Rapport de Cas pratique

Vérificateur de l’uniforme

Modification et développement d’une application pour vérifier l’uniforme des travailleurs (casque et gilet).

PLESSIS Loïc

14/11/2022

Plan du rapport

- Demande Client / Besoin
    - Contexte du projet
- Présentation de l’existant
    - Existant
    - Description des données
- Solution
    - Choix techniques
    - Description des améliorations apportées
    - Mesures des nouvelles performances
- Conclusion et axes d'amélioration

Contexte :

Les caméras sont devenues un moyen important pour développer plusieurs applications en termes de surveillance et de sécurité.

Une entreprise qui propose des solutions technologiques en caméra de surveillance et de sécurité a développé une application IA capable de détecter et localiser la présence du port d'un casque de chantier ou non à partir d'une vidéo. Cette entreprise cherche à améliorer et étendre cette application pour détecter et/ou localiser le gilet de sécurité, si la personne porte un gilet ou non.

L'application développée par l'entreprise est une application Web sous Streamlit-webrtc, elle est capable d'activer la camera du PC et affichée sur la vidéo le résultat de la détection.

Existant :

- Une application Streamlight-webrtc basée sur un code python permettant de lancer l'application sur une page web.
- Un CNN (Convolutional Neural Network) Yolov5s préalablement entraîné (transfer learning) .
- Un jeu de données tiré de la base de données COCO (Common object in context) , découpé en deux parties : images et labels pour permettre au modèle de fonctionner.

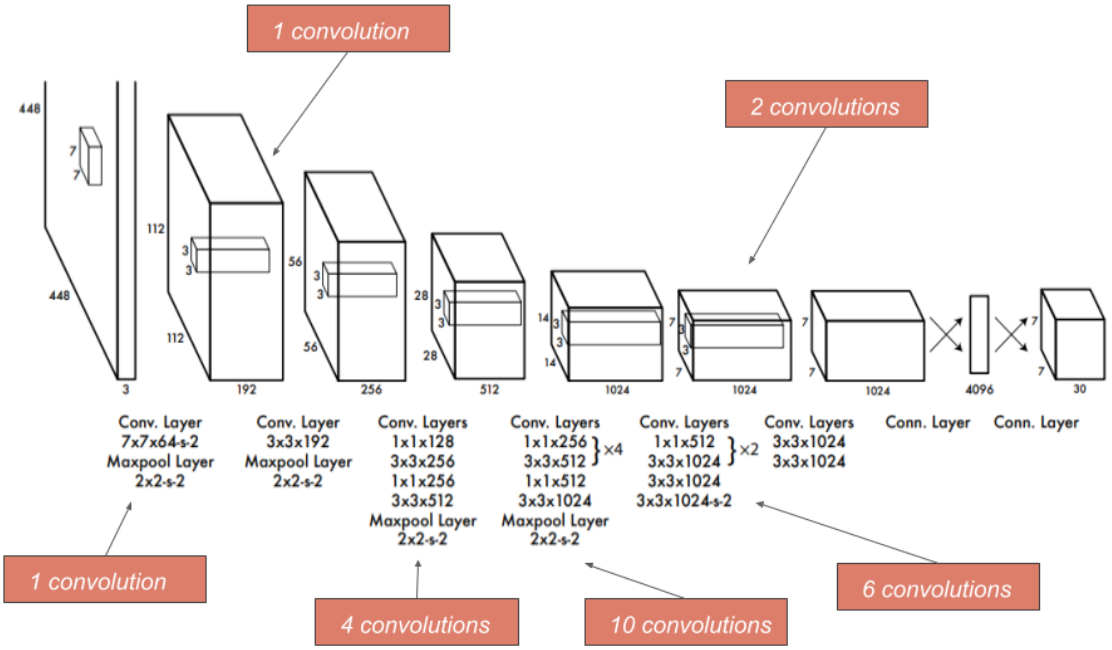
les points à améliorer a partir de l'existant :

- Détection et/ou localisation de la présence/absence du gilet de sécurité.
- Un message d'alerte à afficher sur la page web (en dehors de la vidéo) si une seule personne ne porte pas son casque ou son gilet.

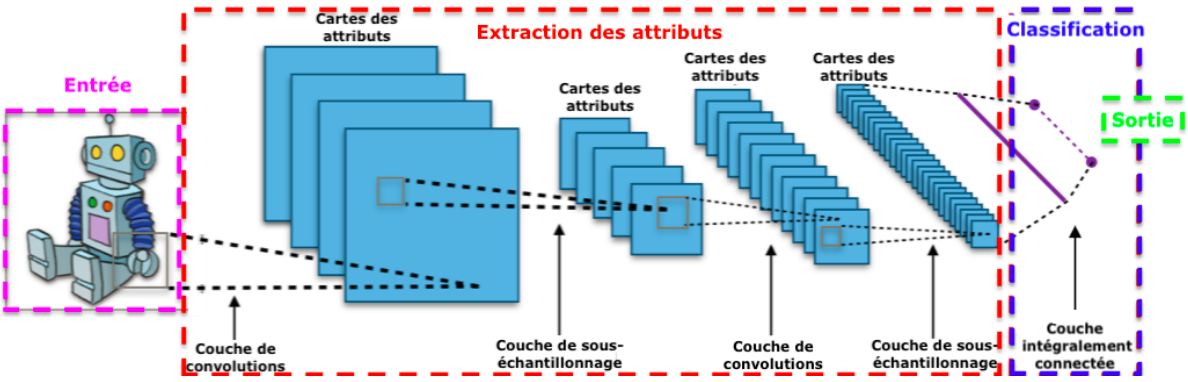
Solution :

Le choix du CNN **YOLOv5s** s'avère pertinent pour répondre au besoin posé par l'entreprise , YOLOv5s est un modèle en open source et est particulièrement bien adapté pour les applications en temps réel et la détection d'objets, il est possible d'entraîner et tester un réseau de neurones convolutif (CNN) YOLOv5s simplement à partir d'un google colab lié à un drive pour y stocker le jeu de données, afin d'en tirer les meilleurs paramètres (ou poids) après avoir entraîné notre modèle.

Schéma de l'architecture du modèle CNN YOLOv5:



Exemple de schéma d'architecture d'un Réseau de neurone convolutif CNN



Modifications apportées à l'existant :

Application streamlit :

Modification du code python pour afficher le message de conformité de l'équipement le résultat est stocké dans un fichier texte , celui-ci est écrasé à chaque fois que l'utilisateur appuie sur le bouton pour lancer l'application;

Retraitement des données :

- **labellisation** : ajout de nouvelles images labellisées (+ de 1000 au total) à l'aide de l'outil **makesense.ai**
- ajout de plus de 50 nouvelles images prises avec la webcam de l'ordinateur (avec casque/sans casque , avec gilet/sans gilet)
- données sont réparties en **4 classes**:
    - avec casque
    - avec gilet
    - sans casque
    - sans gilet

pour information : Yolov5 applique naturellement une data augmentation sur le dataset donné avec le paramètre : "hyp-scratch-low".

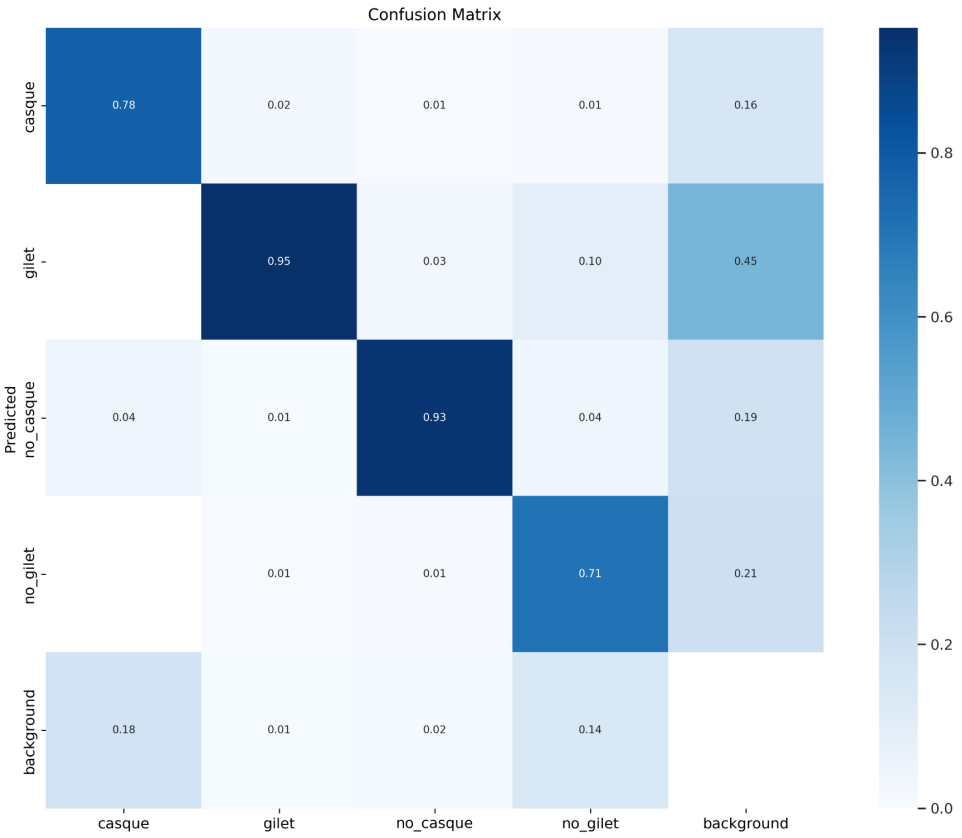
Fonctionnement de l'application

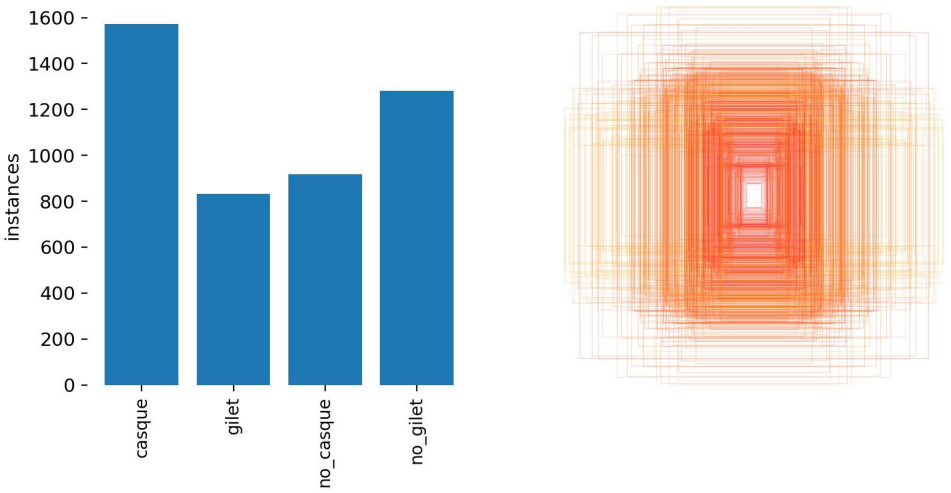
L'utilisateur est face à la caméra de l'ordinateur et l'application indique à ce dernier par un message si l'équipement est complet ou non complet (casque et gilet) .

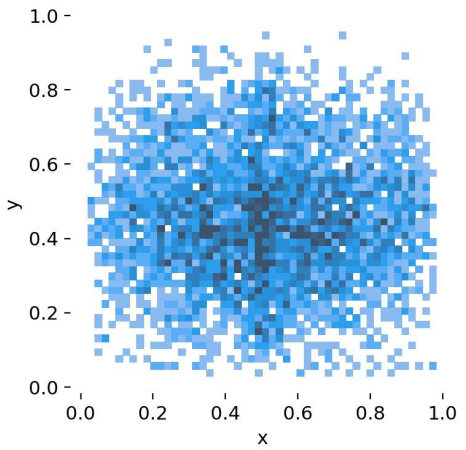
Mesure de performances du modèle :

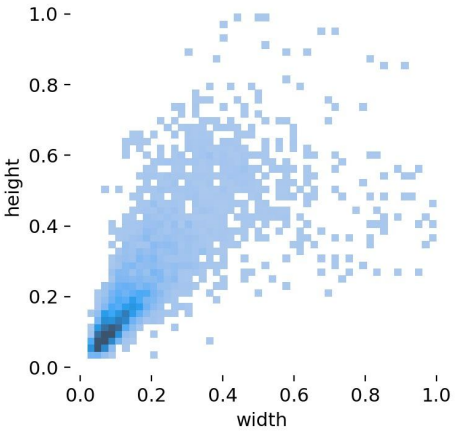
tous les metrics et graphiques du modèle sont disponibles intégralement dans le répertoire github du projet.

Ci dessous la **matrice de confusion** des résultats de prédictions du modèle, ainsi que la **répartition du nombre de labels** pour les 4 classes : casque ,gilet ,casque_NO, gilet_NO.


Confusion Matrix
casque
gilet
Predicted
no_casque
no_gilet
background
0.78
0.02
0.01
0.01
0.16
0.95
0.03
0.10
0.45
0.04
0.01
0.93
0.04
0.19
0.01
0.01
0.71
0.21
0.18
0.01
0.02
0.14
casque
gilet
no_casque
no_gilet
background
0.8
0.6
0.4
0.2
0.0


instances
1600
1400
1200
1000
800
600
400
200
0
casque
gilet
no_casque
no_gilet


y
x
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


height
width
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

En conclusion

L’application est fonctionnelle et répond au besoin. Cette dernière pourrait cependant être améliorée en augmentant par exemple la taille du jeu de données ou en utilisant un autre CNN type VGG16, ou encore en utilisant un framework autre que Streamlight-webrtc.