



25 MAI 2025

2ÈME ANNÉE SN

---

## Rapport du Projet d'Apprentissage

---

*Élèves :*

Baptiste RAJAONAH  
Théo NOUET  
Laura LARGEAUD  
Lisa PRAINo

*Enseignant*  
A. CARLIER

## Table des matières

<b>1 Composition du Groupe</b>	<b>2</b>
<b>2 Description du Sujet</b>	<b>2</b>
<b>3 Constitution de la base de donnée</b>	<b>2</b>
3.1 Méthodologie pour l'Acquisition et l'Annotation des Données . . . . .	2
3.1.1 Acquisition des Données . . . . .	2
3.1.2 Annotation . . . . .	2
3.2 Méthodologie de Partitionnement des Données . . . . .	2
3.3 Pronostic . . . . .	3
3.3.1 Difficulté du Problème . . . . .	3
3.3.2 Résultats Attendus . . . . .	3
3.4 Script de Chargement des Données . . . . .	3
3.5 Exemples de la Base de Données . . . . .	3
<b>4 Base de données (Modification)</b>	<b>3</b>
<b>5 Elaboration d'une solution</b>	<b>3</b>
5.1 Entraînement classique du modèle . . . . .	4
5.1.1 Analyse de la courbe de perte d'entraînement . . . . .	4
5.2 Amélioration de l'entraînement du modèle . . . . .	5
5.2.1 Changement d'échelle (scale) . . . . .	5
5.2.2 Augmentation de la luminosité (hsv_v) . . . . .	5
5.2.3 Rotation des images (degrees) . . . . .	5
5.2.4 Augmentation combinée . . . . .	6
<b>6 Analyse des résultats</b>	<b>6</b>
6.1 Sans augmentation de données . . . . .	6
6.1.1 Résultats et hypothèses . . . . .	6
6.1.2 Observations . . . . .	8
6.2 Augmentation de données avec zoom et dézoom . . . . .	9
6.3 Résultats en augmentant les données avec des images "de nuit" . . . . .	11
6.4 Résultats en augmentant les données avec des images tournées . . . . .	13
6.5 Résultats en augmentant les données avec des images modifiées aléatoirement . . . . .	14
<b>7 Conclusion</b>	<b>16</b>
<b>8 Bibliographie</b>	<b>16</b>

# 1 Composition du Groupe

**Nom du Projet :** Détection de places de parking

**Membres du groupe :** Baptiste RAJAONAH, Théo NOUET, Laura LARGEAUD, Lisa PRAINo

**Lien vers la base de données :** <https://github.com/lauralargeaud/Projet-parking>

# 2 Description du Sujet

Le manque de places de stationnement est un problème courant dans les zones urbaines. Ce projet vise donc à développer un modèle capable de détecter automatiquement les places de parking libre ou occupée à partir d'images capturées par des caméras de surveillance. Dans notre cas nous utilisons donc des images satellites.

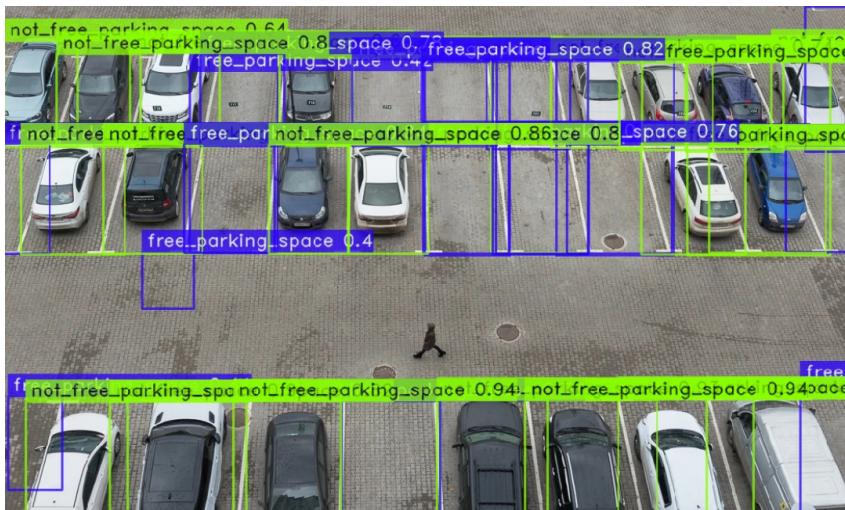


FIGURE 1 – Exemple de résultat final souhaité

# 3 Constitution de la base de donnée

## 3.1 Méthodologie pour l'Acquisition et l'Annotation des Données

### 3.1.1 Acquisition des Données

Les données ont été obtenues à partir de différentes images trouvées sur internet. Nous n'avons pas trouvé de bases de données gratuite déjà faite.

### 3.1.2 Annotation

Nous avons labélisé nos images sur le site : <https://www.makesense.ai> via lequel nous avons pu exporter nos données labélisé directement au format YOLO.

Deux labels ont été définis :

- Place libre
- Place occupée

## 3.2 Méthodologie de Partitionnement des Données

Pour entraîner et évaluer notre modèle, nous avons divisé les données en trois ensembles distincts :

- **Entraînement (70%)** : Utilisé pour l'apprentissage du modèle. Il contient un plus grand nombre de données permettant au modèle de s'ajuster aux différentes variations présentes dans les données.
- **Validation (15%)** : Utilisé pour ajuster les hyperparamètres et éviter le sur-apprentissage (overfitting). Il permet d'évaluer les performances du modèle en cours d'apprentissage.

- **Test (15%)** : Réservé à l'évaluation finale du modèle après son entraînement. Il sert à mesurer ses performances sur des données qu'il n'a jamais vues.

### 3.3 Pronostic

#### 3.3.1 Difficulté du Problème

Le problème est relativement complexe, car différencier une place de parking vide d'une portion de route peut être difficile. La détection des places occupées pourrait être plus simple, bien qu'elle pose aussi des défis, notamment la distinction entre les voitures stationnées et celles en circulation sur la chaussée.

#### 3.3.2 Résultats Attendus

Nous supposons que les places occupées seront plus souvent détectées que les places libres.

### 3.4 Script de Chargement des Données

Voici le code à exécuté (disponible sur GitHub) pour charger notre base de données. Celle-ci est déjà formatée pour être utilisée avec YOLO :

```
!git clone https://github.com/lauralargeaud/Projet-parking.git
```

Cela permet de récupérer rapidement toutes les images annotées.

### 3.5 Exemples de la Base de Données

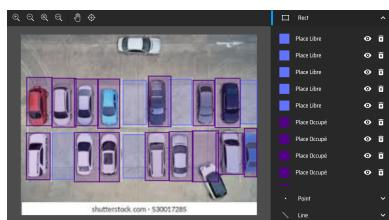


FIGURE 2 – Image 1

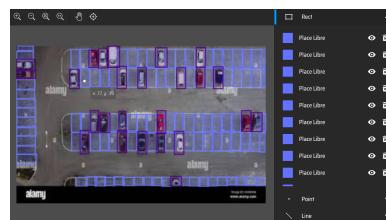


FIGURE 3 – Image 2

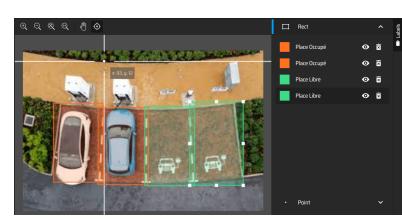


FIGURE 4 – Image 3

FIGURE 5 – Trois exemples d'images annotées de la base de données

## 4 Base de données (Modification)

Dans le but d'améliorer les performances de notre modèle, nous avons entrepris de réaliser de l'augmentation de données. Cependant, nous nous sommes rapidement rendu compte que notre jeu de test était trop limité pour évaluer de manière fiable l'impact de ces augmentations. Afin de pallier ce manque, nous avons donc enrichi notre base de données de test avec de nouvelles images, plus variées. Nous avons notamment modifié l'une de nos images existantes en diminuant la luminosité afin de simuler une condition de prise de vue nocturne. De plus, nous avons ajouté une image représentant un parking dans lequel toutes les voitures sont garées horizontalement, afin d'évaluer la capacité du modèle à s'adapter à une configuration bien spécifique. Ces ajouts visaient à mieux tester la généralisation du modèle face à des cas qu'il rencontre moins fréquemment.

## 5 Elaboration d'une solution

Pour notre problème, nous avons fait le choix d'utiliser le modèle pré-entraîné **YOLOv8**, un modèle publié en 2023 et particulièrement adapté à la détection d'objet.

## 5.1 Entrainement classique du modèle

Le modèle utilisé est un **YOLOv8 nano** (yolov8n.pt) pré-entraîné, entraîné ensuite sur notre dataset pendant **50 epochs**.

L'entraînement a été réalisé avec une taille d'image fixe de **640x640 pixels** et un **batch size de 16**.

L'optimiseur utilisé est **AdamW**, réglé automatiquement par la librairie Ultralytics.

### 5.1.1 Analyse de la courbe de perte d'entraînement

Cette première courbe représente la perte totale calculée pendant l'entraînement, qui regroupe les différentes composantes de la perte :

- **box\_loss** : la perte de localisation des boîtes
- **cls\_loss** : la perte de classification
- **dfl\_loss** : la Distribution Focal Loss.

Cette courbe est tracée en fonction du nombre d'epochs.

La perte totale est une mesure de l'erreur entre les prédictions du modèle et les véritables annotations. Elle est censée diminuer au fur et à mesure que le modèle s'améliore et apprend à détecter plus précisément les objets dans les images.



FIGURE 6 – Training Losses

On peut observer une diminution générale des trois courbes qui montrent une tendance à la baisse au fil des époques, ce qui indique que le modèle apprend correctement et parvient à minimiser son erreur.

La **Class Loss** initialement très élevée, diminue fortement et rapidement jusqu'à converger vers des valeurs proches de 1 à partir de l'époque 20. Cela indique que le modèle a, au début, beaucoup de mal à distinguer les 2 classes correctement mais il apprend rapidement à mieux classer les objets présents dans les images.

On observe cependant une stagnation de la courbe à partir de l'époque 35 environ.

La **Box Loss** diminue plus lentement que la Class Loss. Elle semble d'ailleurs décroître de plus en plus doucement à partir de l'époque 15. Cela peut indiquer que la précision des détections a plafonné, ou que le modèle atteint une limite de performance sur les coordonnées des boîtes. La détection des box (objets)

est donc très bonne dès le début mais très vite elle atteint son maximum.

La **DFL Loss** baisse similairement à la Box Loss et se stabilise rapidement vers 1.0 dès l'époque 15. Cette stabilité montre que le modèle devient progressivement plus précis dans ses localisations fines, sans grand ajustement.

En résumé de cette courbe, on voit qu'à partir de l'epoch 30, on remarque que les trois pertes (box, class, DFL) cessent de diminuer significativement et stagnent autour d'une valeur constante.

Or, il n'y a pas de remontée brutale des pertes, ce qui est typique d'un surapprentissage très prononcé. On pourrait donc en conclure que le modèle a atteint sa capacité maximale à apprendre des données d'entraînement, et qu'il pourrait commencer à surapprendre si on continuait davantage l'entraînement.

## 5.2 Amélioration de l'entraînement du modèle

Afin d'améliorer la robustesse de notre modèle face à différentes conditions de prise de vue (variation de luminosité, d'angle de prise de vue, etc.), nous avons introduit plusieurs techniques d'**augmentation de données** lors de la phase d'entraînement. Ces transformations permettent de générer de nouvelles images synthétiques à partir des données existantes, augmentant ainsi la diversité du jeu de données et réduisant le risque de surapprentissage.

### 5.2.1 Changement d'échelle (scale)

Nous avons remarqué que notre modèle avait du mal à identifier les places occupées sur des images de petites tailles. Nous avons donc voulu améliorer cela en augmentant le nombre d'images de petites tailles dans notre base de données. Voici donc les paramètres que nous avons ajoutés :

- **Hyperparamètre utilisé :** scale = 0.75
- **Effet :** Zoom ou dézoom l'image de manière aléatoire en multipliant la taille de l'image par un coefficient égale à 1+scale ou 1-scale
- **Commande d'entraînement :**

```
model.train(data="data/dataset.yaml", epochs=50, imgsz=640, batch=16, scale=0.75)
```

### 5.2.2 Augmentation de la luminosité (hsv\_v)

Nous avions trouvé très peu d'images de nuit pour notre base de données nous avons donc modifié la luminosité des images en ajustant la composante V (Value) dans l'espace HSV. Cela permet de simuler des conditions d'éclairage variables, notamment des scènes nocturnes ou faiblement éclairées.

- **Hyperparamètre utilisé :** hsv\_v = 0,5
- **Effet :** assombrit ou éclaircit les images aléatoirement, avec une variation maximale de 50.
- **Commande d'entraînement :**

```
model.train(data="data/dataset.yaml", epochs=50, imgsz=640, batch=16, hsv_v=0.5)
```

### 5.2.3 Rotation des images (degrees)

Pour améliorer l'invariance du modèle aux orientations des voitures (nous avions beaucoup d'images avec des places en verticale), nous avons appliqué des rotations aléatoires sur les images d'entraînement. Cela permet de mieux gérer les cas où les véhicules sont garés sous des angles atypiques.

- **Hyperparamètre utilisé :** degrees = 90
- **Effet :** rotation aléatoire des images jusqu'à ±90 degrés.
- **Commande d'entraînement :**

```
model.train(data="data/dataset.yaml", epochs=50, imgsz=640, batch=16, degrees=90)
```

#### 5.2.4 Augmentation combinée

Enfin, une configuration plus avancée a été testée, combinant plusieurs types d'augmentation pour enrichir davantage les variations du dataset.

- Hyperparamètres utilisés :

- `hsv_h = 0.03` : variation de la teinte (Hue)
- `hsv_s = 0.6` : variation de la saturation
- `hsv_v = 0.5` : variation de la luminosité
- `translate = 0.25` : translation aléatoire des images jusqu'à 25
- `scale = 0.5` : zoom aléatoire (0.5 à 1.5x)
- `bgr = 1` : inversion aléatoire des canaux de couleur
- `erasing = 0.4` : suppression aléatoire de zones (Random Erasing)

- Commande d'entraînement :

```
model.train(data="data/dataset.yaml", epochs=50, imgsz=640, batch=16,
hsv_h=0.03, hsv_s=0.6, hsv_v=0.5,
translate=0.25, scale=0.5,
bgr=1, erasing=0.4)
```

Cette configuration permet d'introduire une grande variété d'images, simulant par exemple des changements de lumière, de couleur, de position ou encore des obstructions partielles sur les voitures. Cela renforce la capacité du modèle à généraliser à des situations complexes et plus proches du réel.

## 6 Analyse des résultats

Nous avons testé les performances de notre modèle avec les images tests.

Dans un premier temps nous nous sommes limités aux images de notre base de données. Puis, dans l'objectif d'améliorer nos résultats, nous avons utilisé de l'augmentation de données.

### 6.1 Sans augmentation de données

#### 6.1.1 Résultats et hypothèses

Même sans augmentation des données, les résultats obtenus se sont révélés très satisfaisants.

En effet, on observe un MAP@50 de 0.826, ce qui, d'après nos recherches, correspond à un modèle de très bonne qualité. De plus, en fixant un seuil de confiance bas, par exemple à 0.1, on constate que le rappel dépasse 0.8 et que la précision est d'environ 0.7, ce qui témoigne d'excellentes performances.

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95) :
all	6	558	0.741	0.805	0.826	0.384
Place_Libre	6	374	0.701	0.87	0.846	0.42
Place_Occupée	6	184	0.782	0.741	0.806	0.348

FIGURE 7 – Résultats sans augmentation de données

Nous avons toutefois identifié un défaut. Lorsqu'on augmente le seuil de confiance du modèle, on observe rapidement une divergence entre la détection des places occupées et celle des places libres. Cela apparaît d'abord sur la courbe du score F1 : la courbe associée aux places libres présente une forme typique d'un modèle de détection d'objets, tandis que celle des places occupées chute de manière trop abrupte.

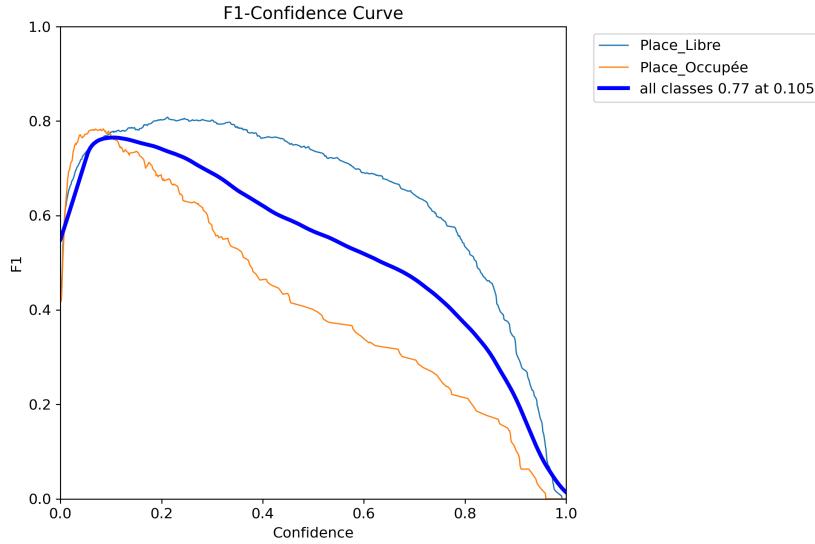


FIGURE 8 – Courbe du score F1

Le problème est particulièrement visible sur la courbe de rappel. On remarque que le rappel pour les places occupées diminue rapidement, même pour de faibles seuils de confiance, ce qui signifie que le modèle manque de nombreuses places occupées malgré une tolérance élevée à l'incertitude. Par ailleurs, la matrice de confusion révèle que le modèle prédit fréquemment (dans 43% des cas) la classe "background" au lieu de détecter une place occupée.

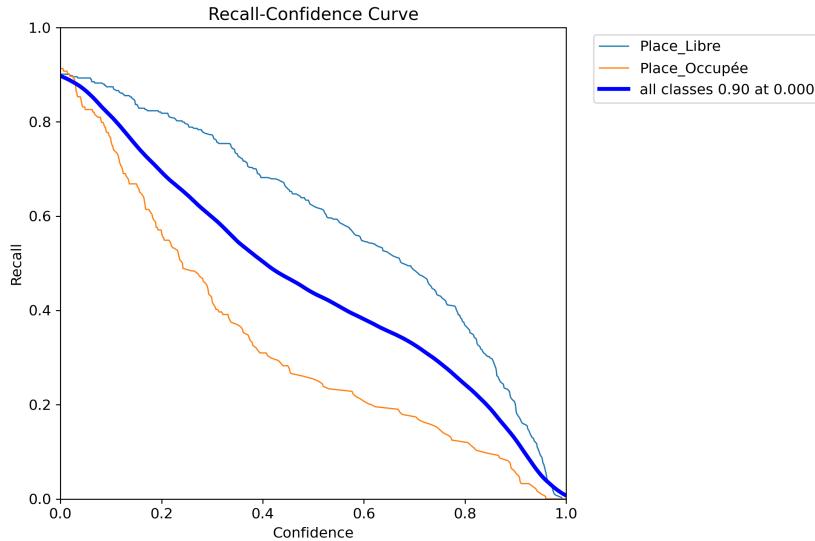


FIGURE 9 – Courbe du rappel

En conclusion, le modèle reste insuffisant pour détecter correctement les places occupées.

Selon nous, cette faiblesse peut s'expliquer par des ambiguïtés visuelles : une place occupée peut parfois être difficile à distinguer de son environnement (ombres, objets, marquages au sol, etc.), ce qui tend à faire baisser le rappel pour cette classe. De plus, la grande diversité des véhicules (couleur, forme, taille, etc.) complique la tâche du modèle. Pour améliorer les performances, il serait donc nécessaire d'enrichir significativement la base de données avec des images variées de voitures.

À l'inverse, on observe que la précision pour les places libres, bien que déjà élevée, reste inférieure à celle des places occupées. Cet écart peut, selon nous, s'expliquer par une tendance du modèle à détecter à

tort une place libre là où il n'y en a pas. En effet, les seules indications visuelles de l'existence d'une place libre sont souvent les lignes de délimitation au sol, ce qui peut induire le modèle en erreur et diminuer sa précision.

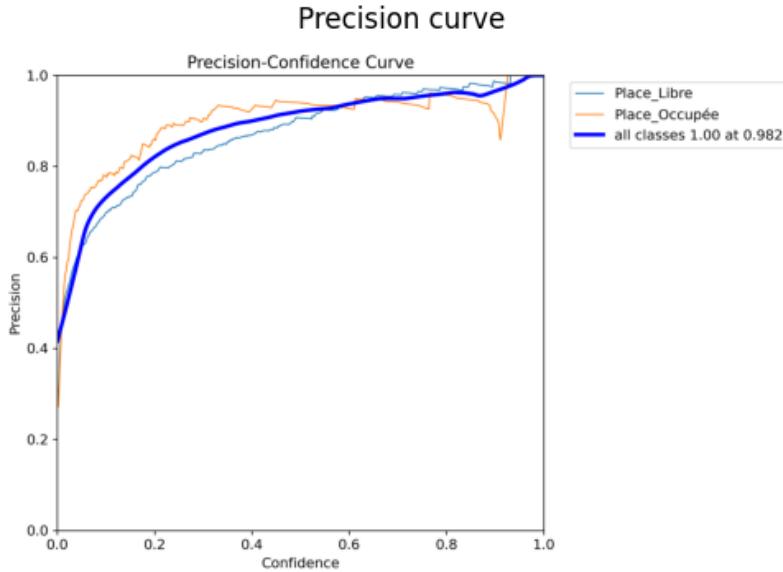


FIGURE 10 – Courbe de précision

### 6.1.2 Observations

```
image 1/1 /content/Projet-parking/data/test/images/39.jpg: 320x640 6 Place_Occupées, 51.8ms
Speed: 2.3ms preprocess, 51.8ms inference, 1.7ms postprocess per image at shape (1, 3, 320, 640)
```

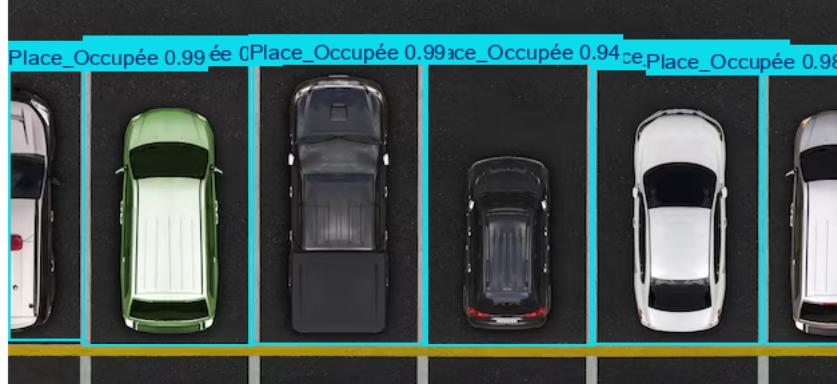


FIGURE 11 – Résultats sans augmentation de données

On observe que, sur les images fortement zoomées et centrées uniquement sur les places de parking, les résultats sont particulièrement satisfaisants. Dans ces cas-là, toutes les places ont été correctement détectées et étiquetées avec le bon label, qu'elles soient libres ou occupées. L'absence de distractions visuelles ou d'éléments d'arrière-plan permet au modèle de se concentrer uniquement sur les zones pertinentes, ce qui facilite considérablement la tâche de classification.



FIGURE 12 – Objets du background détectés

En revanche, lorsque l'image est plus dézoomée et contient de nombreux éléments visuels autour des places de parking — notamment dans le cas des places en épi — la détection devient nettement moins efficace.

En effet, sur les images plus dézoomées, les voitures apparaissent plus petites. Elles occupent donc moins de pixels et contiennent moins d'informations visuelles, ce qui rend leur identification plus difficile pour le modèle.

De plus, la présence d'un environnement complexe, avec des voitures en arrière-plan, des bâtiments, ou encore des marquages au sol variés, peut perturber le modèle et rendre plus difficile l'identification précise des places et de leur statut (libre ou occupé).

Afin d'améliorer la détection des places occupées, nous avons tenté d'utiliser l'augmentation de données pour pallier ce défaut.

## 6.2 Augmentation de données avec zoom et dézoom



FIGURE 14 – Avant l'augmentation de données



FIGURE 13 – Voitures de petites tailles non détectées



FIGURE 15 – Après l'augmentation de données

Dans un premier temps, l'observation des images de test suggère que l'augmentation de données n'a pas eu l'effet escompté. En effet, les places occupées qui n'étaient pas détectées par le premier modèle ne le sont toujours pas. Pire encore, certaines places occupées auparavant correctement détectées ne le sont plus.

Cependant, il semblerait que le modèle commette moins d'erreurs. L'augmentation de données aurait peut-être permis d'améliorer sa précision.

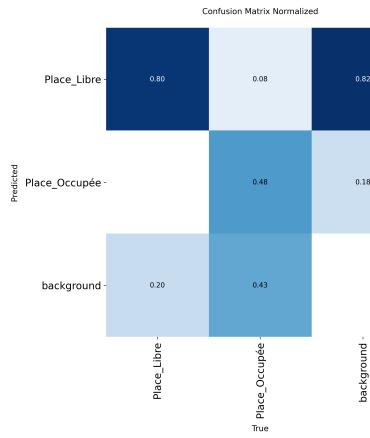


FIGURE 16 – Matrice de confusion - sans augmentation de données

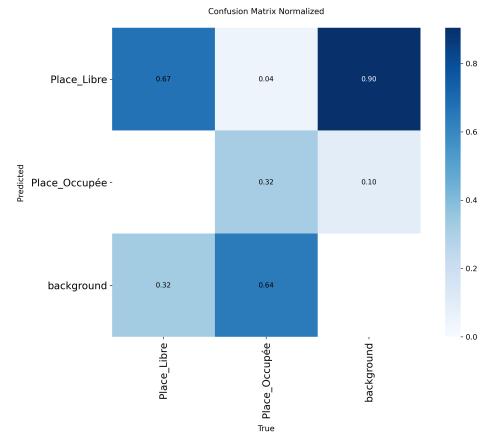


FIGURE 17 – Matrice de confusion - avec augmentation de données

Les deux matrices de confusion présentées ci-dessus confirment nos observations : notre modèle détecte encore moins bien les places, qu'elles soient libres ou occupées.

Sans augmentation de données, il détectait 80% des places libres, contre seulement 67% après augmentation. Pour les places occupées, le taux de détection passe de 48% à 32%. Le rappel global du modèle s'en trouve donc diminué.

La comparaison des deux courbes de rappel présentées ci-dessous confirme cette dégradation.

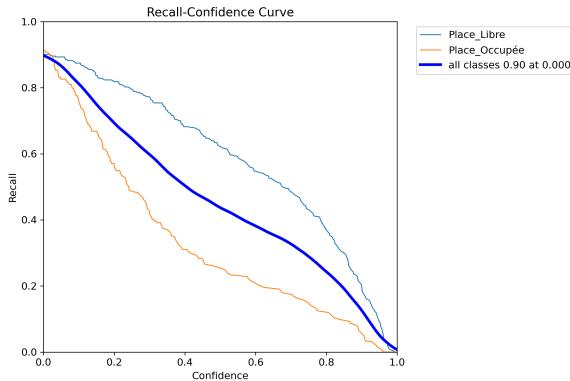


FIGURE 18 – Courbe de rappel - Sans augmentation de données

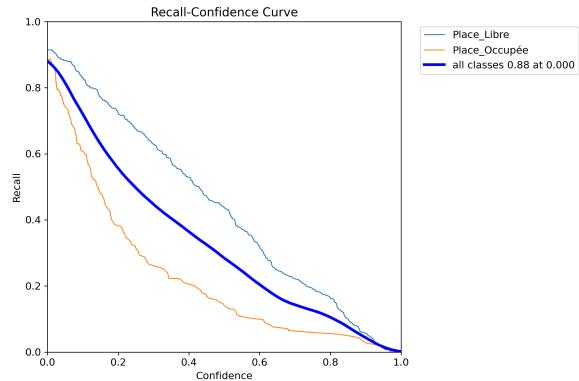


FIGURE 19 – Courbe de rappel - Avec augmentation de données

En revanche, pour ce qui est de la précision, nos premières observations étaient également justes. Les deux courbes ci-dessous nous montrent bien que notre modèle est devenu plus précis.

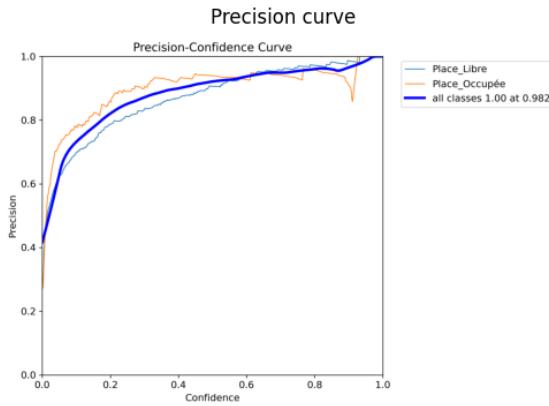


FIGURE 20 – Courbe de précision - Sans augmentation de données

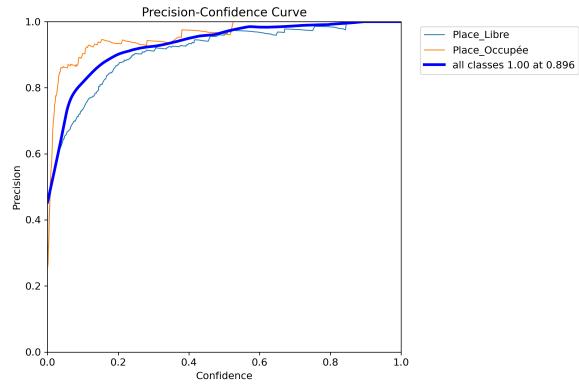


FIGURE 21 – Courbe de précision - Avec augmentation de données

Selon nous, ce résultat est cohérent. En effet, grâce à l'augmentation de données, le modèle dispose de davantage de critères pour reconnaître les places occupées ou libres. Ainsi, lorsqu'il identifie une place, il le fait avec davantage de confiance, ce qui peut expliquer l'amélioration de la précision.

En revanche, notre base de données reste de taille réduite. Le modèle apprend donc des critères spécifiques, mais en quantité limitée. Par conséquent, il rejette rapidement certaines places qu'il aurait dû considérer comme occupées ou libres, ce qui entraîne une baisse du rappel.

### 6.3 Résultats en augmentant les données avec des images "de nuit"

Dans un second temps, nous avons utilisé l'augmentation de données pour élargir les capacités de notre modèle. Plus précisément, nous avons ajouté des images prises de nuit générées grâce à YOLO, ce qui a permis d'enrichir notre jeu de données avec des conditions d'éclairage variées. Suite à ce second test, les résultats se sont effectivement améliorés : le mAP@50 est passé à 0,833, le rappel à 0,825, et la précision à 0,76. Cette progression s'explique par le fait que le modèle, entraîné avec un ensemble de données plus diversifié, devient plus robuste et moins enclin à se tromper face à des situations visuelles complexes ou inattendues. En d'autres termes, l'ajout de ces nouvelles images aide le modèle à mieux généraliser et à éviter certains pièges liés à la variabilité des environnements.

Class	Images	Instances	Box (P)	R	mAP50	mAP50-95 :
all	6	558	0.76	0.825	0.833	0.389
Place_Libre	6	374	0.663	0.898	0.863	0.422
Place_Occupée	6	184	0.857	0.751	0.802	0.357

FIGURE 22 – Résultats avec les images de nuit

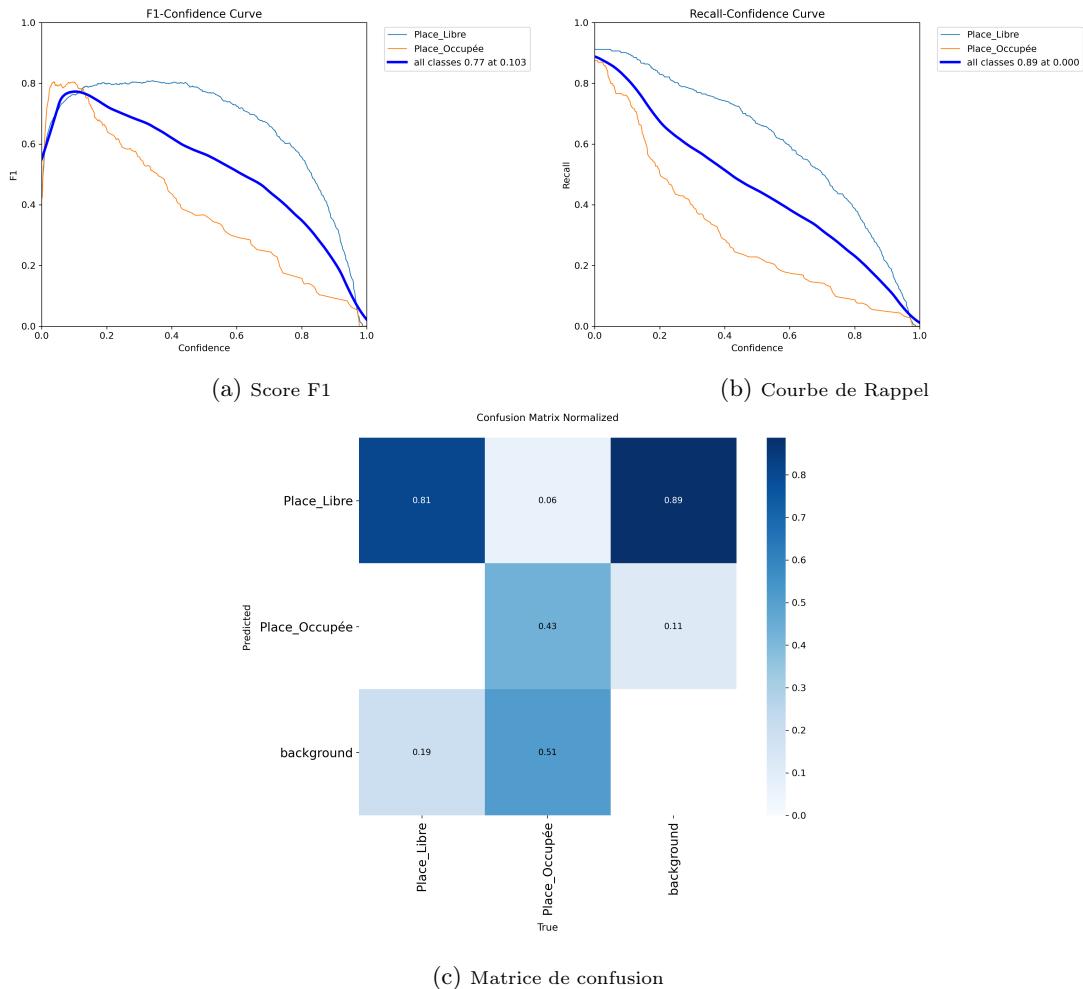


FIGURE 23 – Courbes résultat 2

Nous observons que les courbes sont sensiblement similaires à celles du test sans augmentation de données.

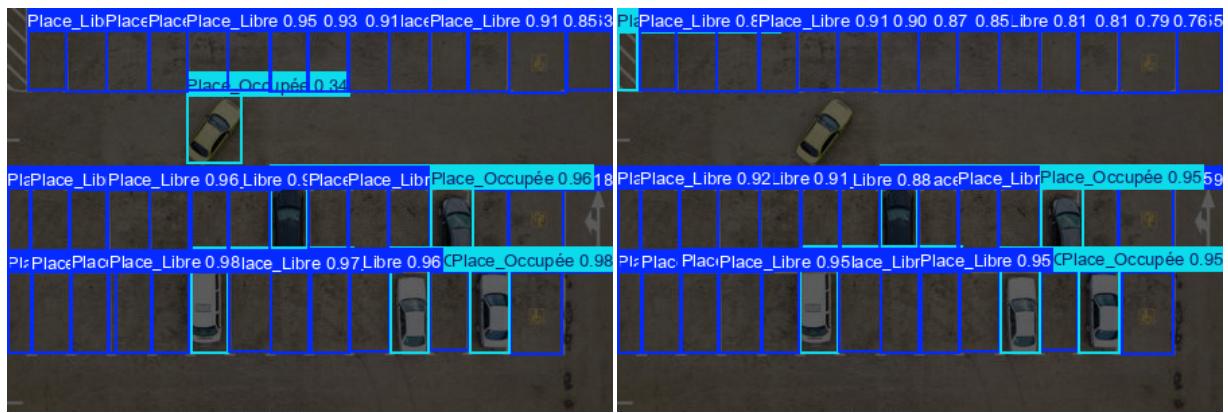


FIGURE 24 – Résultat avec augmentation de données

FIGURE 25 – Résultat sans augmentation de données

On observe désormais que les lignes situées en haut à gauche ne sont plus erronément détectées comme des places occupées, ce qui montre une meilleure compréhension du modèle. En revanche, la voiture en train de quitter sa place est maintenant identifiée comme une place occupée. De plus, on constate une augmentation du taux de confiance associé aux places détectées, ce qui traduit une plus grande fiabilité dans les prédictions du modèle.

## 6.4 Résultats en augmentant les données avec des images tournées

Nous avons ensuite testé une nouvelle forme d'augmentation de données en ajoutant des images pivotées à 90 degrés, dans le but d'aider le modèle à mieux reconnaître les places de stationnement, qu'elles soient orientées horizontalement ou verticalement. Cependant, les résultats obtenus ne correspondaient pas à nos attentes. Contrairement à ce que nous espérions, les performances ont diminué : le mAP@50, le rappel et la précision ont tous baissé de manière inattendue.

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95 :
all	6	558	0.712	0.702	0.704	0.317
Place_Libre	6	374	0.646	0.676	0.638	0.303
Place_Occupée	6	184	0.777	0.728	0.77	0.331

FIGURE 26 – Résultats avec les images tournées

Cette dégradation pourrait s'expliquer par plusieurs facteurs. Tout d'abord, la rotation des images modifie l'orientation naturelle des repères visuels que le modèle avait appris à reconnaître, comme les lignes de stationnement, les bordures de trottoir ou encore l'alignement des véhicules. Ces éléments, lorsqu'ils sont présentés sous un angle inhabituel, peuvent être interprétés différemment, voire mal interprétés, par le modèle. Par ailleurs, l'ajout d'images trop différentes de celles du jeu d'entraînement initial peut introduire une variabilité excessive, qui perturbe l'apprentissage au lieu de l'améliorer. Si cette nouvelle diversité n'est pas représentative de cas fréquents, elle risque d'agir comme du bruit, et donc de dégrader la performance globale du modèle en diluant les motifs visuels appris jusque-là.



FIGURE 27 – Résultat avec augmentation de données



FIGURE 28 – Résultat sans augmentation de données

De plus, on constate une nette dégradation de la détection sur certaines images. Par exemple, des images qui comportaient à la fois des places horizontales et verticales — et sur lesquelles les deux types de places étaient correctement détectés auparavant — ne le sont plus de manière satisfaisante. En particulier, les places verticales ne sont désormais plus détectées du tout, ce qui souligne un effet négatif de l'augmentation par rotation sur la capacité du modèle à généraliser correctement dans ces cas-là.

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95 :
all	6	558	0.775	0.82	0.832	0.387
Place_Libre	6	374	0.723	0.885	0.864	0.415
Place_Occupée	6	184	0.828	0.755	0.799	0.359

FIGURE 31 – Résultats avec les images augmentées de manière aléatoire



FIGURE 29 – Résultat avec augmentation de données



FIGURE 30 – Résultat sans augmentation de données

On constate que même sur les images contenant uniquement des places de stationnement horizontales — qui étaient auparavant bien détectées — les performances du modèle se sont dégradées. Certaines places ne sont plus reconnues, ou bien elles le sont avec un taux de confiance beaucoup plus faible. Cela suggère que l'ajout d'images pivotées a non seulement perturbé la détection des places verticales, mais a également affecté négativement la capacité du modèle à détecter correctement des configurations pourtant similaires à celles déjà présentes dans le jeu d'origine. Cette baisse de performance généralisée montre que l'augmentation par rotation n'a pas permis de renforcer la robustesse du modèle, mais a au contraire introduit une confusion dans sa compréhension des orientations des places.

## 6.5 Résultats en augmentant les données avec des images modifiées aléatoirement

Pour cette dernière phase d'expérimentation, nous avons tenté de modifier plusieurs paramètres aléatoires lors de l'augmentation de données, dans l'objectif d'observer si une simple augmentation de la quantité de données — même peu cohérentes — pouvait suffire à améliorer les performances du modèle. Concrètement, nous avons appliqué diverses transformations sur les images : modification de la couleur, de la saturation, de la luminosité, changement de la position dans le cadre (translations), zoom aléatoire, inversion des canaux de couleur pour passer en BGR, ainsi que le masquage partiel de certaines zones de l'image. Ces manipulations avaient pour but de tester la tolérance du modèle à des perturbations visuelles importantes et de voir s'il pouvait malgré tout apprendre à détecter correctement les places de stationnement dans des conditions très variées.

Les résultats obtenus avec cette augmentation de données aléatoires se sont révélés légèrement meilleurs que ceux des tests précédents. Bien que les transformations appliquées aient été parfois peu cohérentes ou éloignées des situations réelles, elles ont permis de rendre le modèle plus robuste face à la variabilité des images. En exposant le modèle à un large éventail de modifications visuelles, même artificielles, on l'aide à moins dépendre de certains repères visuels spécifiques et à mieux généraliser. Cela peut expliquer l'amélioration modérée des performances : le modèle devient plus tolérant aux variations de luminosité, de couleur ou de cadrage, ce qui le rend plus efficace face à des images nouvelles ou légèrement dégradées. Ainsi, même si l'augmentation n'était pas parfaitement réaliste, elle a contribué à une meilleure capacité d'adaptation du modèle.

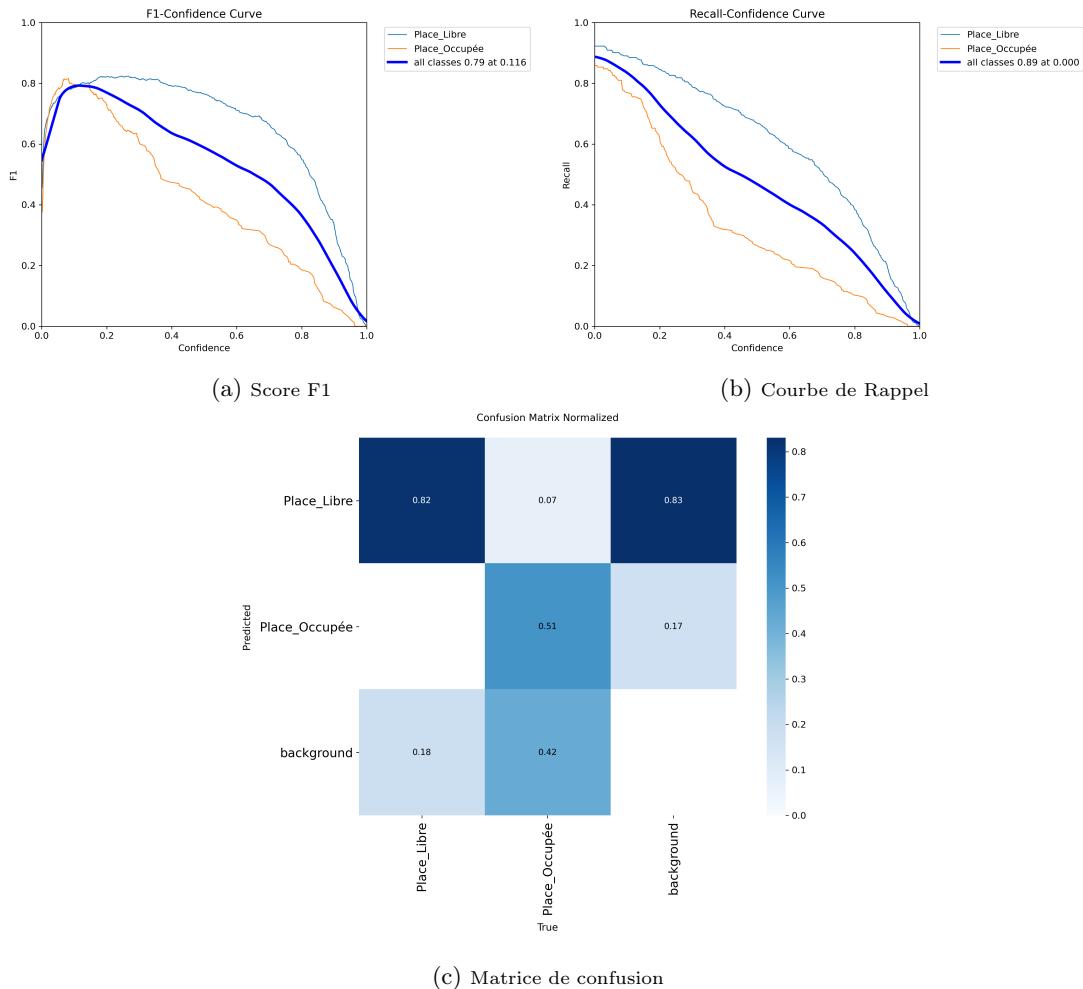


FIGURE 32 – Courbes résultat 4

Nous remarquons à nouveau que l'augmentation de données n'a qu'une faible influence sur les courbes et qu'elles gardent la même allure.



FIGURE 33 – Résultat avec augmentation de données



FIGURE 34 – Résultat sans augmentation de données

Sur les deux images présentées ici, on observe clairement une amélioration : un plus grand nombre de places, qu'elles soient libres ou occupées, sont correctement détectées, et certaines erreurs présentes auparavant ont disparu. Ces résultats confirment que l'augmentation de données aléatoire a eu un effet bénéfique dans certains cas. Cependant, cette amélioration n'est pas systématique. Sur d'autres images,

les changements sont plus subtils : le modèle commet encore des erreurs, parfois simplement différentes, ou en quantité légèrement réduite. Ces ajustements restent trop mineurs pour que l'on puisse considérer les résultats comme réellement satisfaisants de manière générale. Cela montre que, bien que prometteuse, cette stratégie d'augmentation ne suffit pas à elle seule pour obtenir des performances solides et homogènes sur l'ensemble du jeu de données.

## 7 Conclusion

En conclusion, on peut considérer que notre modèle fournit des résultats globalement satisfaisants, en particulier compte tenu des contraintes sur les données disponibles. Il est probable que de meilleures performances auraient pu être atteintes avec une base de données plus étendue et plus diversifiée. Cependant, la collecte d'images de parkings vus du dessus reste difficile, notamment en raison du manque de ressources gratuites accessibles. C'est pourquoi nous avons principalement compté sur l'augmentation de données via YOLO pour enrichir notre jeu d'apprentissage. Par ailleurs, nos observations mettent en évidence une tendance du modèle à confondre plus facilement une place libre (en la détectant alors qu'elle n'existe pas) qu'une place occupée, qui est parfois oubliée. Cela souligne une sensibilité particulière du modèle aux repères visuels associés aux places vides, souvent plus ambigus dans certaines configurations, et sa difficulté à identifier clairement une voiture.

Par ailleurs, il aurait été intéressant de tester plus systématiquement les limites de notre modèle. Nous avons commencé à explorer cette piste en ajoutant une image prise de nuit, mais d'autres conditions extrêmes auraient pu être envisagées : des images avec de la neige, de la pluie, du brouillard, une orientation plus horizontale, ou encore des scènes composées uniquement de motos, afin d'évaluer la capacité du modèle à s'adapter à différents types de véhicules.

Nous aurions également pu inclure des images de parkings comportant une route, pour observer la réaction du modèle face à des véhicules en mouvement.

Malgré cela, les performances obtenues restent encourageantes pour une application pratique de détection de stationnement.

## 8 Bibliographie

- Ultralytics Performance Metrics Deep Dive : <https://docs.ultralytics.com/fr/guides/yolo-performance-metrics/>
- Ultralytics Data Augmentation : <https://docs.ultralytics.com/fr/guides/yolo-data-augmentation/>