



UNIVERSITÉ LUMIÈRE LYON 2

DÉPÔT GITHUB
RAPPORT

Traffic Sign Detection

Élèves :

Nathan GRIMAUT
Victor SIGOGNEAU

Enseignant :

Carlos CRISPIM-JUNIOR

23 février 2024

Table des matières

1	Introduction	2
2	YoloV5	2
2.1	Architecture	2
2.2	Choix du modèle pré-entraîné	3
3	Pré-traitement des Données	4
4	Re-Entraînement du Modèle	6
4.1	Méthodologie	6
5	Expérimentations et Résultats	7
5.1	Validation des modèles	7
5.2	Résultats	8
5.3	Comparaison	9
6	Conclusion	10

1 Introduction

La détection et la reconnaissance de panneaux de signalisation sont essentielles pour la sécurité routière et la conduite autonome. Ce projet s'inscrit dans le cadre du cours de deep learning et se concentre sur l'utilisation du modèle YOLOv5, un modèle pré-entraîné, pour détecter et classifier quatre classes de panneaux :

- Limitation de vitesse
- Stop
- Sens interdit
- Cédez le passage.

Nous explorons l'efficacité des CNN pour cette tâche, en mettant l'accent sur la précision et la robustesse du modèle dans des environnements routiers réels. Nous procéderons au ré-entraînement des poids du modèle en utilisant nos propres images, dans le but d'améliorer sa performance, notamment pour la détection précise des classes spécifiques de panneaux de signalisation que nous ciblons.

2 YoloV5

2.1 Architecture

YOLOv5 est un algorithme de détection d'objets en temps réel. Il utilise une architecture de réseau de neurones convolutifs pour détecter et localiser différents objets dans une image. Voici les principales composantes de l'architecture YOLOv5 :

- **Backbone** : YOLOv5 utilise un backbone constitué de couches convolutionnelles pré-entraînées. Ces couches sont responsables de l'extraction des caractéristiques de l'image en entrée.
- **Neck** : Le "neck" est une série de couches qui agissent comme un pont entre le backbone et les têtes de détection. Ces couches sont utilisées pour fusionner les caractéristiques provenant de différentes échelles spatiales.
- **Head** : Puis plusieurs têtes de détection pour prédire les boîtes englobantes et les classes des objets. Chaque tête de détection est responsable de prédire les boîtes englobantes à une certaine échelle.
- **Loss function** : La fonction de perte utilisée dans YOLOv5 est une combinaison de différentes composantes, y compris la perte de localisation, la perte de classification et la perte d'objectivité. Cette fonction est optimisée pour minimiser l'erreur de prédiction du modèle.

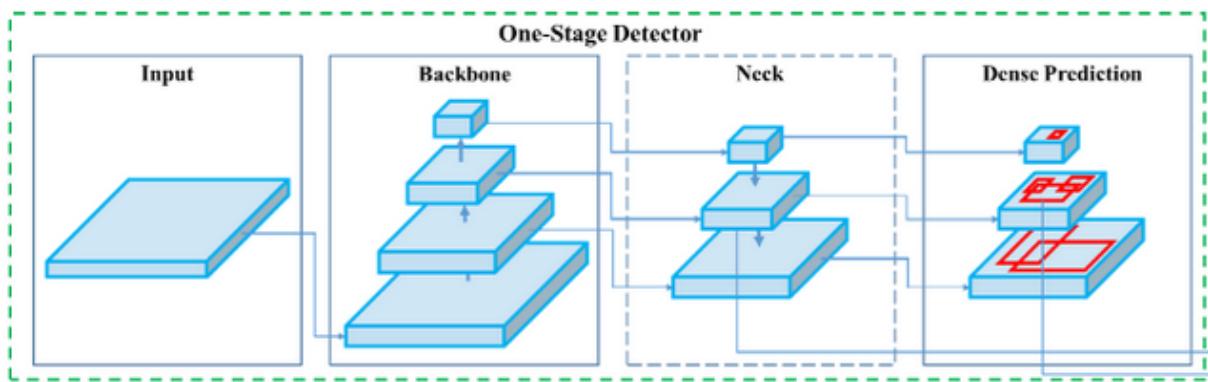


FIGURE 1 – Architecture de YOLOv5

La figure 2 montre une représentation schématique de l'architecture de YOLOv5.

2.2 Choix du modèle pré-entraîné

Avec YOLOv5, il est possible d'utiliser des modèles pré-entraînés, et il existe une multitude de modèles disponibles, chacun ayant ses forces et ses faiblesses.

Nous avons choisi le modèle *small* pour plusieurs raisons. Tout d'abord, il nous permet de mettre en œuvre un nombre d'époques plus élevé et de converger plus rapidement vers des poids qui nous conviennent. En raison de sa taille plus petite et de sa complexité moindre par rapport aux modèles *medium* et *large*, le modèle *small* peut être entraîné plus rapidement, ce qui nous permet de profiter pleinement des ressources de calcul disponibles.

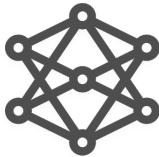
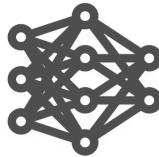
			
Small YOLOv5s	Medium YOLOv5m	Large YOLOv5l	XLarge YOLOv5x
15 MB _{FP16} 2.4 ms _{V100} 37.0 mAP _{coco}	42 MB _{FP16} 3.4 ms _{V100} 44.3 mAP _{coco}	92 MB _{FP16} 4.4 ms _{V100} 47.7 mAP _{coco}	170 MB _{FP16} 6.9 ms _{V100} 50.8 mAP _{coco}

FIGURE 2 – Spécificités des modèles YoloV5

Tous les modèles YOLOv5 sont pré-entraînés sur le dataset COCO, qui est un ensemble de données largement utilisé contenant un grand nombre d'images annotées avec divers objets répartis dans différentes classes. Ce pré-entraînement initial sur un ensemble de données diversifié permet aux modèles YOLOv5 de capturer des caractéristiques génériques des objets et de bénéficier d'une certaine capacité de généralisation à différentes tâches de détection d'objets.

En optant pour le modèle *small*, nous recherchons un équilibre entre la taille du modèle, la vitesse d'entraînement et la performance de détection des objets. Nous sommes conscients que le modèle *small* peut avoir une habileté de capture de caractéristiques plus limitée par rapport aux autres modèles, mais nous considérons que cela est compensé par sa rapidité d'entraînement et sa capacité à s'adapter à nos besoins spécifiques en matière de détection d'objets. De plus, nos classes n'étant pas très complexes et facilement reconnaissables (principe logique pour des panneaux de signalisation), un modèle moins complexe devrait être plus adapté, surtout dans un contexte où le nombres d'images d'entraînement est limité, un modèle trop complexe n'est pas adapté.

3 Pré-traitement des Données

L'ensemble de données utilisé dans ce projet a été constitué à partir d'images extraites de Google Maps. Chaque image a une résolution de **448** pixels sur **448** pixels, ce qui permet d'obtenir une qualité suffisante pour la détection des panneaux de signalisation.

L'ensemble de données est composé de quatre classes de panneaux de signalisation :

- **Limitation de vitesse** : Nous disposons de 25 images contenant au total 24 panneaux de limitation de vitesse, répartis comme suit : 7 panneaux de limitation à 30 km/h, 6 panneaux à 50 km/h, 7 panneaux à 70 km/h, 1 panneau à 90 km/h et 2 panneaux à 110 km/h.
- **Stop** : Nous disposons de 27 images contenant chacune un panneau d'arrêt.
- **Sens interdit** : Nous disposons de 24 images contenant chacune un panneau de sens interdit.
- **Cédez le passage** : Nous disposons de 24 images contenant chacune un panneau de cédez le passage.



FIGURE 3 – Exemples des images d'entraînements

Chaque image a été annotée avec les coordonnées exactes des panneaux de signalisation présents dans l'image, à l'aide du site web www.makesense.ai. Ces annotations ont été stockées dans des fichiers au format `.txt`, associant chaque image à ses coordonnées de panneaux correspondantes.



FIGURE 4 – Image et son label

Aussi Yolo permet une vue globale de notre jeu de données, en affichant des informations cruciales pour un modèle de deep learning.

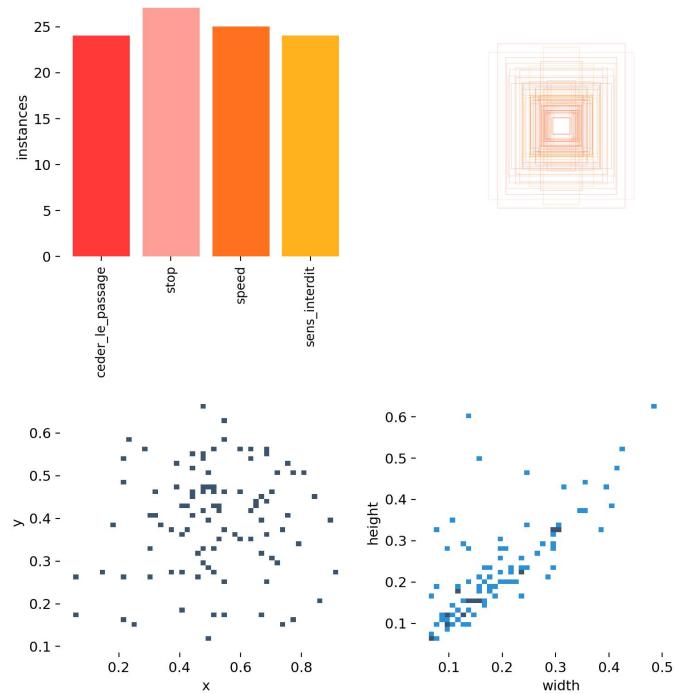


FIGURE 5 – Résumé du jeu de données

Comme le nombre d'image par classe (en haut à gauche), une synthèse des positions des étiquetages dans les images (haut à droite), les positions des labels avec les coordonnées x et y et enfin la taille des labels.

4 Re-Entraînement du Modèle

Dans cette section, nous détaillons le processus d'entraînement du modèle de détection de panneaux de signalisation. Nous commençons par présenter les paramètres d'entraînement utilisés, puis nous décrivons la méthodologie mise en œuvre.

Pendant l'entraînement, nous avons utilisé la technique du transfert d'apprentissage (ré-entraînement de la dernière couche) en initialisant les poids du modèle à partir du modèle pré-entraîné YOLOv5, puis en ajustant ces poids sur notre ensemble de données spécifique. Ce qui nous permet même avec un très petit nombre d'images de quand même obtenir des performances convaincantes. Sans ça il faudrait au moins 10 fois plus de données pour atteindre les mêmes performances.

4.1 Méthodologie

Nous avons divisé notre ensemble de données en ensembles distincts pour l'entraînement, la validation et les tests, avec des proportions respectives de 84%, 8% et 8%. Les classes sont bien évidemment équilibrés dans chaque dossier, afin de parvenir aux meilleurs résultats.

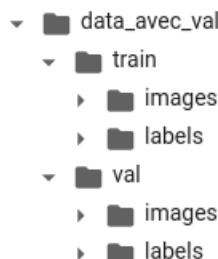


FIGURE 6 – Arborescence des données

Plusieurs modèles et paramétrages ont été testés :

- yoloV5s => 197 époques => batch 8
- yoloV5s => 800 époques => batch 8
- yoloV5m => 268 époques => batch 8
- yoloV5m => 800 époques => batch 8

NB : À chaque fois le taux d'apprentissage initial a été fixé à 0,001, et il a été adapté dynamiquement pendant l'entraînement à l'aide d'une méthode d'ajustement automatique.

Pour ce qui est du batch nous avons préféré le mettre à une valeur petite étant donné le faible nombre d'images que nous disposons. Le batch est le nombre d'exemples donnés à l'algorithme avant de mettre à jour les poids, un batch faible permet donc de converger un peu plus vite vers des poids optimaux.

5 Expérimentations et Résultats

Dans cette section, nous présentons les résultats de nos expérimentations, en mettant en évidence la performance initiale du modèle, les résultats après l'apprentissage par transfert et en fournissant des analyses quantitatives et qualitatives.

5.1 Validation des modèles

Après ré-entraînement d'un modèle, il est possible d'avoir des métriques d'évaluations de performances (calculé à partir des images de validation). Analysons le modèle le moins puissant (en théorie) par rapport au plus puissant.

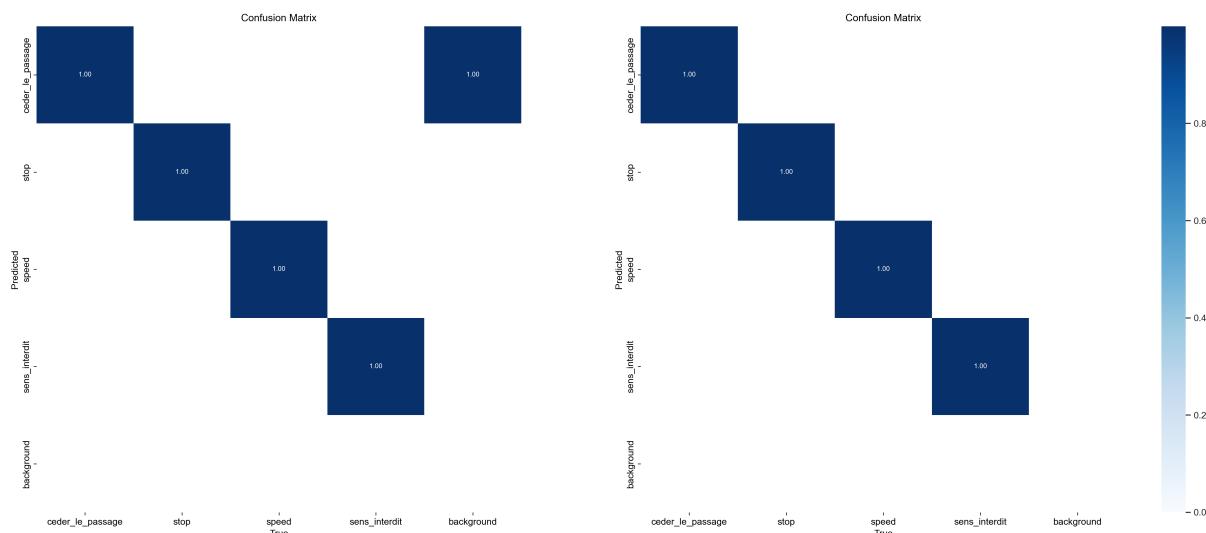


FIGURE 7 – Modèle S -> 197 époques

FIGURE 8 – Modèle M -> 800 époques

Ici, les modèles détectent donc toutes les classes du jeu de validation, sans aucune faute. Cependant, le petit modèle (à gauche), a tendance à halluciner la présence des panneaux "cédez le passage", alors que qu'il y a rien du tout. Un nombre d'époque suffisant règle ce problème.

Un graphique F1-Confidence est un outil utile pour évaluer la performance d'un modèle de classification. Il représente la relation entre la mesure F1 et le niveau de confiance des prédictions du modèle. Si la ligne de tendance monte vers le coin supérieur droit du graphique, cela indique que le modèle obtient de meilleurs F1 scores à des niveaux de confiance plus élevés. Cela signifie que le modèle est plus performant lorsqu'il est plus confiant dans ses prédictions.

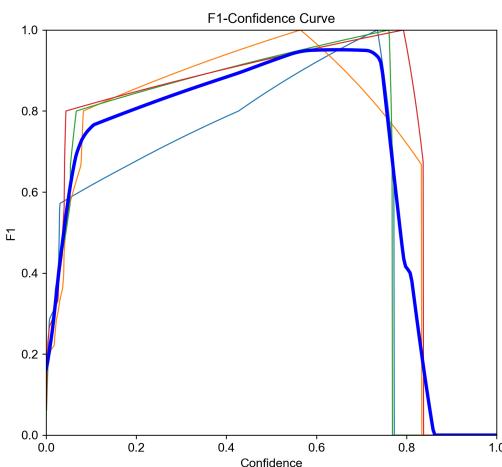


FIGURE 9 – Modèle S -> 197 époques

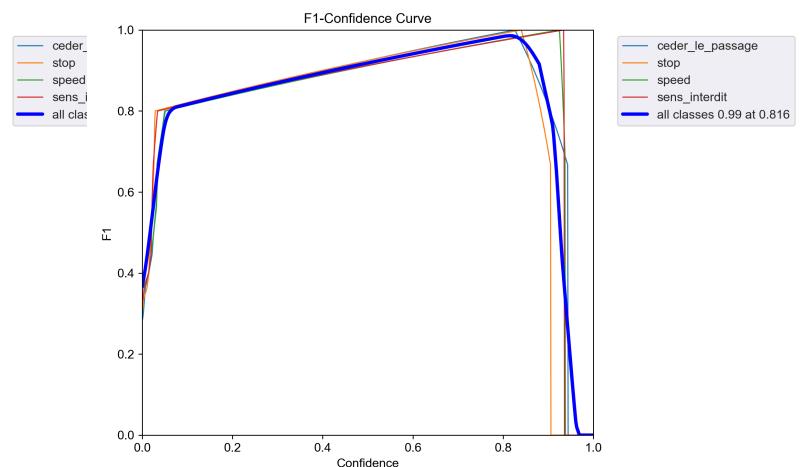


FIGURE 10 – Modèle M -> 800 époques

Si la ligne de tendance est plate ou descendante, cela indique que le modèle obtient des F1 scores similaires ou inférieurs à des niveaux de confiance plus élevés. Cela peut indiquer des problèmes tels que la surconfiance du modèle ou des lacunes dans sa capacité à discriminer correctement les exemples difficiles. Ici nous voyons qu'il y a une fracture du F1-score lorsque les deux modèles sont sur-confiants. Aussi le modèle puissant à des résultats plus stable pour toutes les classes contrairement à l'autre.

5.2 Résultats

Pour tester les modèles, nous avons essayé de les pousser dans leurs retranchements. En essayant faire les faire généraliser, par exemple en utilisant des panneaux de limitations de vitesses qu'il n'as jamais vu (panneaux 130 et 110), mais aussi mettre plusieurs panneaux différents dans une même image.



FIGURE 11 – Généralisation panneaux de vitesse

Sur cette image, l'algorithme détecte bien les deux panneaux de limitation de vitesse avec une confiance élevée. Même sans avoir vu des panneaux 130 ou 110.



FIGURE 12 – Différents labels dans une même image

Ici notre modèle détecte aussi les deux labels dans l'image même en ayant jamais vu deux labels différents dans la même image.

5.3 Comparaison

Afin de comparer les modèles nous avons pour chacun créer la matrice de confusion sur l'échantillon test, 8 images au total, dont des images qui possèdent plusieurs labels à détecter.

		PREDICTED				
		Interdit	Stop	Speed	Cédez	Background
TRUE	Interdit	2	1	0	0	0
	Stop	0	2	0	0	0
	Speed	0	0	1	0	1
	Cédez	0	0	0	2	0
	Background	0	0	1	0	0

FIGURE 13 – Matrice de confusion pour le modèle S avec 197 époques

		PREDICTED				
		Interdit	Stop	Speed	Cédez	Background
TRUE	Interdit	3	0	0	0	0
	Stop	0	2	0	0	0
	Speed	0	0	3	0	0
	Cédez	0	0	0	2	0
	Background	0	0	1	0	0

FIGURE 14 – Matrice de confusion pour le modèle S avec 800 époques

Avec ces matrices de confusion nous pouvons calculer le taux de précision sur chacun des modèles.

Modèle	Précision
Modèle S => 197 époques	0.7
Modèle S => 800 époques	0.9090
Modèle M => 268 époques	0.8181
Modèle M => 800 époques	0.9090

TABLE 1 – Comparaison des précisions des quatre modèles

Bien sûr pour des résultats plus précis il nous faudrait plus de données test, mais cela n'est pas possible, nous avons privilégié l'entraînement des modèles au vu de la faible quantité de données.

6 Conclusion

En conclusion, nos expérimentations ont démontré que l'utilisation du modèle YOLOv5 pour la détection de panneaux de signalisation a donné des résultats globalement satisfaisants. Nous avons réussi à obtenir des performances élevées pour la plupart des classes de panneaux, avec une précision et un rappel élevés après l'apprentissage par transfert. Cependant, nous avons observé des difficultés dans la détection des panneaux dans des situations complexes, notamment lorsque les images contenaient de nombreux panneaux ou des angles de vue compliqués.

Une des principales difficultés rencontrées était la confusion entre les classes de panneaux, en particulier entre le sens interdit et le stop. Ce problème a souligné la nécessité de disposer d'un ensemble de données plus diversifié et d'explorer des techniques de régularisation plus avancées pour améliorer la robustesse du modèle dans des scénarios complexes.

Malgré ces défis, notre travail a contribué à démontrer l'efficacité de l'utilisation de l'apprentissage par transfert avec YOLOv5 pour la détection de panneaux de signalisation.