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

Système IA embarqué pour maisons intelligentes

Edge AI System for Smart Homes

Contents

1 Objectifs

L'objectif est de développer un système d'intelligence artificielle embarqué sur une ressource Edge AI (matériel proche des capteurs de collecte de données). Le système s'appuiera sur des modèles de Deep Learning pour la détection d'objets, similaires à ceux pratiqués lors des défis du certificat IA, mais aussi des modèles pour la détection de points ou l'extraction d'empreintes. Ces modèles seront combinés pour fournir un module « Edge AI » appliqué à des flux vidéos en temps réel au service des maisons intelligentes.

Avant de passer à l'énoncé, nous présentons quelques notions nécessaires pour la compréhension de la suite.

2 Module « Edge AI » pour maisons intelligentes

Le support pour maisons intelligentes utilisera des modèles Deep Learning pour détecter et localiser des personnes, des objets et des événements particuliers à partir d'images provenant de plusieurs caméras. Ce module devra au minimum comporter 2 fonctionnalités de base (« programme imposé ») ainsi qu'une fonctionnalité additionnelle au choix (« programme libre »).

2.1 Fonctionnalités de base

2.1.1 Fall Safe

La première fonctionnalité servira à détecter quand une personne chute. Cela peut être utile pour les personnes âgées vivant seules la journée par exemple. Nous verrons plus loin que plusieurs approches alternatives reposant sur des modèles d'IA de différents types (détection d'objet, en considérant qu'une personne debout, assise, ou tombée sont trois classes distinctes; ou détection des points du squelette en préalable d'une algorithmique particulière de prise de décision) peuvent être envisagées. Ces approches ont chacune leurs avantages et inconvénients. Il est sans doute possible de les combiner pour plus de robustesse, même si ce n'est pas une nécessité pour ce workshop.

2.1.2 Lost Item

La deuxième fonctionnalité consiste à pouvoir détecter des objets oubliés et alerter le propriétaire. Par exemple, si on oublie ses clés, une notification est envoyée pour signaler cet oubli et même diriger une caméra vers l'objet. Cela peut se faire avec un modèle de détection d'objets entrainé pour le(s) objet(s) souhaité(s).

2.2 Fonctionnalités additionnelles

2.2.1 Intrusion Detection

Cette fonctionnalité doit permettre de détecter la présence d'une personne qui n'a rien à faire là c'est-à-dire celles qui ne sont pas préalablement enregistrées. Cela peut être réalisée par identification faciale grâce à un modèle

de reconnaissance de visages. Nous donnerons une piste plus loin, car ces modèles peuvent reposer sur une approche un peu différente d'extraction d'empreinte faciale de type FaceNet.

2.2.2 Kids Safe

Cette fonctionnalité doit servir à déterminer lorsqu'on entre dans une zone dangereuse, ou au contraire lorsqu'on quite une zone de sécurité. Nous voulons ainsi pouvoir déterminer quand un bébé par exemple entre dans une zone qui lui est interdite (proximité d'un feu ouvert). Un modèle de détection d'objets ou de tracking du squelette serait approprié, mais il devra fonctionner sur de tout petits humains aussi.

2.2.3 Pets Fun

Cette fonctionnalité pourra détecter un animal de compagnie (chat ou chien) et le suivre dans ses mouvements, afin de pouvoir par la suite continuer à saturer l'internet avec des chats mignons ou amusants, mais aussi et surtout de détecter quand il occupe un espace qui lui est « interdit ». Pour cela, un modèle de détection d'objets est le plus approprié. La fonctionalité est très semblable à la précédente, si ce n'est le type d'objet à détection qui diffère.

Comme nous pouvons le constater, un grand nombre de ces fonctionnalités peuvent reposer sur le concept de détection d'objets, voire de détection de points d'intérêts comme les différentes positions des articulations du corps. Cependant, les types d'objets à détecter sont spécifiques à chaque application. Des modèles préexistants peuvent être disponibles sur internet, et vous pouvez aussi adapter vos modèles du Défi 1. Cependant, certaines applications nécessitent un fine-tuning et donc des données spécifiques, notamment pour la détection d'objets oubliés ou le fait qu'une personne soit tombée. Une étape de récolte des données et d'annotations sera donc nécessaire. Une fois les données récoltées et annotées, des prétraitements d'images ou des méthodes d'augmentation de données d'apprentissage peuvent être appliqués pour améliorer la qualité des détections et permettre aux modèles de mieux généraliser.

> **Note :** informez vous sur la problématique du "domain shift", ou demandez au coaches de vous expliquez en quoi il peut être cricial ici de s'assurer que les données de fine-tuning représentent au mieux le type d'image qui sera vu par le système, même du point de vue de l'angle de prise de vue par exemple.

Finalement, vous devrez optimiser vos modèles afin de maximiser le nombre de trames par seconde pour avoir une fluidité maximale pour les 3 flux vidéo en parallèle sur le hardware fourni.

N'hésitez pas à imaginer des noms de fonctionnalités plus attrayants que ceux proposés du point de vue du marketing et pour la présentation finale de votre solution.

Objectifs: chaque groupe devra développer/intégrer trois modèles au minimum : les deux fonctionnalités imposées et un troisième au choix. Si un groupe le souhaite, il peut étendre la solution par d'autres modèles. Il est préférable de travailler au raffinement des trois modèles et à l'intégration d'idées innovantes dans ceux-ci, afin d'améliorer la qualité de la solution tout en gardant l'objectif principal, plutôt que de se disperser sans entrer en profondeur dans les solutions. Notez aussi que les modèles doivent couvrir au moins deux types d'approches : détection d'objets, détection de points, extraction d'encodage. Ceci vous deviendra plus clair en lisant la suite.

3 Tâches

Afin de répondre à l'énoncé proposé et d'organiser un partage de tâches entre les membres de votre groupe, nous vous proposons de suivre les étapes suivantes, principalement en séquence, mais aussi avec un chevauchement :

- **Exigences :** choix des fonctionnalités que vous souhaitez réalisées et les approches pour y parvenir;
- **Tâche 0 : collecte et préparation des données** : collecte et sur site d'images d'apprentissage (les organisateurs déclinent cependant toutes responsabilités en cas de mauvaise chute ;-)) et leur annotation, ainsi que la mise en place de procédures d'augmentation de données d'apprentissage (par transformation d'images) ;
- **Tâche 1 : développement et entraînement des différents modèles** : il s'agit ici de mettre en place les modèles et d'effectuer les apprentissages nécessaires ;
- **Tâche 2 : développement du programme de test (inférence + notification)** : à partir de flux vidéo sur la ressource embarquée « Edge AI » Jetson AGX Xavier et intégration du dispositif multi-caméra et robotisé;
- **Tâche 3 : optimisation/compression :** d'au moins un modèle. Pour cette partie, il peut être intéressant de comparer et de combiner différentes approches de compression, telles que la quantification et la distillation, et de tester différents outils tels que TensortRT, ONNX, torchao, ou même directement via Yolo.

3.1 Tâche 0 : collecte et préparation des données

Allez-y. Photographiez et annotez un maximum de personnes tombées, et d'objets à retrouver. Pour les objets, nous prendrons 4 catégories: smartphone, lunettes, portefeuille et trousseau de clés. Pour les personnes, l'annotation peut aller jusqu'au points des articulations. Pour les objets, on se contentera de la boîte englobante.

Chaque photo peut comporter plusieurs objets car c'est le nombre d'instances total d'objets dans la base de données qui est un peu plus important que le nombre total d'images différentes.

3.2 Tâche 1 : choix, développement et entraînement des modèles

Les entraînements (et fine-tunings ou distillations) peuvent être réalisés sous Python avec le Framework Pytorch (pour le Deep Learning) et la librairie OpenCV (pour le traitement d'images). Nous vous permettons aussi d'utiliser la librairie Ultralytics offrant les modèles de détection YOLO (en favorisant le modèle récent Yolo11). Ces librairies sont déjà installées sur la carte Jetson Xavier qui est fournie. En termes de ressources pour ces apprentissages, vous pouvez utiliser des machines ou services sur le cloud comme DataCrunch, Google Colab Pro, vast.ai ou paperspace. Chaque groupe disposera, si besoin, de 24h de calcul sur RTX A6000 (Datacrunch ou Vast.ai) ou d'un abonnement (1 mois) avec Colab Pro ou paperspace. Nous avons également des solutions internes via les coachs techniques présents sur place grâce à des eGPUs et clusters UMONS. Différents types de modèles vont pouvoir vous être utiles pour réaliser les fonctionnalités. Il peut y en avoir 3 types :

1. **Modèles de détection d'objets:** Le modèle Yolo11 est conseillé. Différentes fonctionnalités peuvent en bénéficier :

   (a) « Fall Safe » car vous pouvez développer un détecteur qui comprend 3 classes : personne debout, personne assise, personne tombée par terre.

   (b) « Lost Item », où vous pouvez spécialiser un modèle à détecter des objets types que nous pouvons régulièrement oublier. Les catégories d'objets d'un modèle par défaut peuvent sans doute déjà servir de base mais ne seront pas suffisantes.

   (c) « Pet Fun » et « Kids Safe », les catégories d'objets d'un modèle par défaut peuvent sans doute déjà servir de base aussi.

   Il s'agira alors de trouver un tel modèle de base et de l'améliorer (apprentissage pour classifier de nouvelles catégories, ou sur données additionnelles), et ensuite de l'optimiser pour le Edge.

2. **Modèles de détection de points spécifiques:** Ce genre de détecteur, que vous n'avez peut-être pas encore pratiqué, permet de déterminer précisément sur une l'image la position de points d'intérêt spécifiques. On peut ainsi notamment identifier les positions des différentes articulations d'une personne et ainsi obtenir une information complète sur sa posture et ses gestes. Un tel modèle est par exemple entrainé sur le corpus COCO-Pose. Différentes fonctionnalités peuvent en bénéficier :

   (a) « Fall Safe » : vous pouvez développer un détecteur sur base des positions des différents points du squelette pour ensuite prendre une décision sur sa posture. Il s'agit donc d'une approche alternative à celle basée sur le détection d'objets.

(b) « Kids Safe » : il est probable qu'un challenge est que les détecteurs préentrainés existants soient biaisés et fonctionnent alors moins bien sur des petits enfants.

3. **Modèles d'extraction d'encodage d'un objet ou d'un visage:** Ces approches fonctionnent sur base d'un principe différent qui consiste à calculer une sorte d'empreinte visuelle (faciale par exemple). Cette empreinte peut alors être comparée avec celles calculées au préalable sur des images de référence, et déterminer si l'image présente ou non le même objet ou visage que ceux référencés. C'est une approche idéale pour la reconnaissance de visage et elle sera donc utile pour :

   (a) « Intrusion Detection » en permettant de pré-référencer des images des membres de la famille (de l'équipe pour ce week-end).

Lorsque vous établirez vos exigences fonctionnelles et les approches pour les implémenter, vous pouvez par exemple choisir : « Fall Safe » via détection d'objets, « Lost Item » via détection d'objets, « Intrusion Detection » via encodages de visages. Ou alors vous faites « Fall Safe » via détection de points, « Lost Item » via détection d'objets, « Kids Safe » via détections d'objets. Il est en tout cas demandé de ne pas baser toutes vos fonctionnalités uniquement sur de la détection d'objets.

Recommandations Nous listons dans cette partie quelques recommandations pour chaque méthodes qui pourraient vous être utiles.

- **Détection d'objets.** Nous conseillons ici le modèle Yolo11, ainsi que la pratique dès le début du workshop des outils permettant de réaliser des entrainement, fine-tunings, distillations et compression de ce modèle car la détection d'objet fera d'office partie de votre solution. Un tel modèle peut déterminer la localisation exacte d'objets à l'aide de rectangles englobants (Bounding Boxes). Si l'image présente plusieurs objets prévus, ils seront chacun entourés par un rectangle en fonction de leurs positions.
- **Détection de points spécifiques.** Nous conseillons également Yolo11 et particulièrement la version "pose" lorsque vous voulez travailler avec le squelette.
- **Reconnaissance de visage par encodage d'empreinte** Ce modèle permet d'identifier et reconnaître les visages des personnes présentes dans la scène. Pour cela, nous vous recommandons d'utiliser le programme Facenet permettant de détecter et reconnaître des personnes (Nom de la personne) à partir de l'introduction d'une seule image (visage) par personne. Vous êtes invités à bien analyser le code et l'appliquer correctement dans votre projet pour permettre la détection de personnes non autorisées si c'est votre objectif.

Réalisation des apprentissages pour la détection

1. **Préparation et annotation de données :** pour vos modèles de détection, il faudra réaliser des entraînements sur des données avec autant de variété que possible. Par exemple pour la détection de chute ou d'objets par cette méthode :

   - des images présentant des personnes dans différentes postures ainsi que d'autre objets ;
   - des cadres indiquant les positions et tailles exactes des objets/personnes/zones présents dans chaque image ;
   - une étiquette (« labels ») indiquant la classe pour chaque cadre créé ;

   Comme point de départ, nous vous fournissons des bases de données déjà annotées pour certains des problèmes visés. Il est demandé d'accroitre ces bases de données avec des images à annoter par vos soins, images que vous allez même capturer sur place pendant le workshop. Pour annoter vos images, vous pouvez utiliser différents outils d'annotations tels que : Roboflow, labelme , labelImg, etc. Si vous souhaitez annoter des images avec des points d'intérêt plutôt que des cadres d'objets, c'est également possible. Le processus est très similaire, avec une étiquette à fournir pour chaque point d'intérêt, sur base d'une liste de catégories également prédéfinie.

2. **Entraînement des modèles :** une fois que votre base de données est annotée, vous pouvez commencer l'entraînement de votre modèle. La technique de fine-tuning par Transfert Learning est fort conseillée en vue de réduire les temps de calcul et d'améliorer la précision des résultats.

3. **Test et évaluation du modèle :** une fois que votre entrainement est finalisé, vous pouvez tester le modèle résultant via le dispositif Edge, afin d'évaluer votre modèle en condition réelle.

> **Note :** des codes de démarrage ou des fonctions de base vous serons fournis.

3.3 Tâche 2 : intégration du dispositif multi-caméra robotisé

Après la conception des modèles, il faudra les porter vers la ressource Edge Nividia Jetson AGX Xavier afin de développer une solution embarquée et appliquée aux flux vidéo provenant des caméras connectées à la carte (ici via USB, bien que d'autres options soient aussi possible sur le Jetson). L'idée sera d'intégrer tous vos modèles pour fournir un module Edge AI pour maisons intelligentes en fonction de vos choix. Deux objectifs sont visés ici. Tout d'abord, il faudra valider que les modèles fonctionnent bien, et commencer à évaluer les capacités du hardware à traiter trois flux vidéo en temps réel, et chiffrer le frame rate atteignable pour ces modèles. C'est aussi à ce moment que vous prendrez en main le dispositif multi-caméras avec pour objectifs de :

1. pouvoir faire tourner les modèles sur les différents flux en même temps.
2. déterminer la position spatiale des objets/personnes avec pour objectifs,
   (a) de diriger la caméra robotisée pan-till dans la direction souhaitée (ex : personne tombée, ou objet à retrouver).
   (b) de déterminer si une personnes est hors d'une zone de sécurité ou s'approche d'une zone de danger.

Pour la localisation spatiale, il vous est proposé de travailler avec la stéréoscopie. En détectant le même point spécifique sur un objet observé par deux caméras distinctes, il est ensuite possible d'en retrouver la position (x,y,z). Le problème peut devenir complexe lorsqu'on s'intéresse à plusieurs points ou qu'il y a plusieurs objets/personnes car ceci implique de trouver les associations, c'est-à-dire à quel point de la caméra 1 correspond un point particulier de la caméra 2. Nous proposons de commencer par une solution qui est limitée un seul objet/personne. Par exemple, vous déterminez dans les deux images la position du nez de la personne, ou bien le centre de gravité de celle-ci, selon les informations qui sont visibles et détectées dans les images. Pour le pilotage de la caméra pan-tilt, cela devrait vous être assez simple car des méthodes python seront disponibles pour piloter la caméra pour qu'elle s'oriente dans une direction précise. Quelques simples calculs trigonométriques suffisent pour obtenir cette direction sur base du point à viser. Pour terminer, notez que la présence de plusieurs flux vidéo vous offre aussi la possibilité de rendre plus fiables vos détections, si vous avez le temps d'implémenter une telle solution. Par exemple, si au moins 2 des trois caméras détectent la chute, alors on pourra le confirmer. Dans le cas contraire, on pourrait en déduire qu'il s'agit d'une fausse alerte. Ceci est bien sûr à valider plus précisément.

Note : l'utilisation de caméras pour déterminer la position spatiale d'objets est un domaine de la photogrammétrie. Une des difficultés résulte du fait que les caméras ne sont pas idéales et leur optique en particulier crée des distorsions de l'image. Ces distorsions sont notamment très marquées quand on s'approche des bords ce qui peut rendre les estimations effectuées très imprécises. La solution à ce problème consiste à effectuer une calibration des caméras, ce que OpenCV fait très bien. En très bref, cela consiste à corriger les images pour que les lignes droites soient droites dans l'image résultante. Ce n'est pas une obligation, mais gardez cela en tête car si vous avez bien avancé sur le reste, vous pouriez vous réserver un peu de temps pour tenter de calibrer vos caméras avec un script qui vous sera fourni, et d'ajouter dans votre solution l'usage de cette calibration pour corriger les images fournies par les caméras, et même calibrer les positions relatives des 2 caméras. La précision de vos estimations de position n'en devrait être que bien meilleure.

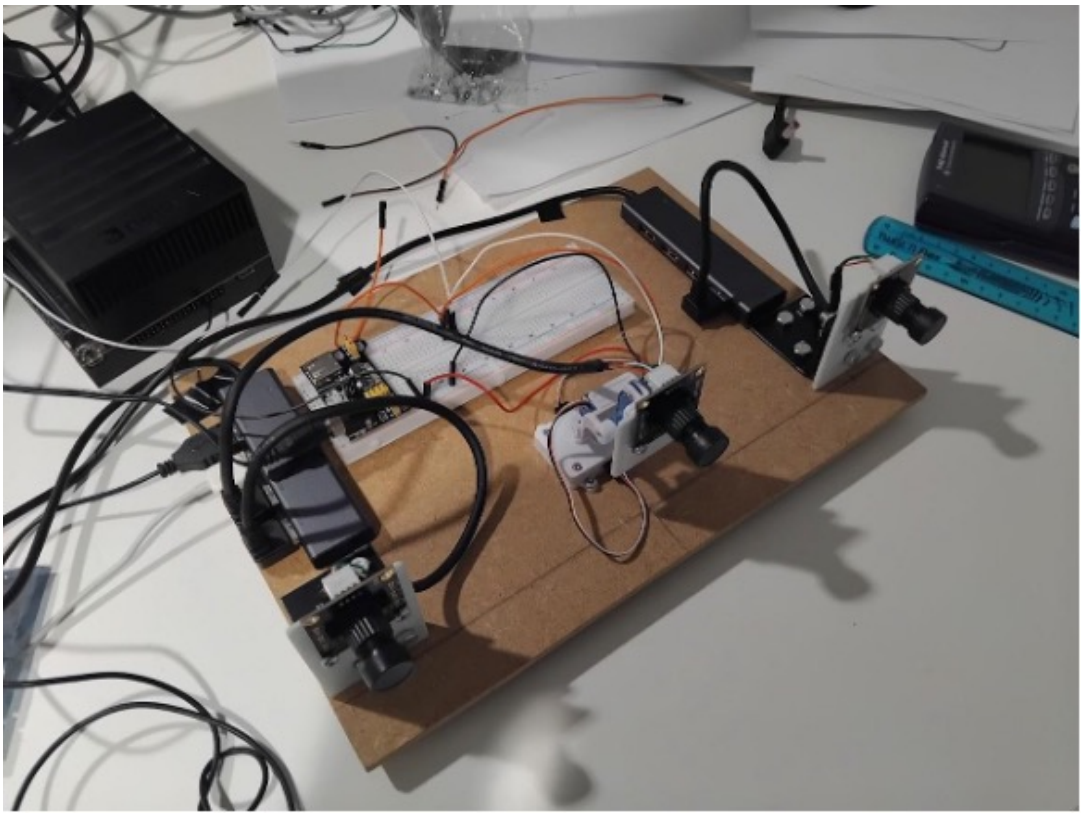

Figure 1: Illustration du dispositif multi-caméras

Recommandations

- **Interface graphique** Pour l'interface graphique, vous êtes libres de proposer une interface graphique à votre application avec l'outil de votre choix (tkinter, PyQt, etc.) ou d'utiliser celle fournie (en tkinter). Quel que soit le choix, votre application peut aussi toujours interagir via le périphérique clavier et dans un terminal. Cependant, elle doit au moins présenter les flux vidéo, et un maximum d'informations provenant des détections faites par les modèles en les affichant. Notons que chaque groupe disposera d'une carte préconfigurée avec les différentes librairies : PyTorch, OpenCV, Sickit-learn, Matplotlib, Ultralytics, etc. Les modalités d'accès aux environnements préconfigurés seront fournies durant le Workshop.
- **Envoi des notifications** Certaines des détections effectuées par votre solution sont destinées à déclencher l'envoi de notifications/alertes par exemple par email ou bot Telegram, ou encore un message WhatsApp voire même SMS si vous le souhaitez. Pour ce faire, vous pouvez utiliser le service Twilio qui propose un compte pour essai gratuit et une librairie twilio-python permettant de l'utiliser dans vos scripts, et c'est assez simple aussi pour Telegram. Vous êtes bien sûr libre d'intégrer toute solution alternative similaire.

3.4 Tâche 3 : optimisation/compression des modèles

Afin d'avoir une solution lpus rapide et moins gourmande en ressources de calcul, nous vous proposons de l'optimiser en travaillant sur les points suivants :

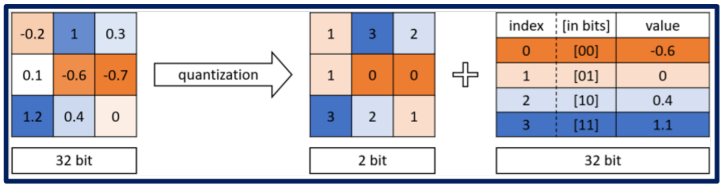

Figure 2: Illustration du principe de quantification

Analyse des performances : Il est important d'analyser les performances des modèles en termes de métriques indicatives de différentes qualités :

1. la qualité de modèle à réaliser sa tâche (précision)
2. le nombre de trames vidéo par seconde qu'il est capable de débiter (FPS – frames per second)
3. l'espace de stockage et l'espace mémoire nécessaires lors de son utilisation ;

Optimiser et compresser les modèles : à l'aide des techniques de quantification et de distillation de connaissances. Notez qu'une autre technique, celle de l'élagage, est aussi largement répandue. Nous ne l'utiliseront pas ici car il est plus difficile d'en tirer parti, à moins d'avoir des architectures de calcul plus spécifiques, comme un FPGA (réseau de portes programmables in situ) par exemple.

- **La quantification :** consiste à appliquer un processus d'approximation sur les réseaux de neurones afin de représenter les poids avec des nombres à plus faible précision (nombre de bits réduit) sachant que les poids sont initialement représentés par des nombres en virgule flottante et en 32-bis (format float). Ce processus permet de réduire considérablement la taille de stockage, mémoire et le temps de calcul des réseaux de neurones profonds. La Figure 2 illustre un exemple de quantification de poids d'un réseau de neurones. Le nombre de bits doit être choisi en veillant à ce que la précision reste maintenue proche de celle du modème initial. En effet, une réduction trop importante de la précision des valeurs de poids risquera d'affecter négativement la précision du modèle.
- **Distillation de connaissances « Knowledge distillation » :** consiste à développer un petit réseau de neurones « Student » qui pourra apprendre durant par entrainement à partir d'un grand réseau de neurones « Teacher ». L'objectif est d'avoir un réseau de neurones simplifié mais qui prendra des décisions semblables aux décisions d'un réseau de neurones grand et complexe. Cette approche est illustrée à la Figure 3. En effet, si l'on fait confiance au gros modèle, on peut aussi considérer que celui-ci peut fournir des annotations sur lesquelles le petit modèle peut apprendre. Cela permet de bénéficier des données spécifiques, même non étiquetées lors de cette forme particulière d'apprentissage.

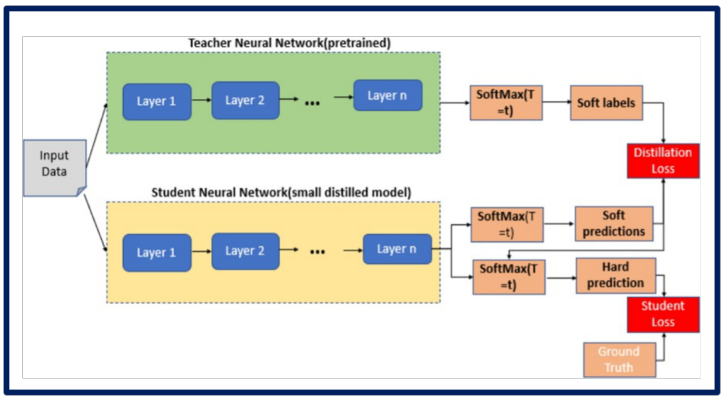

Figure 3: Illustration du principe de distillation de connaissances

Note : Vous avez le libre choix d'identifier et utiliser la ou les méthodes de compression qui répondent aux mieux à vos besoins de déploiement (temps de calcul, mémoire, espace de stockage, etc.). Les coaches peuvent vous aider à faires les choix les plus raisonnables.

4 Conclusions

En conclusions, l'objectif de ce Workshop est de mettre en œuvre des modèles de détection qui fonctionneront en temps réel sur un module « Edge AI » pour maison intelligente, et dont les sorties serviront notamment à orienter une caméra pan-tilt et à envoyer des notifications immédiates. Une fois les objectifs choisis, nous vous invitons à vous partager le travail assez rapidement, de sorte à pouvoir d'une part effectuer le développement des modèles (car il impliquera de jouer avec des bases données, en partie à constituer vous-même), et en parallèle commencer à vous approprier le dispositif multi-caméras (stéréoscopie et contrôle pan-tilt). Ensuite, une fois que vous avez des premières bribes de solutions, vous pourrez travailler en parallèle sur l'amélioration de ces modèles (encore plus de données, modèle éventuellement plus gros, etc) et l'intégration des tâches de compression et optimisation des modèles. Le mécanisme d'envoi d'alertes/notifications ne devrait quant à lui pas vous prendre trop de temps à réaliser, mais ne l'oubliez pas.

ATTENTION : le dimanche après-midi sera consacré à la création de votre présentation finale qui aura lieu devant un jury à 17h30.

5 Quelques liens intéressants

- Quantization

- Distillation de connaissances sur Yolo
- Quantization sur Yolo11
- Types de nombres à faible résolution sous TensorRT
- Quantization sous Pytorch