Système embarqué de vidéo surveillance utilisant l'apprentissage profond (Deep Learning)

• Introduction

Le challenge consiste à développer un système de vidéosurveillance exploitant les techniques du Deep Learning vues durant les défis du certificat IA. Le système devra contenir au moins deux modèles de Deep Learning. Le premier permettra de détecter la présence d'un ou plusieurs objets suspects dans une scène filmée par la caméra. Le deuxième modèle consistera à identifier et reconnaître les visages présents dans la scène (en cas de détection d'objets suspects via le premier modèle). Une fois que vous avez un résultat satisfaisant pour les deux premières étapes, nous vous proposons de porter votre application sur un matériel embarqué composé de : une carte Movidius[1], un microcontrôleur Raspberry[2] et une caméra qui permettra de filmer la scène. Pour réaliser ce travail, nous vous proposons de suivre le protocole suivant, incluant quatre parties : connexion à la machine à distance, développement du modèle de classification (détection), développement du modèle de reconnaissance de visages, développement d'un système embarqué.

1. Partie 1 : connexion à la machine distante :

Afin d'avoir des temps d'apprentissage et de test rapides, nous offrons pour chaque groupe deux (02) accès à des machines distantes (équipées de GPU GTX 1080). Notre choix s'est porté sur Shadow[3], une plateforme facilement configurable et connectable avec les caméras. Afin d'exploiter au mieux cette machine, nous vous proposons de réaliser les étapes suivantes :

1.1. Téléchargement de l'application d'accès : pour télécharger l'application, il faudra aller sur le site de la plateforme Shadow et se connecter avec les logins et mots de passe fournis. Une fois connecté, il suffit d'aller sur l'onglet « Applications », télécharger et exécuter l'application d'accès en fonction de votre système d'exploitation (Fig. 1). Pour ceux qui utilisent Linux, ils peuvent télécharger l'application en version BETA (voir en bas de la page). Après avoir installé l'application Shadow, vous pouvez la lancer (sans oublier d'activer l'option USB à distance) pour vous connecter en vue d'avoir accès à une machine équipée d'un GPU NVIDIA GTX 1080, 12 Go de RAM, 256 Go d'espace de stockage et un processeur à 8 threads (Fig. 2 et Fig. 3).

[1] Movidius. https://www.movidius.com/
[2] Rasbperry PI. https://www.raspberrypi.org/products/
[3] Shadow. https://shadow.tech/int

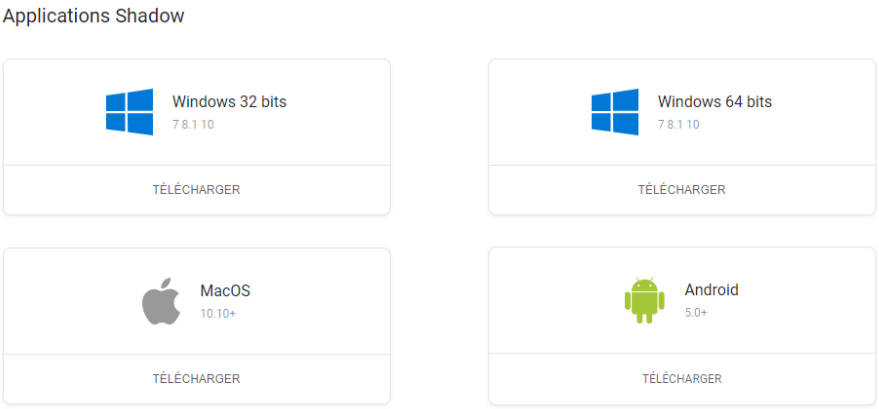

Figure 1: Applications Shadow

Figure 2: connexion avec l'application Shadow

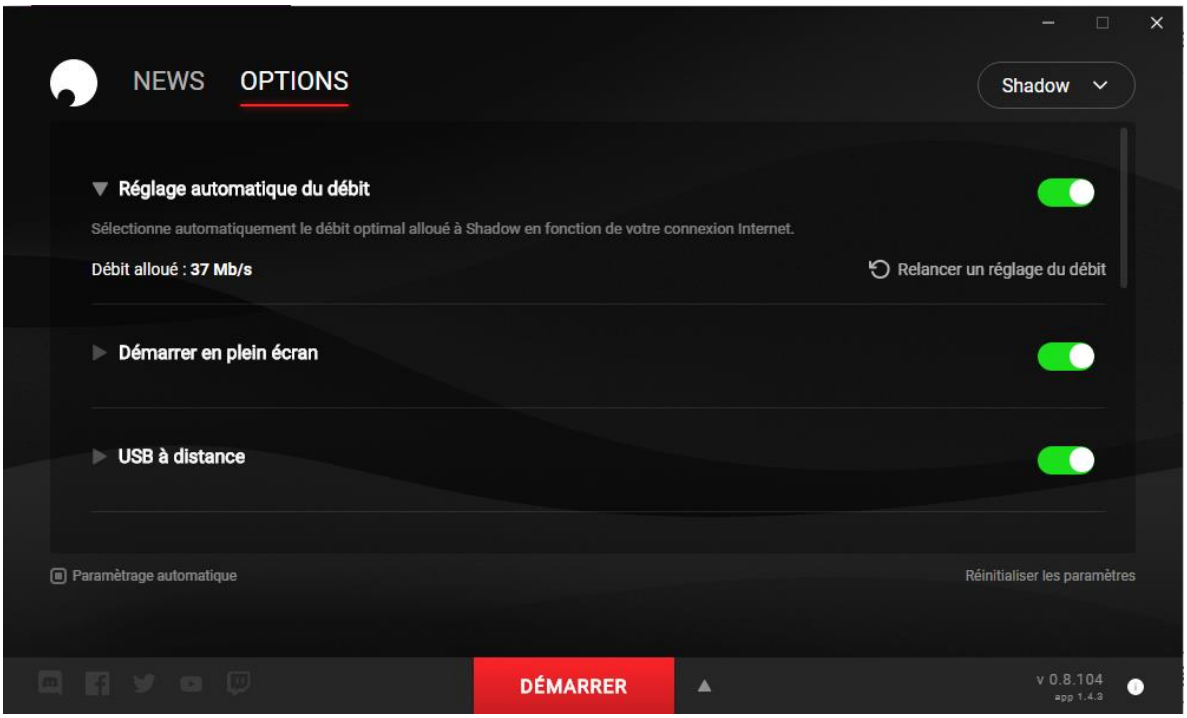

Figure 3: Option de connexion à la machine Shadow

Note : pour quitter le mode plein écran de la machine Shadow, il suffit de cliquer sur « WIN + Ctrl + F »

1.2. Lancement de l'environnement Anaconda : pour vous faciliter le travail, vos machines Shadow sont déjà configurées avec un environnement Anaconda utilisant les librairies OpenCV, Keras et Tensorflow. Pour avoir accès à cet environnement, il suffit de :

- Lancer l'environnement Anaconda Prompt
- Sur le terminal, aller vers le dossier que vous souhaitez utiliser pour le projet
- Activer l'environnement (via le terminal) avec la commande : conda activate challenge
- Lancer l'environnement Jupyter avec la commande : jupyter notebook (Fig. 4)
- Créer votre programme en cliquant sur « New ---> Pyhton » pour avoir un environnement semblable à celui de Google Colab (utilisé lors des défis)

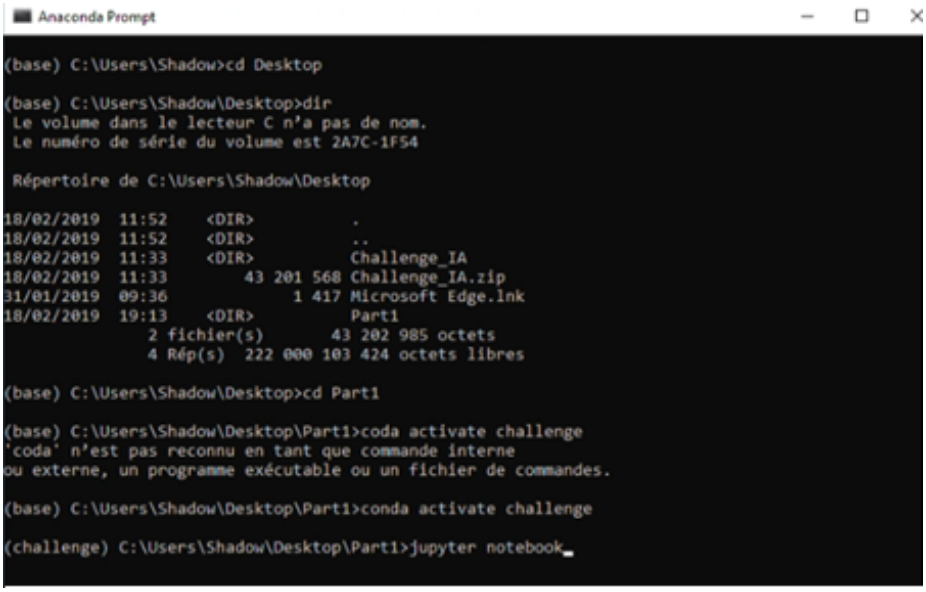

Figure 4: lancement de l'environnement anaconda

2. Partie 2 : développement du premier modèle de détection en temps réel : de la même manière avec laquelle vous avez travaillé lors du Défi 1, vous pouvez développer un modèle de classification permettant de détecter la présence ou pas d'un objet suspect dans une image. Il faudra donc utiliser la technique du « fine-tuning » 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. Notons que la classe personne ne présente pas un objet suspect dans notre cas. Pour démarrer cette étape, nous vous fournissons un code de démarrage téléchargeable en cliquant sur ce lien (dossier contenant d'autres fichiers à utiliser ultérieurement).

Après avoir défini et entraîné le meilleur modèle avec votre base d'images, il faudra créer un programme de test du modèle à partir de la vidéo capturée via la webcam de votre PC (ou une caméra USB). Pour cela, nous vous conseillons de réaliser les tâches suivantes :

2.1. Vérifier l'accès à la webcam : la webcam de votre PC (ou une caméra USB) peut être accessible dans la machine Shadow si vous activer l'option « USB à distance » lors du lancement de la machine. Pour vérifier si votre caméra est bien accessible, il faudra aller dans l'onglet « Périphériques USB » et cocher votre caméra. Pour la tester, il suffit de lancer l'application « Caméra » permettant de visualiser la scène filmée depuis votre PC sur la machine Shadow.

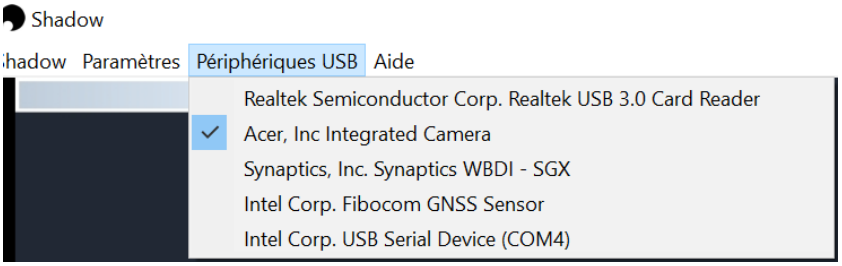

Figure 5: activation d'accès à la webcam

2.2. Développer un programme de capture de flux vidéo : pour cette tâche, nous pouvons créer un code Python permettant de capturer la vidéo provenant, soit de la webcam ou d'une vidéo déjà enregistrée, avec la fonction OpenCV VideoCapture. Chaque image de la vidéo pourra être visualisée à l'aide d'une fenêtre OpenCV. Dans le dossier « Part1 », téléchargé plus haut, vous trouverez un code de lecture et affichage de vidéo avec Pyhton « lecture_video_py ».

2.3. Tester la présence d'un ou plusieurs objets suspects pour chaque frame de la vidéo : pour chaque image de la vidéo, il faudra tester la présence d'un ou plusieurs objet suspect en utilisant la fonction model.predict() (model représente votre modèle de détection).

Figure 6: exemples de détection des classes : personne et manifestants

Note : nous laissons la liberté pour ceux qui préfèrent utiliser une architecture de localisation d'objets.

3. **Partie 3 : développement du modèle de reconnaissance de visages :** cette étape dépend du résultat de la première. En effet, si le premier modèle détecte un objet suspect, le deuxième modèle devra être utilisé pour 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 :

3.1. 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 : Part2. Attention, le lien présente d'autres fichiers qui seront utilisés plus tard. Pour lancer le programme, il faudra :

- Toujours via le terminal Anaconda, aller sur le dossier FaceNet (cd FaceNet)
- Lancer l'environnement Jupyter avec : jupyter notebook (S'il n'est pas déjà ouvert)
- 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 » récupéré plus haut. Vous aurez donc 13 classes (8 classes représentant les membres du personnel du service INFO et 5 classes représentant les membres de votre groupe). Veuillez garder la confidentialité de ces images.

3.2. Extraction et localisation des visages : cette étape consiste à extraire les visages à partir des images collectées plus haut (Fig. 7), une étape primordiale pour la reconnaissance des visages. L'extraction se fera via un algorithme de détection, qui est également fourni dans le dossier téléchargé plus haut. Pour lancer ce programme, il faudra :

- Lancer la commande : python src/align/align_dataset_mtcnn.py PERSONS PERSONS_ALIGNED

Où :

- 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.

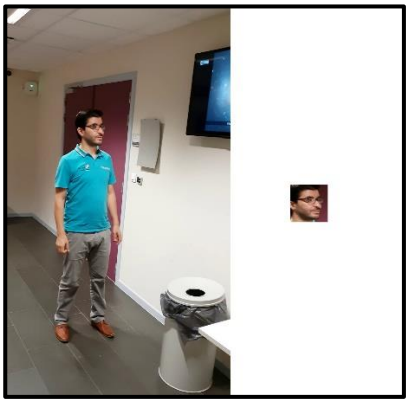

Figure 7: exemple de détection et localisation de visage

3.3. Reconnaissance des visages : une fois que la base de visages est bien préparée, nous pouvons développer 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 « fine-tunig » 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 »

Par ailleurs, 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
```

Ce script permet de générer 3 fichiers, qui pourront être utilisés dans la dernière partie (04) de ce workshop, qui consiste à développer une version embarquée.

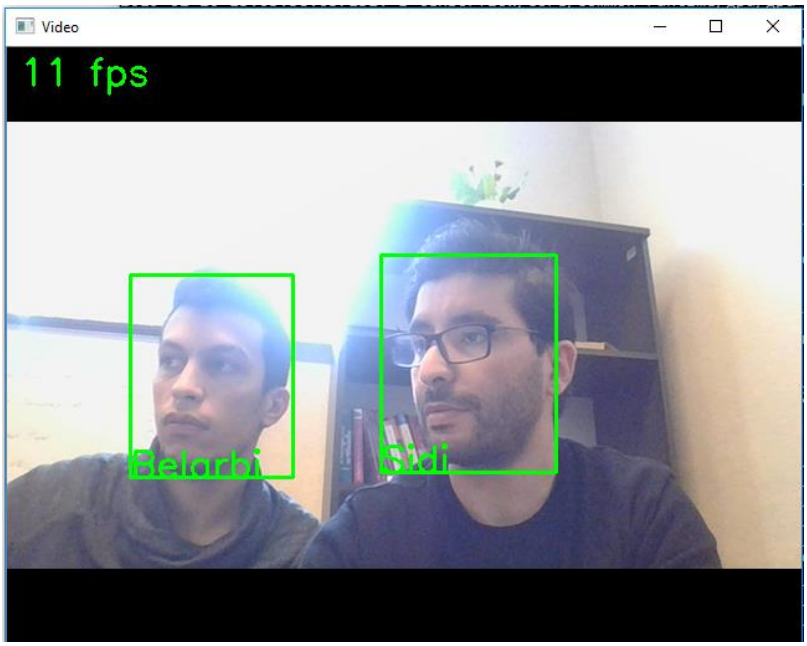

Figure 8 : exemple de reconnaissance de visages

3.4. Reconnaissance de visages à partir de vidéos en temps réel : ici, il faudra appliquer et tester le modèle développé ci-dessus « My_classifier.pkl » avec les images provenant, soit de la webcam ou d'une vidéo déjà enregistrée.

Pour finir la partie 3, vous êtes donc invités à combiner les deux modèles développés lors des deux dernières étapes (2 et 3) pour fournir une application de vidéo surveillance en temps réel, implémentant l'algorithme suivant :

```
# model1 : modèle de détection d'objets suspects
# face_recognition : modèle de reconnaissance de visages
cap = cv2.VideoCapture(0)   // Capture de la vidéo en temps réel
    while(cap.isOpened()):
        frame = cap.read();
        # Préparation de l'image en vue de la tester avec les deux modèles
        Pred = model1.predict(x)[0] ;
        # Vérifier le pourcentage de détection de chaque classe
        # Si (1 ou plusieurs classes présente un pourcentage > Seuil)  {
            faces = face_recognition.identify(frame)   // Lancement du 2ème modèle
            # Affichage des visages reconnus
            # Envoi d'une notification (visages, temps, durée, position, etc.)
            }
cap.release()
cv2.destroyAllWindows()
```

Pour résumer, l'objectif des trois premières étapes consiste à détecter la présence d'un ou plusieurs objets suspects dans la scène filmée par la caméra. En cas de détection positive, il faudra

- Détecter et identifier toutes les personnes présentes dans la scène
- Fournir et notifier le temps, la durée ainsi que les positions liées à toute détection

4. Partie 4 : développement d'un système embarqué :

Dans cette étape, nous vous proposons de développer une application de traitement de vidéo sur le matériel embarqué composé de : une carte Movidius[4], un microcontrôleur Raspberry[5] et une caméra qui permettra de filmer la scène. Le Raspberry Pi fourni est déjà pré-configuré pour qu'il fonctionne avec le Movidius. Si vous souhaitez connaître la procédure de configuration, nous vous orientons vers le site officiel de Intel : https://movidius.github.io/ncsdk/install.html

Etant donné que nous utilisons le Raspberry Pi qui est moins puissant qu'un PC, nous installons que les librairies nécessaires pour faire fonctionner le Movidius. Pour développer votre premier programme embarqué, nous vous proposons de suivre les étapes suivantes :

4.1. Réaliser le branchement Raspberry / Movidius / Webcam

Premièrement, il faut réaliser le branchement comme montré sur la figure suivante :

Figure 9 : Branchement Raspberry / Movidius Webcam

Le matériel vous sera transmis durant le premier jour du workshop. Le nom d'utilisateur du Raspberry ainsi que le mot de passe sont :

- User : pi
- Password : Raspberry

4.2. Tester le bon fonctionnement du Movidius :

- Ouvrir le terminal
- Allez sur le dossier workspace : **cd ~/workspace/ncsdk/examples/apps/hello_ncs_py**

[4] Movidius. https://www.movidius.com/
[5] Rasbperry PI. https://www.raspberrypi.org/products/

- Tapez la commande : **python hello_ncs.py**

Vous devez avoir le résultat suivant :

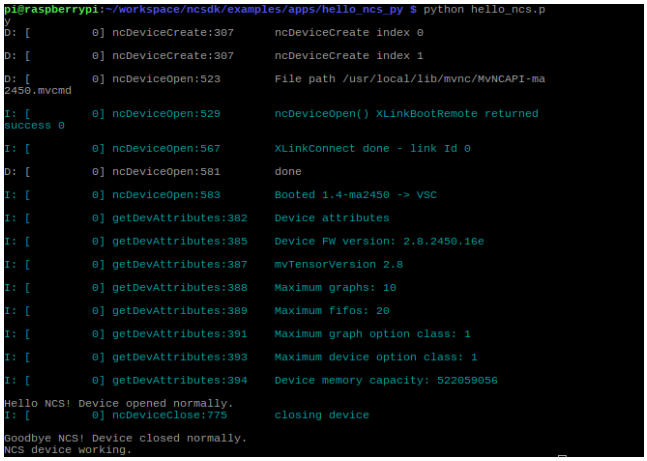

Figure 10 : Résultat du test de Movidius

4.3. Tester un exemple de classification :

Dans le SDK, Il existe un exemple de classification utilisant la carte Movidius. Pour le tester, il suffit de suivre les étapes suivantes :

- Ouvrir le terminal
- Aller sur le dossier workspace : **cd Desktop/workspace/ncappzoo/caffe/SqueezeNet**
- Lancer la commande : **python3 run.py**

Vous devez avoir un résultat semblable à celui présenté dans la figure ci-dessous (Fig. 11) :

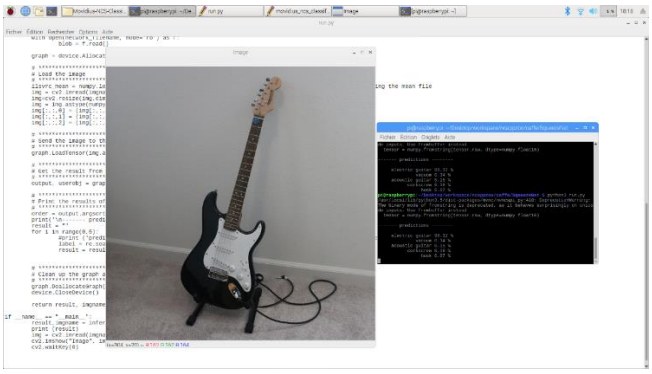

Figure 11 : Résultat de la classification avec matériel embarqué

4.4. Tester un exemple de localisation « Yolo » :

Dans le SDK, Il existe un aussi exemple de localisation utilisant la carte Movidius. Pour le tester, il suffit de suivre les étapes suivantes :

- Ouvrir le terminal

- Aller sur le dossier workspace : **cd Desktop/workspace/ncappzoo/caffe/TinyYolo**
- Tapez la commande : **python3 run.py**

Vous devez avoir un résultat semblable à celui présente dans la figure ci-dessous (Fig. 12) :

Figure 12 : Résultat d'utilisation de Yolo avec une image

Un autre code est mis à votre disposition afin d'utiliser la webcam avec l'algorithme Yolo. Pour le tester, il suffit suivre les étapes suivantes :

- Ouvrir le terminal
- Aller sur le dossier workspace : **cd Desktop/YOLO_WEBCAM**
- Taper la commande : **python3 yolo.py**

Note : il ne faudra pas oublier de brancher la caméra.

4.5. **Question 1 (Partie 4) :** après avoir fait les différents tests sur un matériel embarqué, nous vous demandons de développer un programme Python (utilisant la webcam) permettant de :

- Localiser les objets en utilisant Yolo
- Déterminer le sexe d'une personne (Male ou Female), en utilisant le modèle GenderNet
- Déterminer l'Age de la personne en utilisant AgeNet.

Note : les réseaux GenderNet, AgeNet sont disponibles via : **/Desktop/workspace/ncappzoo/caffe**

4.6. **Question 2 (Partie 4) :** portez les modèles développés durant les parties 2 et 3 sur un matériel embarqué utilisant : Processeur Rasbperry Pi, carte Movidius et une Webcam Pi. Il faudra donc écrire le programme Pyhton faisant appel à ces deux modèles.

Note : La carte Movidius exige un format spécifique pour les modèles (format graph). Il faudra donc d'abord générer les modèles sous ce format. Pour réaliser cette tâche, nous vous invitons à suivre les instructions indiquées sur ce lien : https://movidius.github.io/ncsdk/index.html