Rapport du cas pratique

1. Objectif du cas pratique :

L'objectif du cas pratique est d'améliorer une application de détection d'images. L'utilisateur doit porter un casque et un gilet de chantier. L'application renvoie un message confirmant ou non le port des deux objets.

L'application développée par l'entreprise utilise *Streamlit-webrtc*, elle utilise la webcam de l'ordinateur et affiche le résultat de la détection.

Notre application répond parfaitement aux demandes de l'entreprise :

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

Nous avons choisi d'utiliser Yolov5 (version 6.2 : https://github.com/ultralytics/yolov5) qui est très performant dans la détection d'objet. Il dispose par ailleurs de modèles pré-entraînés sur le *dataset* COCO (https://cocodataset.org/#home). Ce choix nous a permis de gagner du temps dans l'entraînement du modèle : étant donné qu'il s'agit de *transfer learning*, nous disposons d'une architecture et d'un modèle déjà pré-entraîné et n'avons plus qu'à personnaliser notre modèle.

Nous avons entraîné notre modèle sur plus de 1000 photos. Le plus long a été des rassembler les images et de les labelliser. Peut-être aurions-nous gagné du temps en utilisant un modèle d'apprentissage semi-supervisé, en mélangeant des données labelisées et des données non-étiquetées. Cependant il aurait fallu également consacrer beaucoup de temps à la mise en place de ce modèle, aussi la solution de Yolov5 nous semblait la plus appropriée.

2. Traitement des images :

Les images de départ n'étant pas assez nombreuses, un premier entraînement du modèle nous donnait de mauvais résultat.

Pour augmenter le nombre de données nous avons choisi :

- D'intégrer de nouvelles images
- D'avoir recours à la *data augmentation*.

Nous avons également opté pour les solutions suivantes :

- Varier les couleurs des gilets et de casques :
    - Gilets de couleurs différentes (très bon résultat sur la couleur jaune, correcte sur orange, difficile avec d'autres couleurs, comme le rouge ou le vert).
    - Casque de couleurs différentes : blanc, rouge, orange, bleu... Il a fallu cerner les couleurs les moins fréquentes pour les représenter plus largement dans le jeu de données. Si un casque vert était présenté, le modèle risquait de ne pas le reconnaître comme tel.
- Varier les formes des gilets et des casques (de chantier) :
    - Formes du gilet : s'il est ouvert, il présente le risque de ne pas être reconnu.
    - Formes du casque : certains casques de chantiers présentent une visière de protection (notamment pour protéger les yeux de l'utilisateur d'éventuelles projections). Ce genre de casque peut être confondu avec des casques de motos (fermés) par exemple.
    - Nous avons également testé la pertinence du modèle avec différents types de chapeaux pour voir si l'apprentissage se révélait pertinent ou non. Ainsi un casque de moto n'est pas reconnu, mais un casque de vélo peut tromper le modèle.

Nous avons également fait attention aux points suivant :

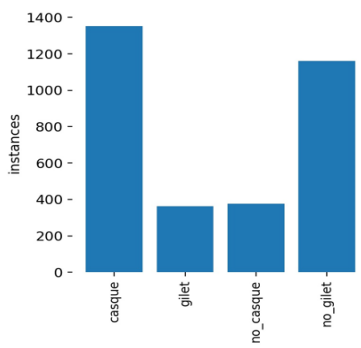
- Variation du sexe (les femmes sont plus rares sur les chantiers selon les statistiques) : il fallait que le modèle puisse reconnaître une femme, s'il y avait trop peu de données, le résultat ne pouvait être que faussé. Pour éviter cette situation nous avons augmenté le nombre de photos de femmes.
- Variation de l'origine ethnique : si le modèle était uniquement entraîné sur des données représentant le type européen il produirait de mauvais résultats sur d'autres données.
- Variation de l'âge : la donnée la plus difficile à obtenir, car il y a peu de représentation de personnes âgées (ou de personnes ayant simplement des cheveux blancs) sur un chantier. Nous avons cependant réussi à augmenter le nombre d'images dans cette catégorie.

3. Labélisation des données :

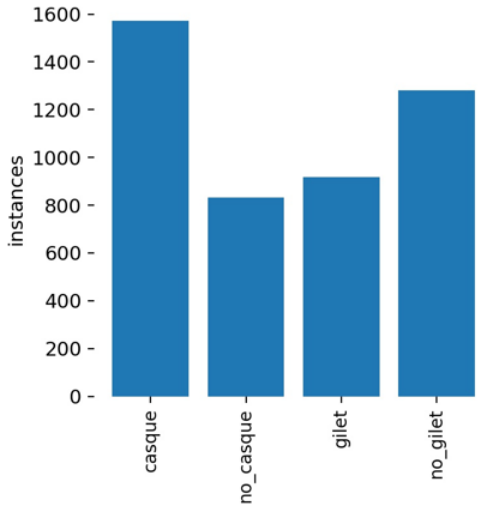
Les données ont été réparties en 4 classes :

- avec casque
- avec gilet
- sans casque
- sans gilet

Pour notre premier modèle, nous avons constaté un déséquilibre entre les différents labels : il y avait moins de gilets représentés et peu de personnes ne portant pas un casque de sécurité.



Les personnes portant un gilet et pas de casque auraient été plus difficiles à détecter pour notre modèle, car il n'avait pas été entraîné avec autant d'images que les autres labels. Pour obtenir de meilleurs résultats, nous avons choisi d'augmenter le nombre d'images des labels en question, comme l'illustre le diagramme suivant.



4. Description du modèle :

Entraînement du modèle :

Notre modèle a été entraîné sur plus de 1000 photos, avec un *batch-size* fixé à 16 et 300 *epochs*. Plusieurs tentatives ont été réalisées avec un échantillon plus ou moins élevé pour le *batch* et en variant le nombre d'*epochs*. Le modèle que nous avons choisi de retenir pour ce cas pratique n'est pas le meilleur sur le papier (ses résultats de prédiction pouvant être améliorés), mais il fonctionne correctement en pratique. Entre la performance et l'efficacité, nous avons choisi la deuxième option.

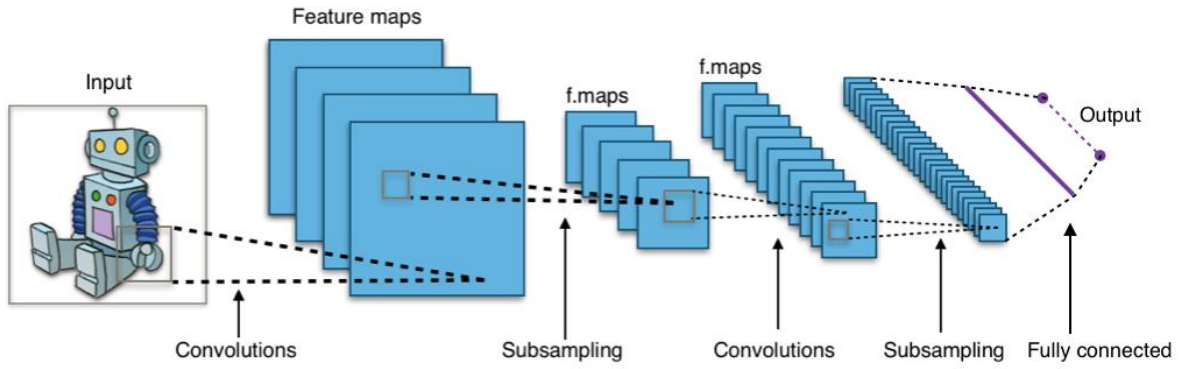
Structure du modèle :

Les modèles pré-entraînés de Yolov5 sont des réseaux de neurones de types convolutifs (*CNN* : *Convolutional neural network*). Ce type de réseau de neurones est le plus efficace en ce qui concerne la reconnaissance d'images.

Prenons un exemple simple : une image de 250 pixels sur 250 contient 62 500 pixels (250*250=62500). Si l'on attribue un pixel à un neurone, notre réseau devrait contenir 62 500 neurones. Pour un réseau multicouche, avec seulement 3 couches, cela nous

donnerait 187 500 neurones. Le réseau serait très compliqué, voire impossible à mettre en place, et ne serait pas rentable.

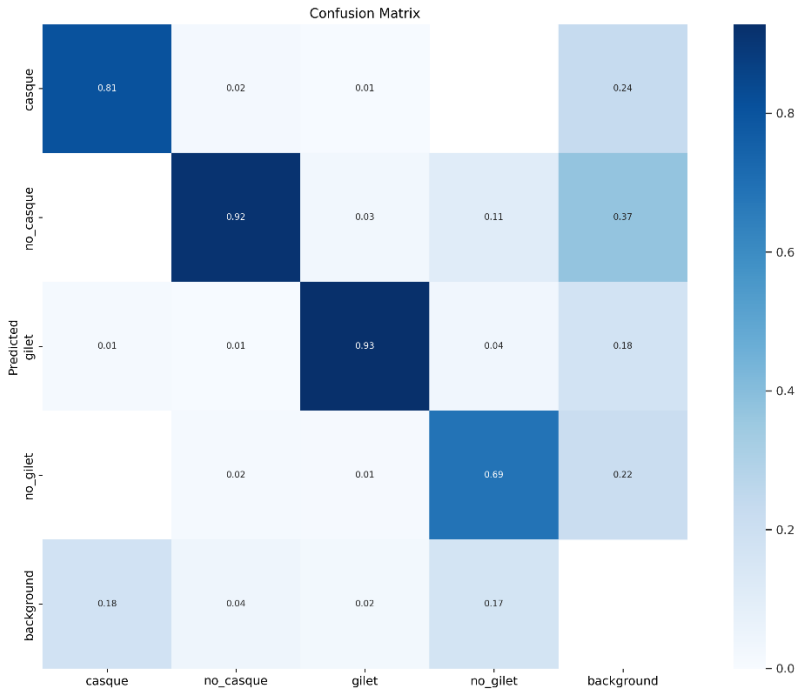
La solution est de passer par un *CNN* : la première partie du réseau vise à extraire les caractéristiques importantes des images (*feature maps*) et de réduire au fur et à mesure l'information circulant dans le réseau à quelques échantillons qui sont ensuite transmis à la deuxième partie du réseau, qui a pour but d'effectuer la phase de reconnaissance.



(Source de l'image: *Wikipédia*)

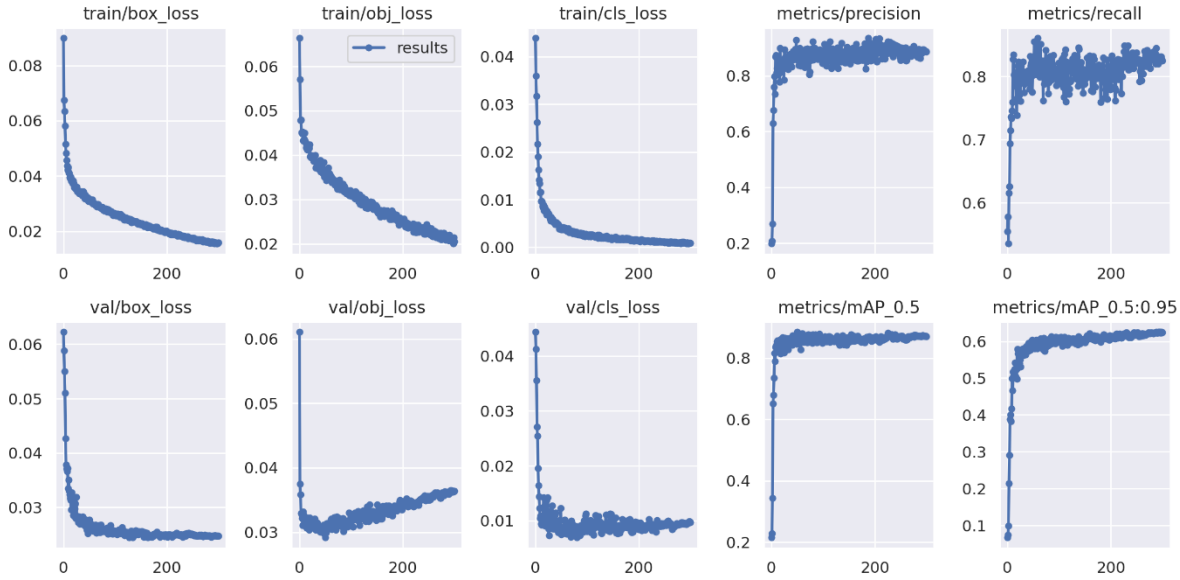
Métriques du modèle :

a) Matrice de confusion :



La matrice de confusion nous renvoie de très bon résultat pour les prédictions sans casque et les prédictions avec gilet (respectivement 0.92 et 0.93). Les résultats baissent avec les prédictions du casque (0.81) et chutent drastiquement avec les prédictions de sans gilet (0.69). Pour améliorer les performances de notre modèle, il faudrait renforcer l'apprentissage sur ces deux dernières catégories, notamment en rajoutant des images adéquates.

b) Fonction de perte et précision

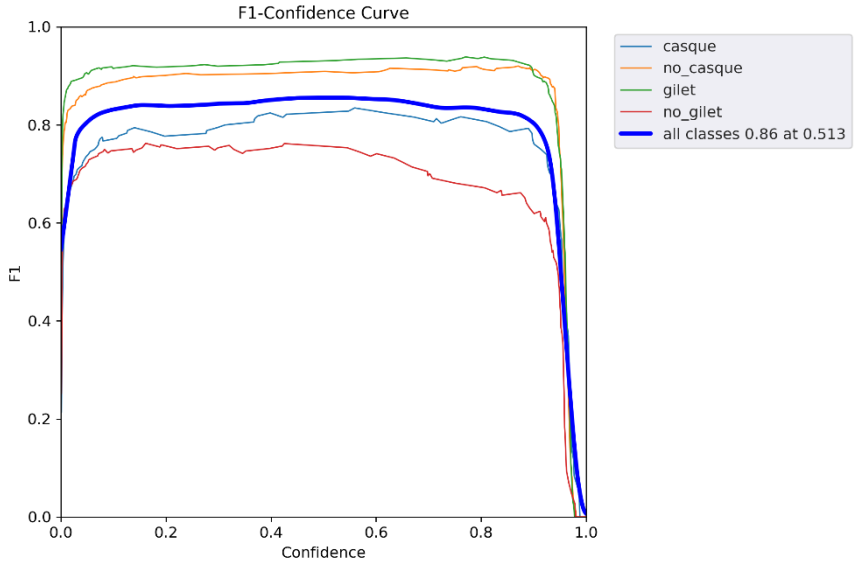


Ce document nous permet de comprendre que notre fonction coût (ou de perte : *loss*) décroît avec le temps, c'est-à-dire plus le nombre d'*epochs* augmente. Cela est vrai et très visible sur notre jeu d'entraînement (*train*) mais l'est beaucoup moins sur notre jeu de validation (*val*) : il faut se méfier du surapprentissage de notre modèle.

Notre taux de précision (Vrai positif / Vrai positif + Faux positif), c'est-à-dire combien de fois nous avons correctement classifié notre prédiction positive est bon très rapidement, il gagne quelques points en augmentant le nombre d'*epochs* mais cela reste très limité.

Notre taux de *recall*, c'est-à-dire combien de fois nous avons classifié à tort quelque chose comme faux (Vrai positif / (Vrai positif + Faux négatif). Ces deux métriques nous permettent d'avoir une meilleure idée de la pertinence de notre modèle que la simple *accuracy*.

c) F1 score



Rassemble la précision et le *recall* et fait la moyenne des deux (2 * ((Precision * Recall) / (Precision + Recall))). Le F1 score a l'avantage de produire un équilibre entre les deux métriques. Un bon résultat pour le F1 score signifie qu'il y a peu de faux positifs et peu de faux négatifs

5. L'application :

Fonctionnement

L'utilisateur doit se présenter face caméra. S'il a un gilet et un casque, alors l'application lui indique par message que sa tenue est correcte. Si un des éléments manque, alors l'application affiche le message contraire.

Affichage du message

Pour permettre d'afficher le message, nous avons choisi de stocker le résultat dans un fichier texte et d'écraser le résultat à chaque nouvelle reconnaissance.

L'application affiche le message indiquant le bon respect (ou non) du port du casque et du gilet à partir du fichier texte.

6. Conclusion :

Au terme de ce cas pratique, nous pouvons affirmer que notre application fonctionne et répond au cahier des charges. Les résultats de la reconnaissance sont corrects. En pistes d'amélioration nous pouvons proposer :

- D'entraîner notre modèle à détecter les personnes, puis un autre modèle pour détecter les casques et les gilets.
- D'entraîner un nouveau modèle avec encore plus de données et en variant les paramètres d'apprentissage.
- De trouver un autre *framework*, peut-être plus performant que *streamlit*.
- De se tourner vers un modèle *VGG-16* et d'observer la qualité des résultats obtenus.