Rapport Technique MyTeam- Détection d'Objets en Temps Réel

Abdel-Hakim Arab Abhakimarab@gmail.com

March 2, 2025

1 Objectif du Projet

Le projet vise à concevoir un système de détection et de reconnaissance d'objets en temps réel à partir d'un flux vidéo de webcam, en utilisant les solutions de vision par ordinateur de Mediapipe. L'objectif est de détecter des objets courants (comme une tasse ou un téléphone) et, le cas échéant, de classifier précisément ces objets afin d'afficher une étiquette correspondante à l'écran.

2 Choix Techniques et Algorithmes Utilisés

2.1 Détection d'Objets avec Mediapipe Tasks

Pour la détection, nous avons utilisé **EfficientDet-Lite0** via l'API de détection d'objets de Mediapipe Tasks. Ce modèle, qui intègre un backbone EfficientNet-Lite0 et un BiFPN, permet de localiser et de classifier simultanément les objets. Deux modes de fonctionnement ont été testés :

- **Mode synchrone (IMAGE)** : Chaque image est traitée immédiatement, garantissant une réponse rapide.
- **Mode asynchrone (LIVE_STREAM)** : Utilise un callback pour traiter les frames en temps réel, mais peut introduire une légère latence.

Pour notre application, le mode synchrone a été préféré afin d'obtenir des résultats immédiats et une fluidité optimale.

2.2 Classification des Objets

Les modèles de détection, tels qu'EfficientDet-Lite0, intègrent déjà une étape de classification en renvoyant directement le label et le score via les métadonnées du modèle. Nous avons également expérimenté l'intégration d'un classificateur autonome, basé sur un modèle TFLite (par exemple, EfficientNet-Lite0 ou MobileNet), appliqué sur la région d'intérêt (ROI) extraite par le détecteur. Deux approches ont été envisagées :

- Une implémentation manuelle utilisant l'interpréteur TFLite avec un fichier de labels pour mapper l'indice prédit à une étiquette.
- L'API **ImageClassifier** de Mediapipe Tasks, qui renvoie directement le label via les métadonnées.

L'approche intégrée au détecteur s'est révélée la plus simple et la plus fiable.

3 Prétraitement des Données

Les modèles de détection d'objets de Mediapipe gèrent automatiquement le redimensionnement (resizing) et la normalisation des pixels, ce qui simplifie considérablement le pipeline. Pour la classification autonome, il est néanmoins nécessaire de redimensionner manuellement la région d'intérêt à la taille requise par le modèle et, selon le modèle, d'appliquer une normalisation (par exemple, convertir en UINT8 ou normaliser en FLOAT32). Par ailleurs, une conversion de l'image de BGR (OpenCV) en RGB est effectuée pour respecter le format attendu par les modèles TFLite.

4 Quantization et Meilleure Configuration

Plusieurs configurations de quantization ont été testées pour le modèle de détection, en comparant différents compromis entre précision et rapidité :

- **Float32** : Ces versions offrent la meilleure précision, mais sont souvent trop lentes pour une application en temps réel. Leur temps d'inférence élevé peut entraîner une fluidité réduite.
- **Int8** : Les modèles quantifiés en INT8 sont très rapides et légers, ce qui est idéal pour des dispositifs aux ressources limitées. Cependant, cette rapidité s'accompagne d'une légère diminution de la précision, particulièrement dans des environnements complexes ou avec de petits objets.
- **Float16 (EfficientDet-Lite0)** : Cette configuration offre un excellent compromis entre précision et rapidité d'inférence. Pour le modèle EfficientDet-Lite0, la quantization en FLOAT16 permet d'obtenir une fluidité optimale tout en conservant une précision satisfaisante.
- **Autres configurations :** Les modèles EfficientDet-Lite2 (basés sur EfficientNet) et SSD MobileNet V2 (basés sur MobileNet) ont également été évalués. EfficientDet-Lite2 en FLOAT16 offre une meilleure précision grâce à une résolution d'entrée plus élevée, mais avec un temps d'inférence généralement supérieur à celui d'EfficientDet-Lite0 FLOAT16. À l'inverse, SSD MobileNet V2, notamment en version INT8, permet une inférence très rapide, bien que sa précision soit légèrement inférieure à celle des modèles basés sur EfficientNet.

Notre configuration par défaut utilise **EfficientDet-Lite0 en quantization FLOAT16**. Ce choix est justifié par son excellent rapport entre fluidité et précision, tout en assurant une intégration simplifiée via l'API de Mediapipe Tasks. Cette solution représente le meilleur compromis pour le projet, sans nécessiter d'ajustements supplémentaires qui pourraient complexifier le pipeline.