cap.isOpened()):
        frame = cap.read(); # lecture de chaque image de la vidéo capturée
        faces = face_recognition.identify(frame)    # Modèle de reconnaissance faciale (à lancer pendant 10 secondes par exemple)
        if (face.name in liste)                     # liste : membres authorisés à utiliser le module
        FirePred1 = fire.predict(x)[0] ;
        PersonalPred1 = personal.predict(x)[0] ;
        .....................................       # Facultatif : autres modèles aux choix
        }
cap.release()
cv2.destroyAllWindows()
```

FIGURE 4: "EDGE AI" ALGORITHM FOR SMART HOMES

```
#        ******************         EDGA AI module For Smart Cities         ******************         #
# face : model de reconnaissance facile
# fire : model de détéction de feu, fumée, etc.
# suspect : modèle de détection ou localisation d'objets suspects
# movement : modèle de reconnaissance de mouvements/actions (facultatif)
cap = cv2.VideoCapture(0)    # Capture de la vidéo en temps réel
    while(cap.isOpened()):
        frame = cap.read(); # lecture de chaque image de la vidéo capturée
        faces = face_recognition.identify(frame)    # Modèle de reconnaissance faciale (à lancer pendant 10 secondes par exemple)
        if (face.name in liste)                     # liste : membres authorisés à utiliser le module
        FirePred1 = fire.predict(x)[0] ;
        SuspectPred1 = suspect.predict(x)[0] ;
        MvtPred1 = movement.predict(x)[0] ;         # Facultatif
        }
cap.release()
cv2.destroyAllWindows()
```

FIGURE 5: "EDGE AI" ALGORITHM FOR SMART CITIES

FIGURE 6 : SYSTÈME IA UTILISANT LA RESSOURCE EDGE "JETSON XAVIER"

Vous avez le libre choix de proposer une interface graphique ou pas et avec l'outil de votre choix (tkinter, PyQt, etc.). Notons que chaque groupe disposera d'une carte Jetson Xavier préconfigurée avec les différentes librairies Python, Tensorflow et Keras : « requirement.txt ». Les modalités d'accès aux environnements préconfigurés seront fournies durant le Workshop.

2.4. Partie 3 (facultative) : portage sur la ressource Edge « low cost » : Jetson Nano

Pour ceux qui veulent aller encore plus loin pour ce Workshop, nous vous proposons de porter votre solution de la partie 2 sur une ressource Edge « Nvidia Jetson Nano ». Cette carte à faible coût (moins de 100 euros) dispose de moins de ressources de calcul et de stockage mémoire. Il faudra donc adapter votre solution en travaillant sur les points suivants :

- Optimiser et accélérer l'algorithme en termes du temps de calcul et d'inférence de modèles ;
- Utiliser des réseaux de neurones de petites tailles vu l'espace mémoire réduit de la carte Jetson Nano ;
- Compresser les modèles développés dans la partie 1 en sélectionnant les poids de modèles les plus représentatifs, vous pouvez utiliser des solutions de la littérature :
    a. A Survey of Model Compression and Acceleration for Deep Neural Networks
    b. Jetson Nano: Deep Learning Inference Benchmarks
    c. Neural Network Compression
    d. Etc.

Là aussi, chaque groupe disposera d'une carte Jetson Nono préconfigurée avec les différentes librairies Python, Tensorflow et Keras. Les modalités d'accès aux environnements préconfigurés seront fournies durant le Workshop.

3. Conclusion

Nous tenons à rappeler que l'objectif de ce Workshop est de mettre en œuvre des modèles de classification et de localisation d'objets avant de les embarquer dans un module « Edge AI » pour maisons ou villes ou intelligentes. Nous vous invitons à commencer par le développement d'un module utilisant un nombre minimal de modèles (2) avant de le porter vers la carte Jetson Xavier en offrant un traitement temps réel à partir d'images capturées via la webcam. A l'issue de cela, vous pourrez intégrer les autres tâches optionnelles choisies (intégration d'autres modèles, envoie d'alertes et notifications, portage vers la carte « Jetson Nano, etc.) ou proposées par vos soins.