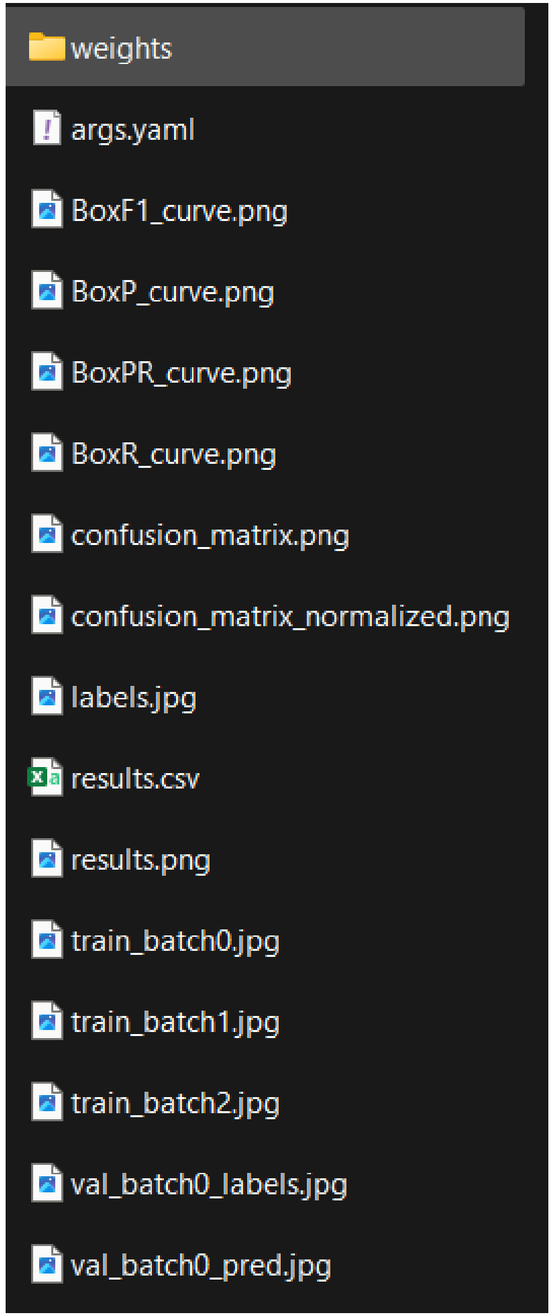
RÉSULTATS ET ÉVALUATION DU MODÈLE

DOSSIER DE RÉSULTATS

Une fois l'entrainement terminé, **vous pourrez retrouver vos résultats dans le dossier "runs/detect/train"** (à part si vous aviez configuré un autre nom pour votre dossier de résultats).

Dans ce dossier on retrouve toutes les choses ci-contre (peut varier) :

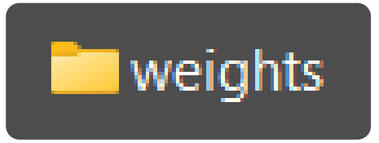
On va voir ce qu'est chaque élément un a un.



ISDM
INSTITUT DE SCIENCE DES DONNÉES
MONTPELLIER

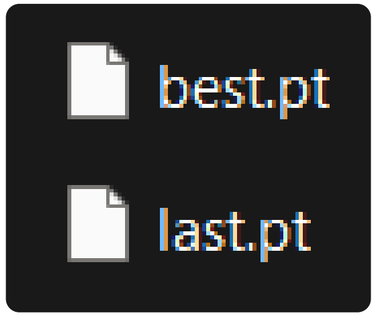
DOSSIER DE RÉSULTATS

Weights



Le dossier “Weights” correspond au **fruit de votre labeur** car dedans il y a les poids attribués aux neurones lors de l’entraînement. **Ce sont les neurones de votre modèle dans un fichier**.

Il y a deux fichiers dedans : “best.pt” et “last.pt”. Cela se comprend de la manière suivante : les poids des neurones changent de manière régulière lors de l’entrainement et la qualité des poids est évaluée grâce à une fonction spécifique.



Donc **last.pt correspond aux derniers poids en date et best.pt correspond aux meilleurs poids qu’il y a eu de tout l’entrainement**. Ils ne sont pas forcément similaires mais ça peut arriver et s’il sont similaires, c’est probablement un signe que votre modèle devrait s’entrainer plus longtemps.

DOSSIER DE RÉSULTATS

args.yaml

args.yaml

args.yaml contient **tous les paramètres** qu'a utilisé YOLO pour son entrainement incluant tous ceux qui avait des valeurs par défault que vous n'avez pas modifié et tous ceux sans argument (donc à 0).

Par exemple, dans la portion de ficher args.yaml ci-dessous, on peut lire que l'entrainement a été lancé avec les valeurs des paramètres suivants :

```
epochs: 1
time: null
patience: 100
batch: 16
imgsz: 640
```

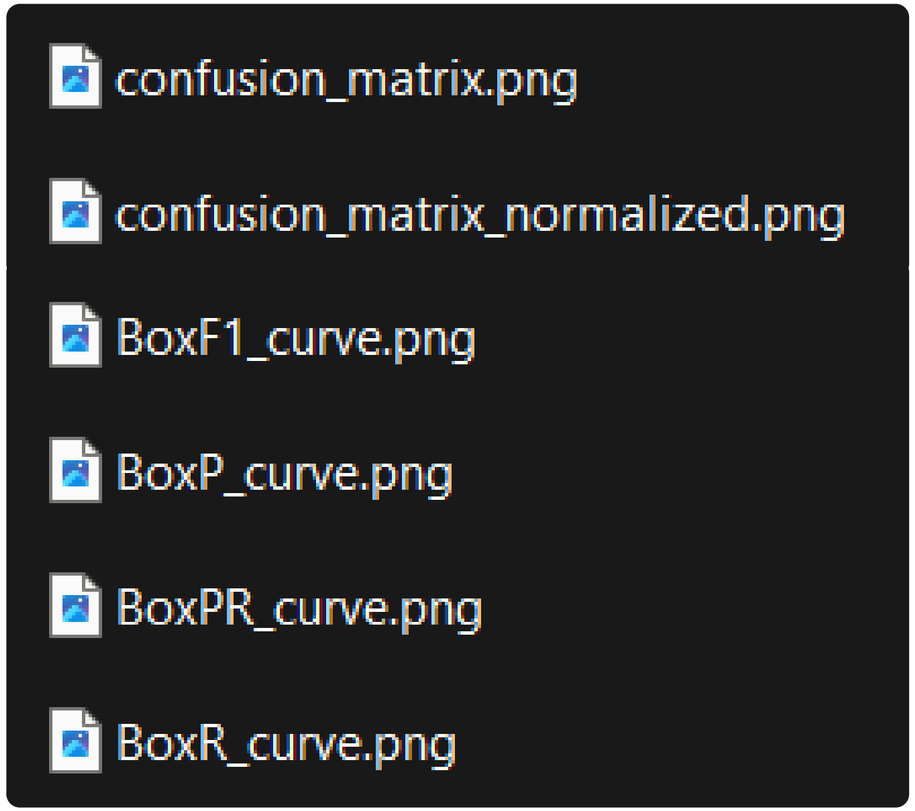
- 1 époque
- le temps limite n'avait pas été paramétré donc est "null"
- la patience est de 100 (valeur par défaut)
- la taille du batch est de 16 (valeur par défaut)
- la taille des images est de 640 pixels de coté (valeur par défaut)

DOSSIER DE RÉSULTATS

Matrices de confusion et courbes associées

En vision par ordinateur, on utilise beaucoup l'outil "**matrice de confusion**" et on peut en déduire beaucoup de statistiques pertinentes pour évaluer un modèle.

On va devoir passer par une partie théorique avant de pouvoir comprendre ce que l'on voit, donc il va falloir s'accrocher un peu !



ISDM
INSTITUT DE SCIENCE DES DONNÉES
MONTPELLIER

MATRICE DE CONFUSION

Approche théorique

Une matrice de confusion est un **tableau** qui va synthétiser et trier les prédictions du modèle pour pouvoir les analyser.

Pour notre cas pratique nous prendrons les classes suivantes :

- “**positive**” qui correspond à ce que l’on souhaite détecter (on choisira la classe “Penny”, une pièce anglaise, pour exemple)
- “**negative**” qui correspond à l’absence de la classe “positive”, donc pas de pièce “Penny”

MATRICE DE CONFUSION

Approche théorique

Les lignes correspondent à ce que le modèle a prédit.

Les colonnes correspondent à la réalité terrain.

Classe prédite		
Positive	**w**	x
Negative	y	z
	Positive	Negative
	Vraie classe	

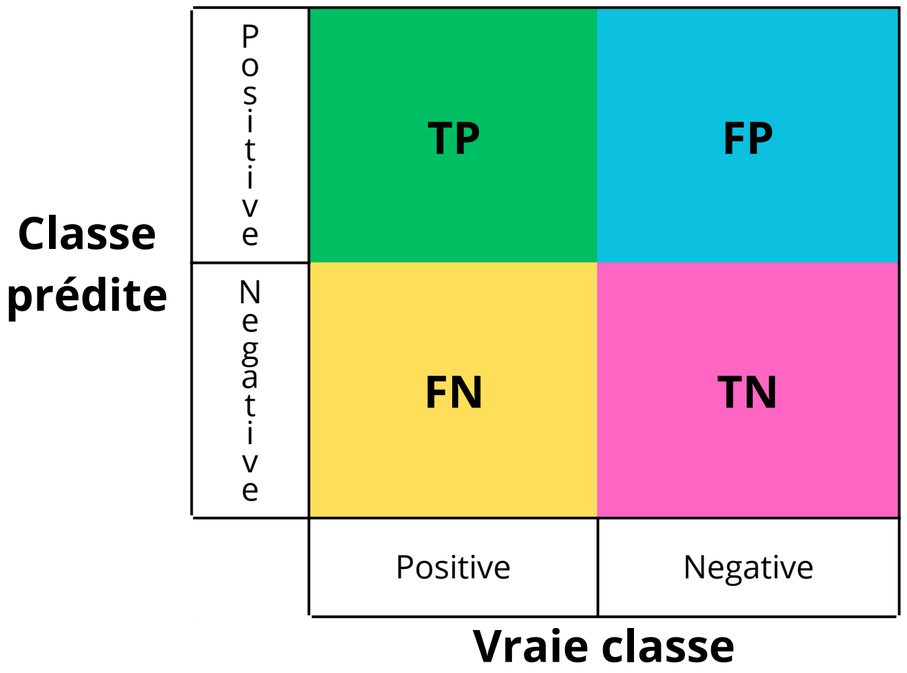
Donc, pour lire une matrice de confusion, on lit l'intersection d'une ligne et d'une colonne :

- w images prédites positives (contenait un "penny") et était bien positive (contenait bien un "penny")
- et ainsi de suite ....

MATRICE DE CONFUSION

Approche théorique

Pour comprendre les valeurs plus facilement, des noms ont été donné aux quatre cases.



TP : True Positive (vrai positif)

Ce qui était à détecter et qui a bien été détecté.
Ex : il y avait un “penny” et il a bien été détecté.

TN : True Negative (vrai négatif)

Ce qui n’était pas à détecter et qui n’a pas été détecté.
Ex : il n’y avait pas de “penny” et pas de “penny” n’a été détecté.

FP : False Positive (faux positif)

Ce qui devrait être détecté comme négatif mais a été détecté comme positif.
Ex : il n’y avait pas de “penny” mais un “penny” à été détecté.

FN : False Negative (faux négatif)

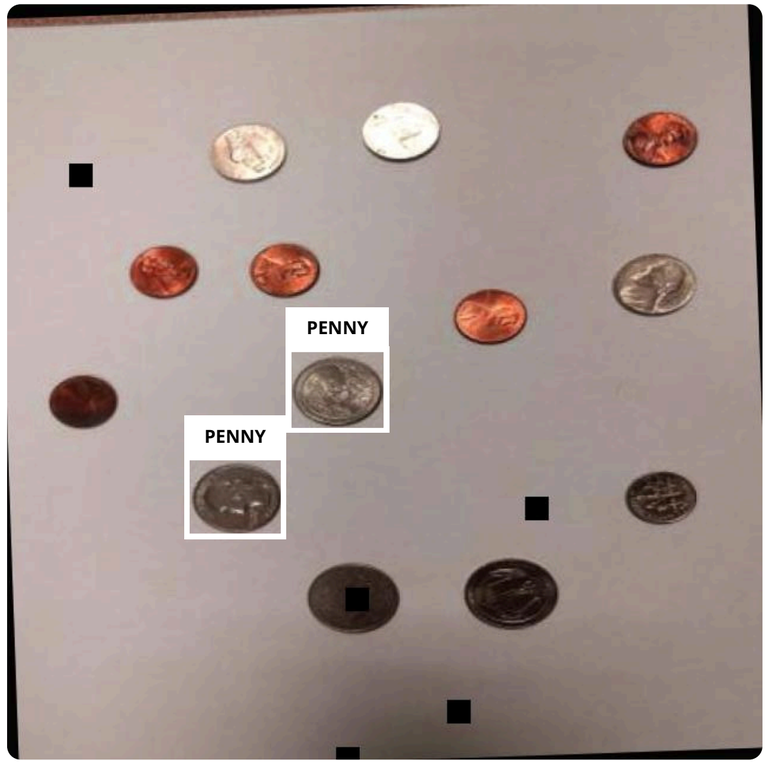
Ce qui devait être détecté comme positif mais a été détecté comme négatif.
Ex : il y avait un “penny” mais il n’a pas été détecté.

(il y a un exemple imagé sur la page d’après)

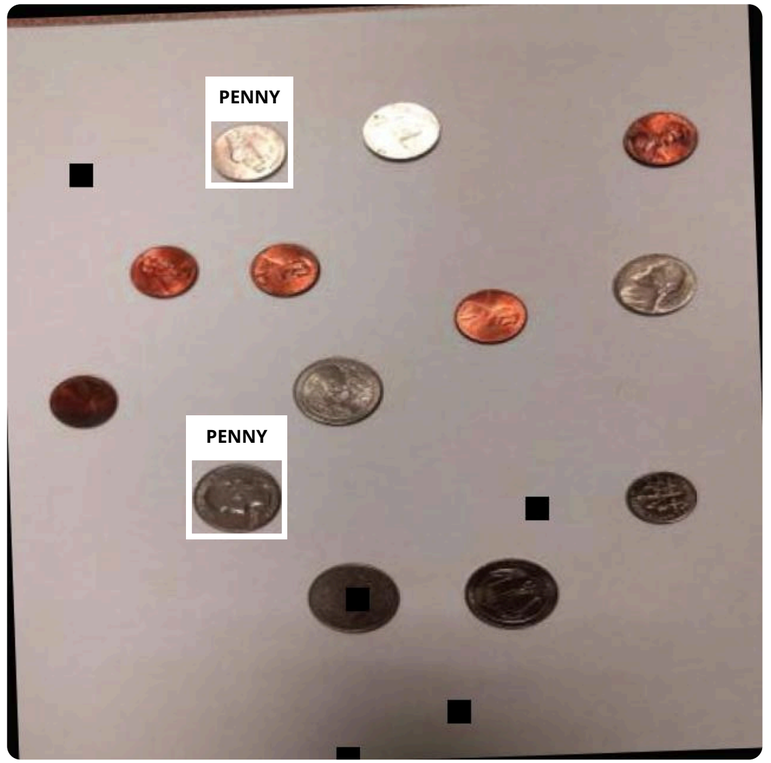
MATRICE DE CONFUSION

Approche théorique

Cette image correspond à la vérité terrain, c’est-à-dire aux annotations faites à la main donc ce que le modèle est censé trouver.



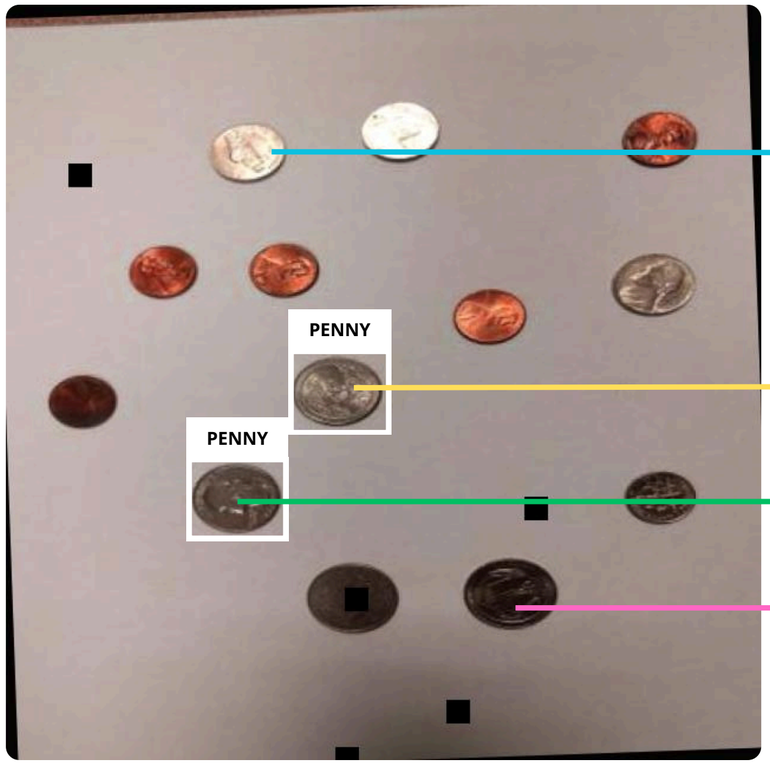
Cette image correspond aux détections faites par le modèle sur la même image après l’entrainement



MATRICE DE CONFUSION

Approche théorique

Si on fait le jeu des 7 erreurs entre les deux images :

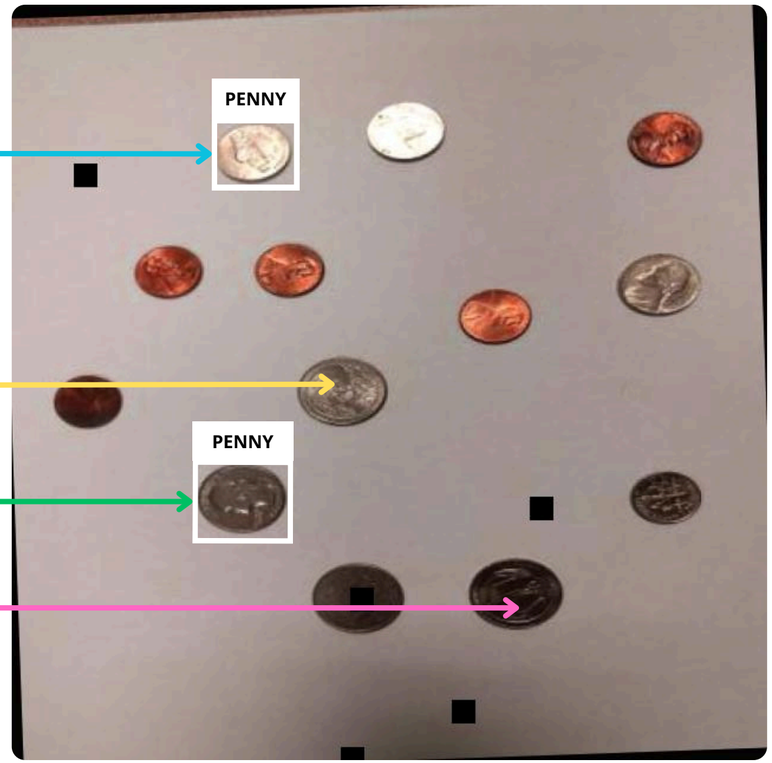


n'était pas un penny mais a été pris pour un penny
(faux positif)

n'a pas été détecté comme penny
(faux négatif)

a bien été détecté comme un penny
(vrai positif)

n'était pas un penny et n'a pas été détecté comme penny
(vrai négatif)



LES VALEURS QUI EN DÉCOULENT

À partir de cette matrice, on peut tirer plusieurs métriques pour caractériser notre modèle. Yolo nous sort 4 courbes : *précision-confiance, rappel-confiance, précision-rappel et score F1*. On va voir comment les interprêter.
D'abord, définissons la confiance :

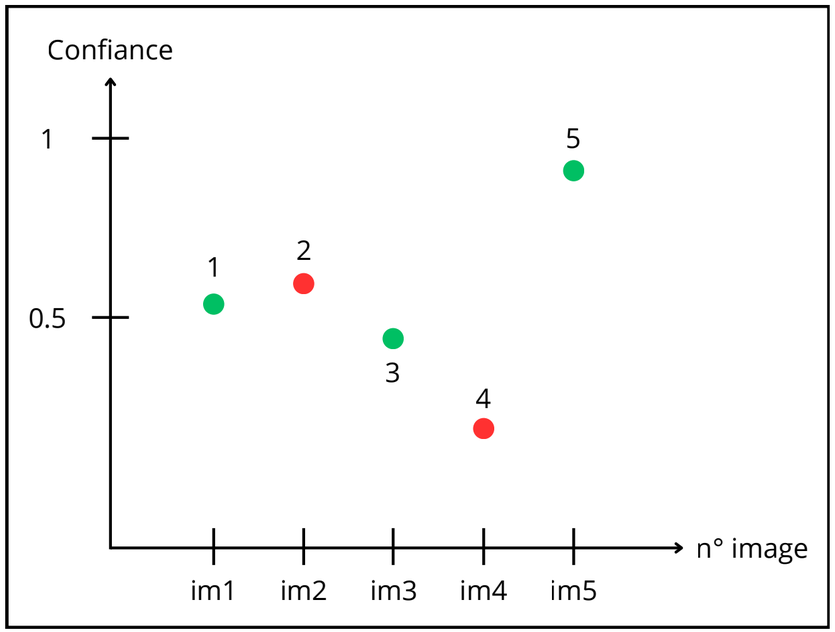
Lorsque le modèle fait une prédiction ce n'est pas binaire comme on pourrait le croire, c'est-à-dire soit c'est un penny soit c'est rien. Le modèle est rarement sûr de lui donc il fourni avec sa détection **une valeur traduisant la certitude de sa prédiction, une probabilité**. Cette valeur, comprise entre 0 et 1 est ce qu'on appelle la *confiance* (1 étant la certitude absolue).

Par exemple, cela pourrait se traduire en langage humain par : "**je pense que cet objet est un penny à 0.75 de confiance**".
Mais aussi : "je pense que cet objet est un penny à 0.1 de confiance".

Il y a donc une nécessité de choisir une **valeur de rupture** pour permettre au modèle de trancher. Comme ci-dessus, si on voit la confiance comme une probabilité, un objet qui a 0.1 de probabilité d'être un penny, n'en est probablement pas un. Donc **il faut fournir au modèle une valeur seuil** pour que si la confiance de sa détection est supérieure, la détection est acceptée (l'objet est labellisé "penny"), sinon, elle est rejeté (l'objet est labellisé "background").

CONFIANCE

Comprendre l’impact de la valeur du seuil de confiance



Prenons le graphique ci-contre comme exemple.

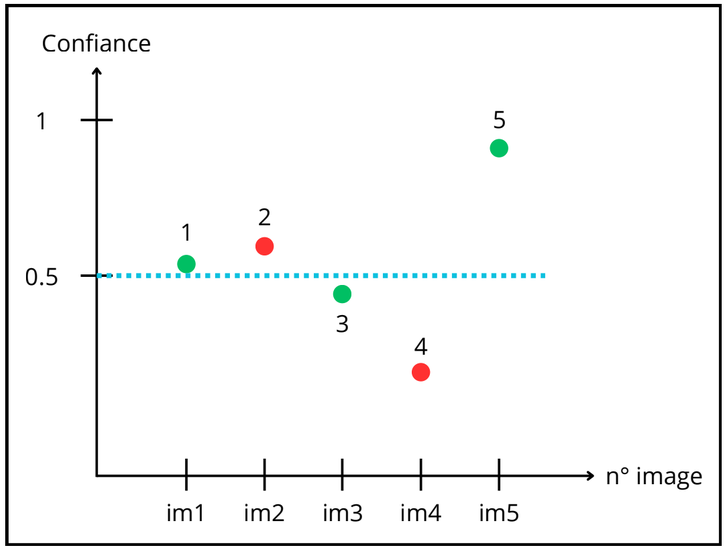
- en abcisse on a des images numérotées de 1 à 5 contenant ou non des “penny”
- en ordonnée on a la confiance accordée aux prédictions
- les données du graphiques (les points verts et rouges) correspondent aux détections du modèle sur les images. Si le point est vert, c’est que l’image contient un penny, s’il est rouge c’est qu’il n’en contient pas.

Par exemple : le point “1” correspond à la prédiction faite sur l’image 1. Le point est vert donc sur l’image il y a un penny et on peut lire en ordonnée, que le modèle détecte la présence d’un penny avec une confiance d’environ 0.55.

Autre exemple : le point “2” correspond à la prédiction faite sur l’image 2. Le point est rouge donc sur l’image il n’y a pas de penny et on peut lire en ordonnée, que le modèle détecte la présence d’un penny avec une confiance d’environ 0.6.

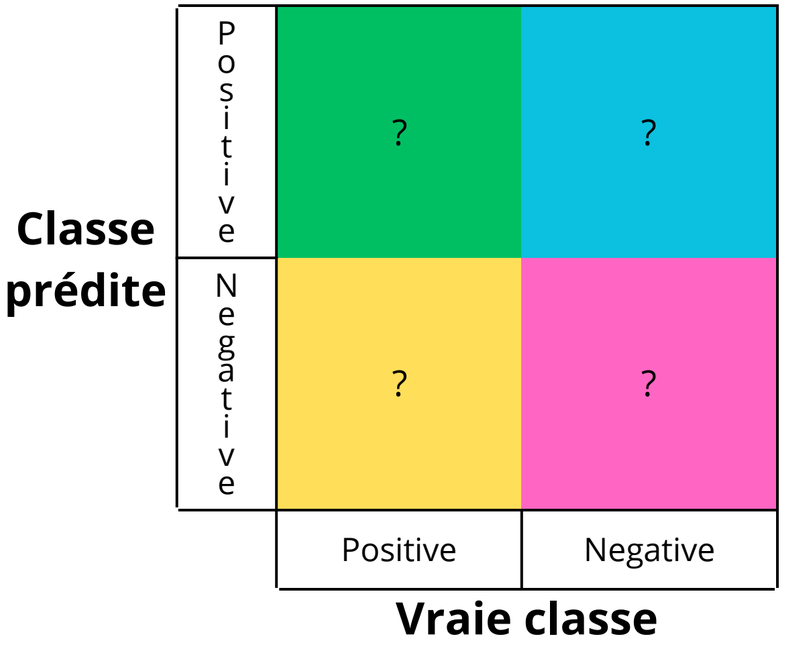
CONFIANCE

Comprendre l'impact de la valeur du seuil de confiance



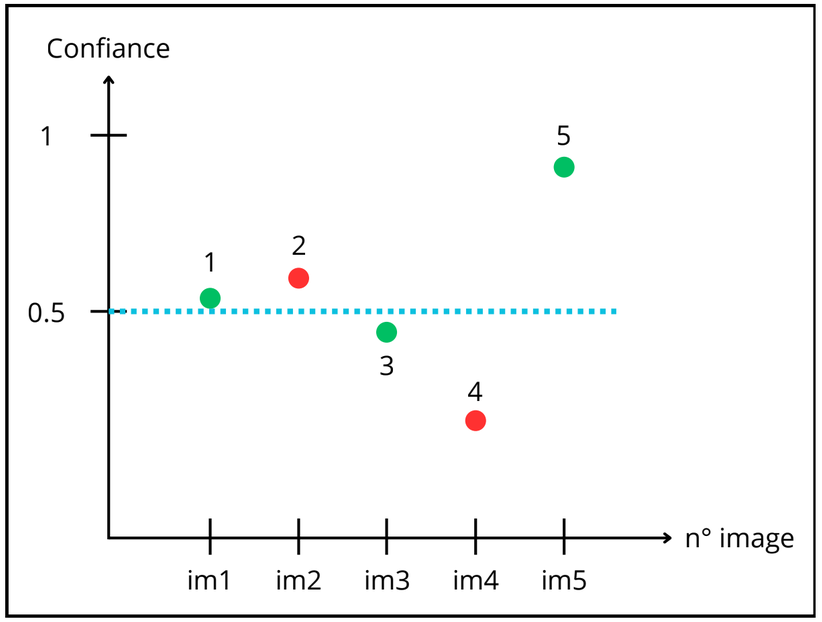
Choisissons arbitrairement un seuil à **0.5**. Visuellement, cela revient à tracer un **trait horizontal** là où la confiance vaut 0.5, puis on "accepte" tous les points au dessus et on "rejète" tous les points en dessous.

Donc maintenant **on est capable de remplir la matrice de confusion** car on a toutes les infos dont on a besoin : la vérité terrain (le point est soit vert soit rouge) et la confiance que le modèle accorde à chacune de ses prédictions.



CONFIANCE

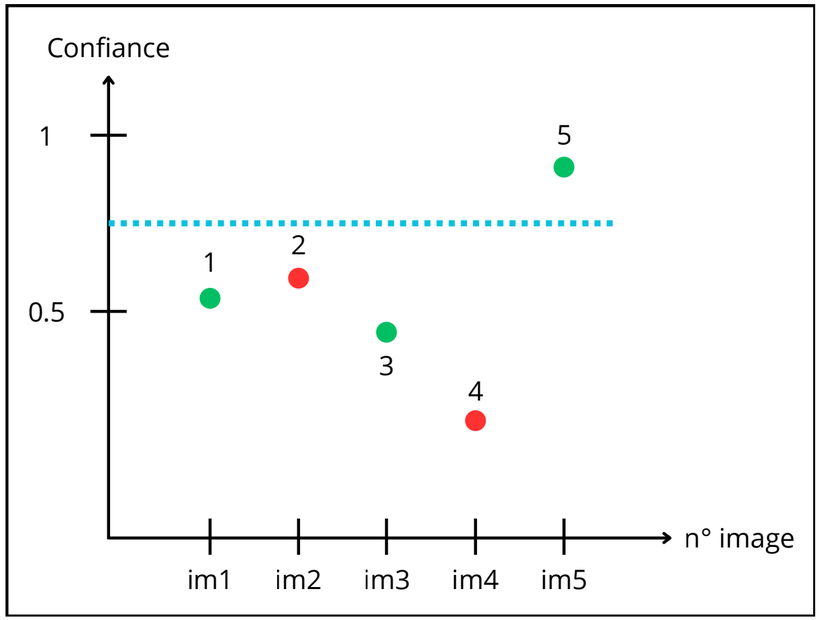
Comprendre l’impact de la valeur du seuil de confiance



Classe prédite \ Vraie classe	Positive	Negative
Positive	im1, im5 = 2	im2 = 1
Negative	im3 = 1	im4 = 1

- le **point “1”** est au dessus du seuil de confiance donc prédit positif (contenant un penny) et est vert (contient bien un penny) → vrai positif (True Positive, TP)
- le **point “2”** est au dessus du seuil de confiance donc prédit positif (contenant un penny) et est rouge (ne contient pas de penny) → faux positif (False Positive, FP)
- le **point “3”** est en dessous du seuil de confiance donc prédit négatif (ne contenant pas de penny) et est vert (contient bien un penny) → faux négatif (False Negative, FN)
- le **point “4”** est en dessous du seuil de confiance donc prédit négatif (ne contenant pas de penny) et est rouge (ne contient pas de penny) → vrai négatif (True Negative, TN)
- le **point “5”** est au dessus du seuil de confiance donc prédit positive (contenant un penny) et est vert (contient bien un penny) → vrai positif (True Positive, TP)

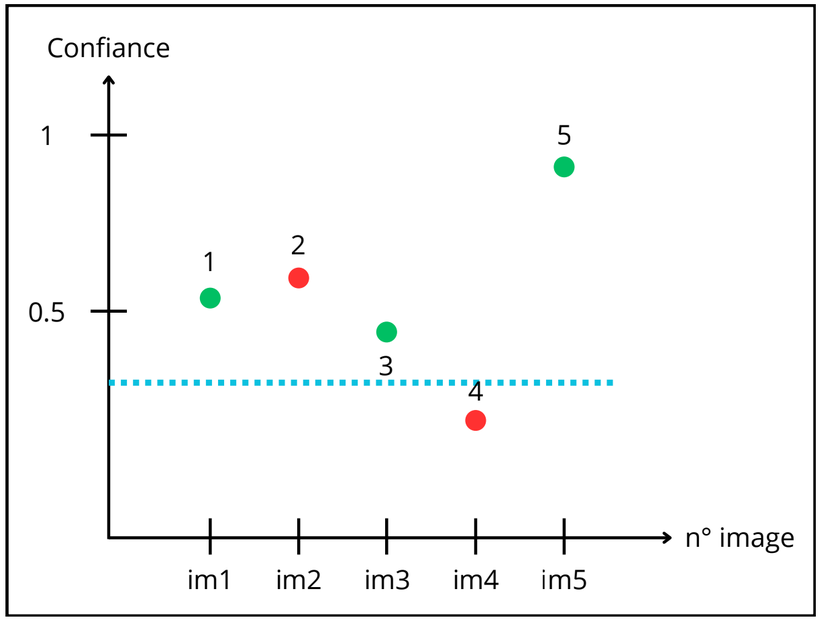
Maintenant prenons un seuil plus haut, vers 0.75.



Classe prédite	Vraie classe: Positive	Vraie classe: Negative
Positive	im5 = 1	0
Negative	im1, im3 = 2	im2, im4 = 2

- le **point “1”** est en dessous du seuil donc prédit négatif mais vert → faux négatif (False Negative, FN)
- le **point “2”** est en dessous du seuil donc prédit négatif et est rouge → vrai négatif (True Negative, TN)
- le **point “3”** est en dessous du seuil donc prédit négatifmais mais est vert → faux négatif (False Negative, FN)
- le **point “4”** est en dessous du seuil donc prédit négatif et est rouge → vrai négatif (True Negative, TN)
- le **point “5”** est au dessus du seuil donc prédit positive et est vert → vrai positif (True Positive, TP)

Enfin, prenons un seuil plus bas, vers 0.35.



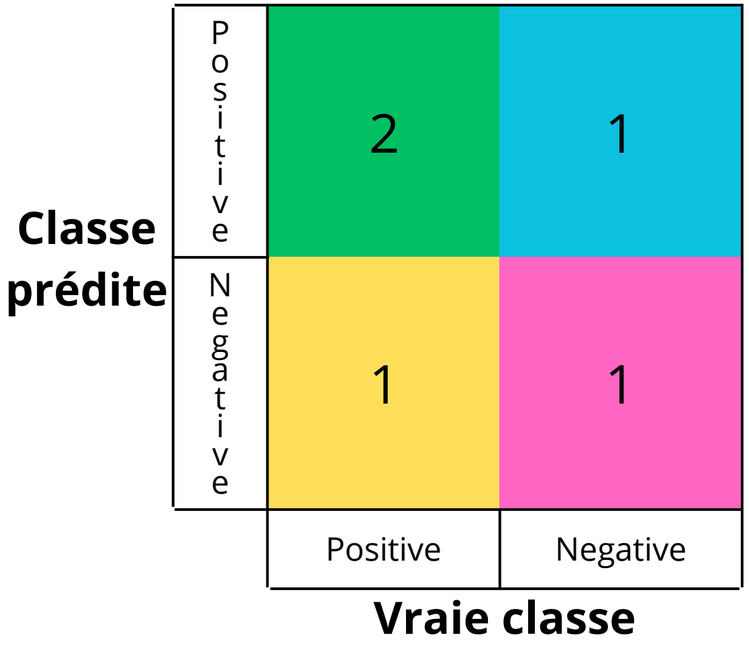
Classe prédite	Vraie classe: Positive	Vraie classe: Negative
Positive	im1, im3, im5 = 3	im2 = 1
Negative	0	im4 = 1

- le **point “1”** est au dessus du seuil donc prédit positif et est vert → vrai positif (True Positive, TP)
- le **point “2”** est au dessus du seuil donc prédit positif mais est rouge → faux positif (False Positive, FP)
- le **point “3”** est au dessus du seuil donc prédit positif et est vert → vrai positif (True Positive, TP)
- le **point “4”** est en dessous du seuil donc prédit négatif et est rouge → vrai négatif (True Negative, TN)
- le **point “5”** est au dessus du seuil donc prédit positive et est vert → vrai positif (True Positive, TP)

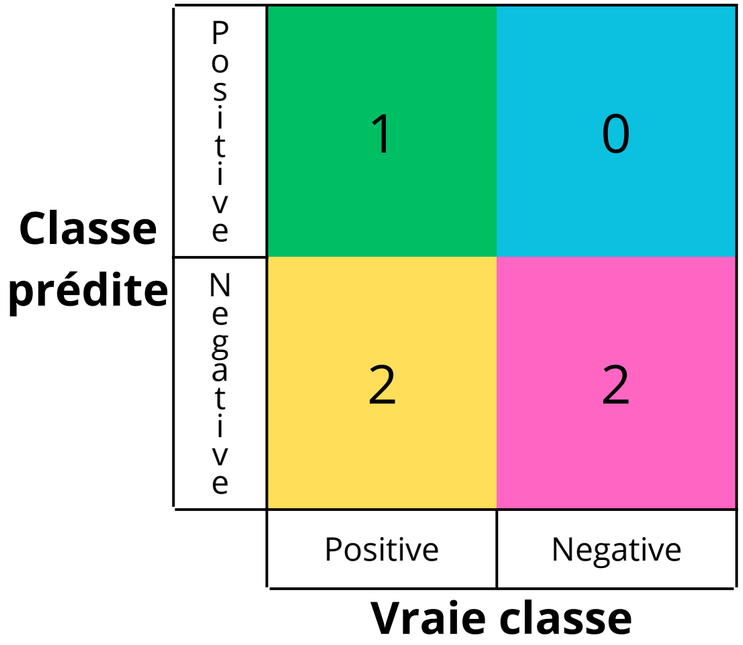
CONFIANCE

Comprendre l’impact de la valeur du seuil de confiance

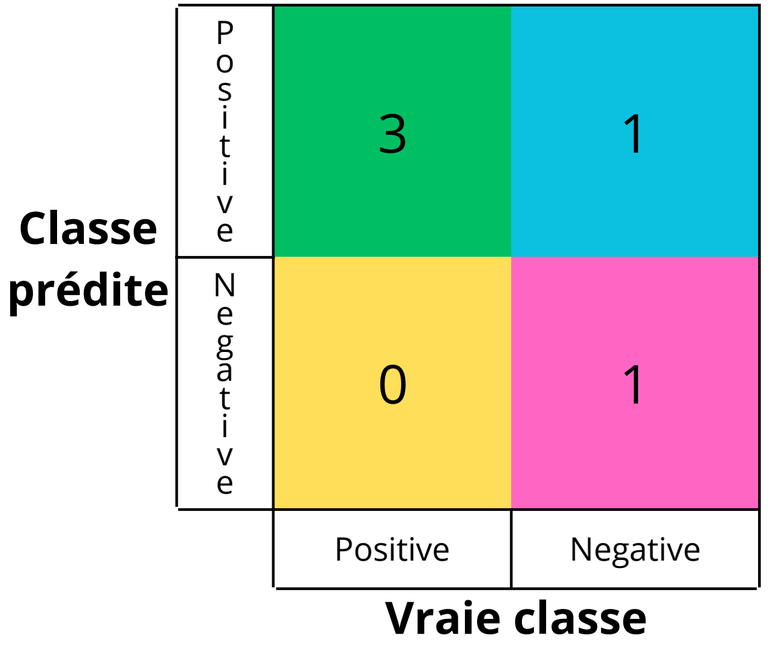
Pour un même modèle, on peut donc voir qu’il y a **plusieurs matrices de confusion possibles**, directement liées à la valeur du seuil de confiance choisi.



seuil de confiance = 0.5



seuil de confiance = 0.75



seuil de confiance = 0.35

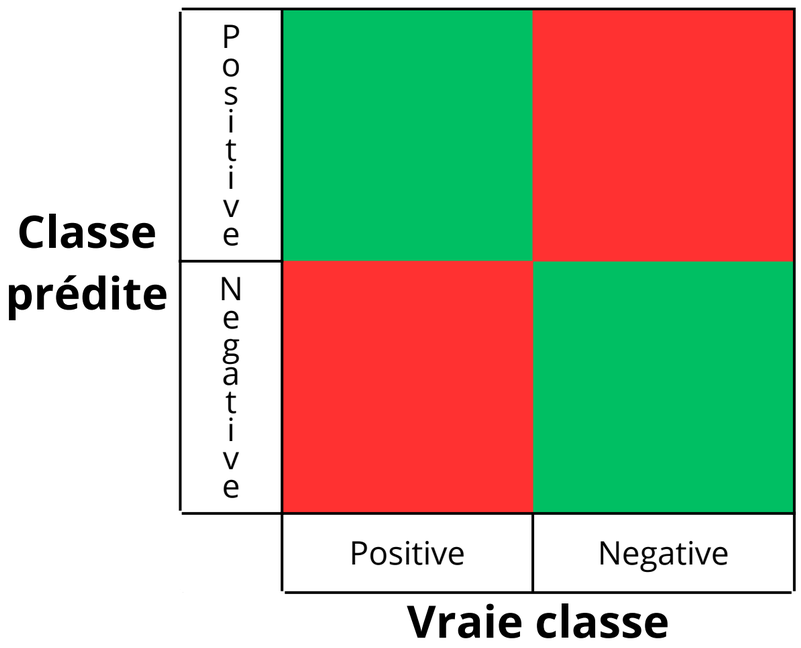
Pour pousser la logique jusqu’au bout, on a un nombre infini de matrice de confusion car on en a autant qu’il y a de nombres décimaux entre 0 et 1 (mais pour des seuils très proches, les matrices ne devraient pas tellement changer).

MATRICE DE CONFUSION

À quoi sert-elle concrètement ?

Cette matrice est un **indicateur de la qualité du seuil de confiance** choisi. Car un des enjeux à la fin de l'entrainement de notre modèle est de **choisir le meilleur seuil de confiance** pour déployer le modèle.

Mais comment savoir si on a une bonne ou une mauvaise matrice de confusion ? Grossièrement, si on veut simplifier la compréhension de la matrice, on peut la voir de manière binaire : il y a la *bonne diagonale* et la *mauvaise diagonale* :



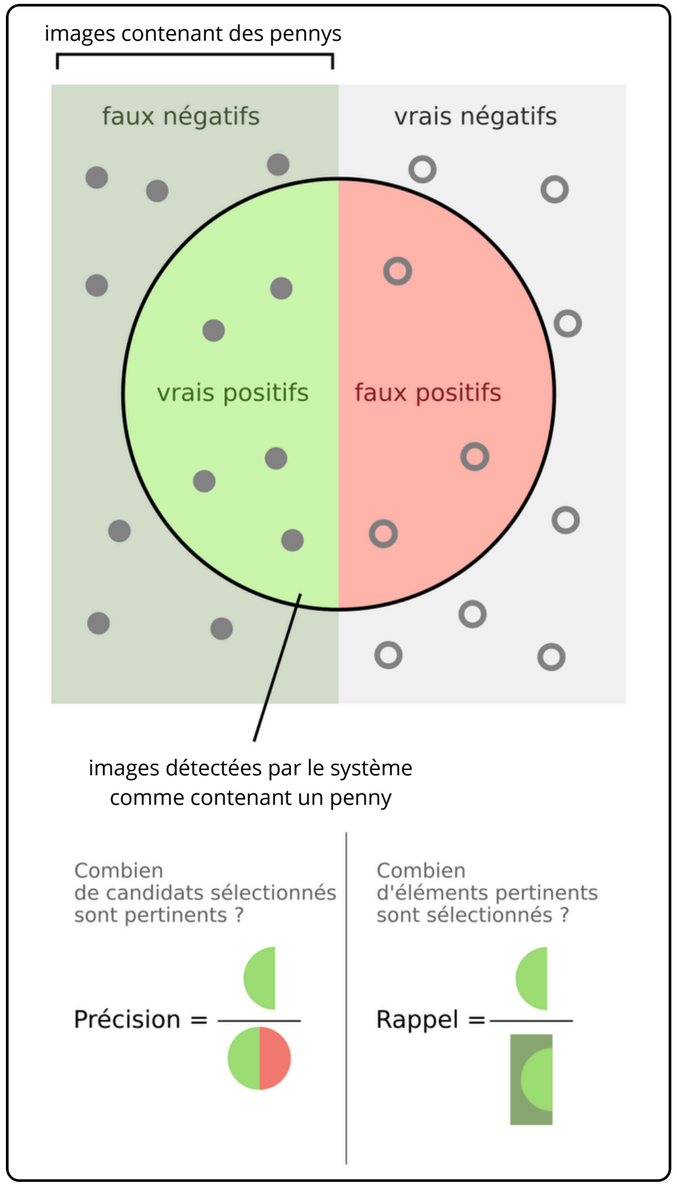
La *bonne diagonale* : la classe prédite est identique à la vraie classe.

La *mauvaise diagonale* : la classe prédite est différente de la vraie classe.

Intuitivement, on comprend qu'**il faut faire en sorte de baisser les valeurs de la mauvaise diagonale**, mais si vous regardez les matrices de la page précédente, vous pouvez observer que ce n'est pas évidement de mettre les deux cases à 0 simultanément.

Alors des fois il faut choisir lequel des deux facteurs est le plus important et "délaisser" l'autre : **en médecine, on ne veut surtout pas de faux négatifs (ne pas détecter un cancer bien présent)** donc on va minimiser au maximum les faux négatifs aux dépens des faux positifs qui sont moins graves (détecter un cancer alors qu'il n'y a rien).

LES MESURES D'ÉVALUATION



Sachant qu'**une matrice est propre à un seuil de confiance**, comment savoir quel seuil de confiance est le meilleur ? On va utiliser des mesures pour pouvoir savoir !

Précision

Désigne la *qualité* des prédictions.

$$\textbf{précision} = \frac{\text{vrai penny détecté}}{\text{vrai penny détecté + faux penny détecté}} = \frac{\text{vrai positif}}{\text{vrai positif + } \textbf{faux } \textbf{\textit{positif}}}$$

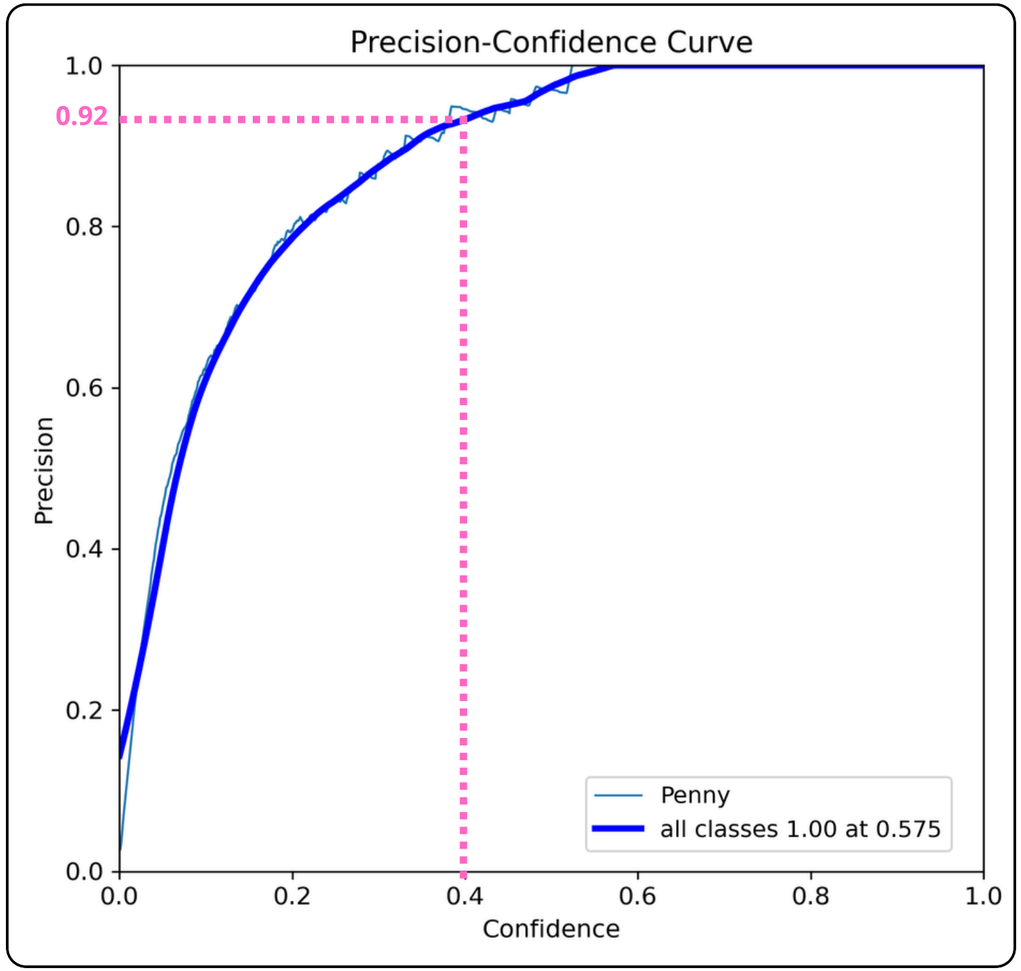
Rappel (ou sensibilité)

Désigne la *quantité* des prédictions.

$$\textbf{rappel} = \frac{\text{vrai penny détecté}}{\text{vrai penny détecté + vrai penny } \textbf{\textit{non}} \text{ détecté}} = \frac{\text{vrai positif}}{\text{vrai positif + } \textbf{faux } \textbf{\textit{négatif}}}$$

Cliquez pour faire apparaître un schéma.

PRÉCISION-CONFIANCE



$$\textbf{précision} = \frac{\text{vrai positif}}{\text{vrai positif} + \textbf{faux positif}}$$

Comment la construire ?

Pour tous les seuils de confiance compris entre 0 et 1 :

- on établit la matrice de confusion pour ce seuil
- on calcule la précision associée à cette matrice spécifique
- on place le point dans le graphique

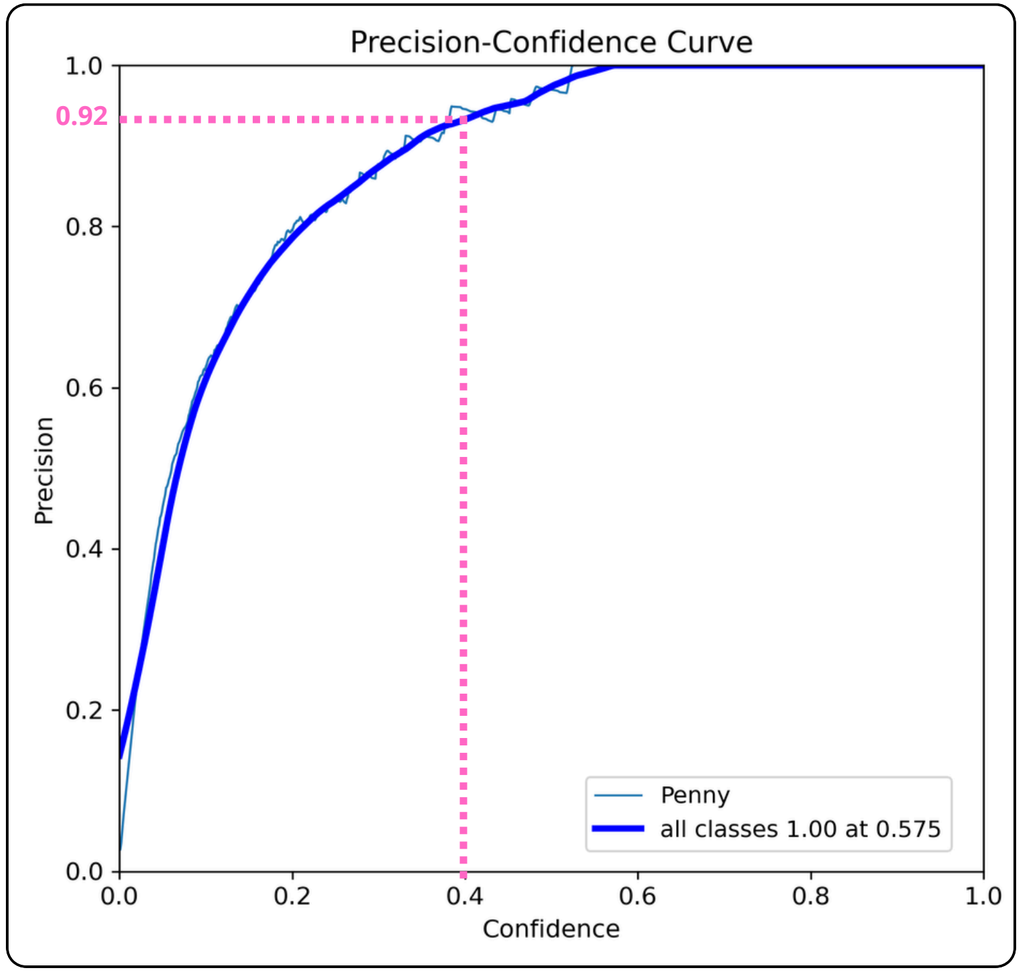
Une fois tous les points placés, on les relie.

À l'inverse, comment la lire ?

Par exemple, pour un seuil de confiance de 0.4, on lit une précision de 0.92, on en déduit que pour un tel seuil on a assez peu de faux positifs.

À retenir : plus la courbe est haute, meilleur est le modèle.

PRÉCISION-CONFIANCE



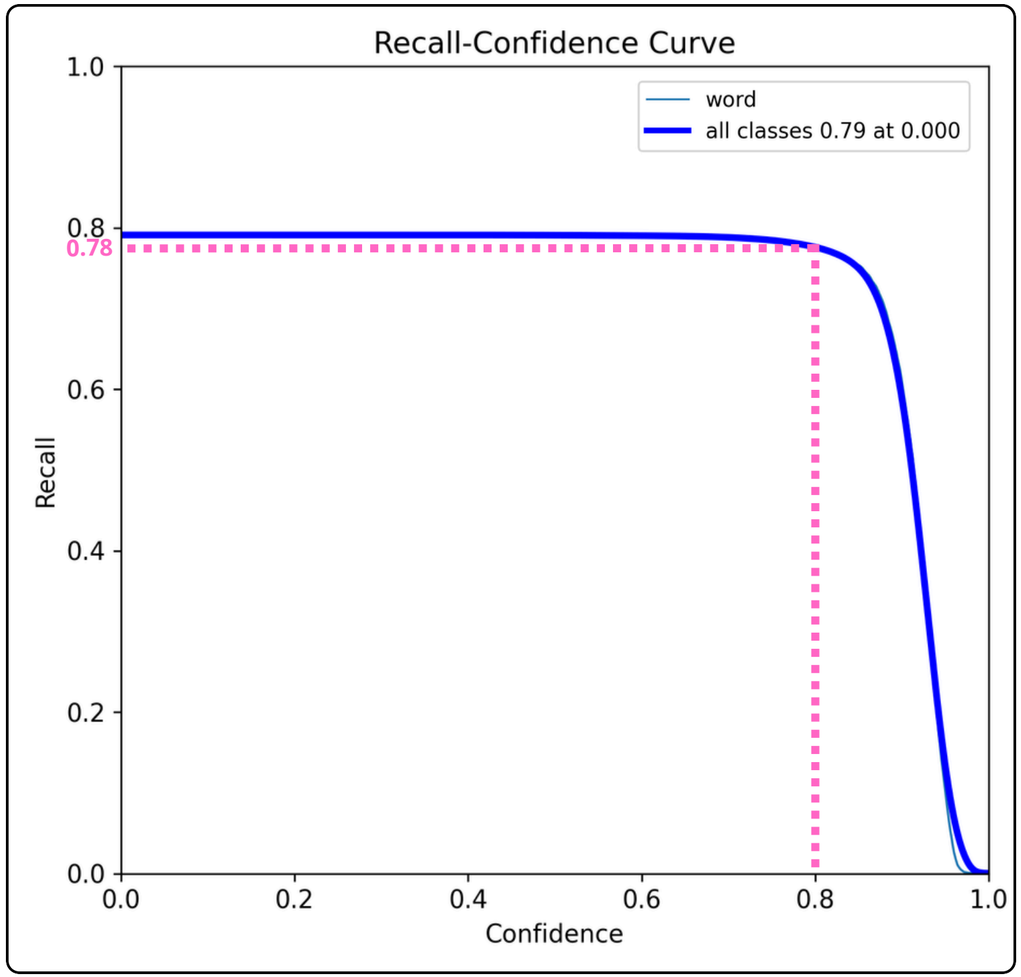
$$\textbf{précision} = \frac{\text{vrai positif}}{\text{vrai positif} + \textbf{faux positif}}$$

Pour aller plus loin (vous pouvez passer si vous le souhaitez)

Il faut faire attention avec cette courbe car elle ne prend pas du tout en compte les faux négatifs, mais surtout, plus le seuil monte, de moins en moins de valeur sont prises en compte donc la montée de la précision est “artificielle”.

Cela s’explique car le modèle n’accepte quasiment plus rien, donc si il ne détecte que un vrai positif avec 98% de confiance et c’est tout, il aura donc 100% de précision. Mais dans ce cas, plein de “vrais positifs”, jugé négatifs car sous le seuil, seront oubliés.

RAPPEL-CONFIANCE



précision = $\frac{\text{vrai positif}}{\text{vrai positif + } \mathbf{\text{faux négatif}}}$

Comment la construire ?

Pour tous les seuils de confiance compris entre 0 et 1 :

- on établit la matrice de confusion pour ce seuil
- on calcule le rappel associé à cette matrice spécifique
- on place le point dans le graphique

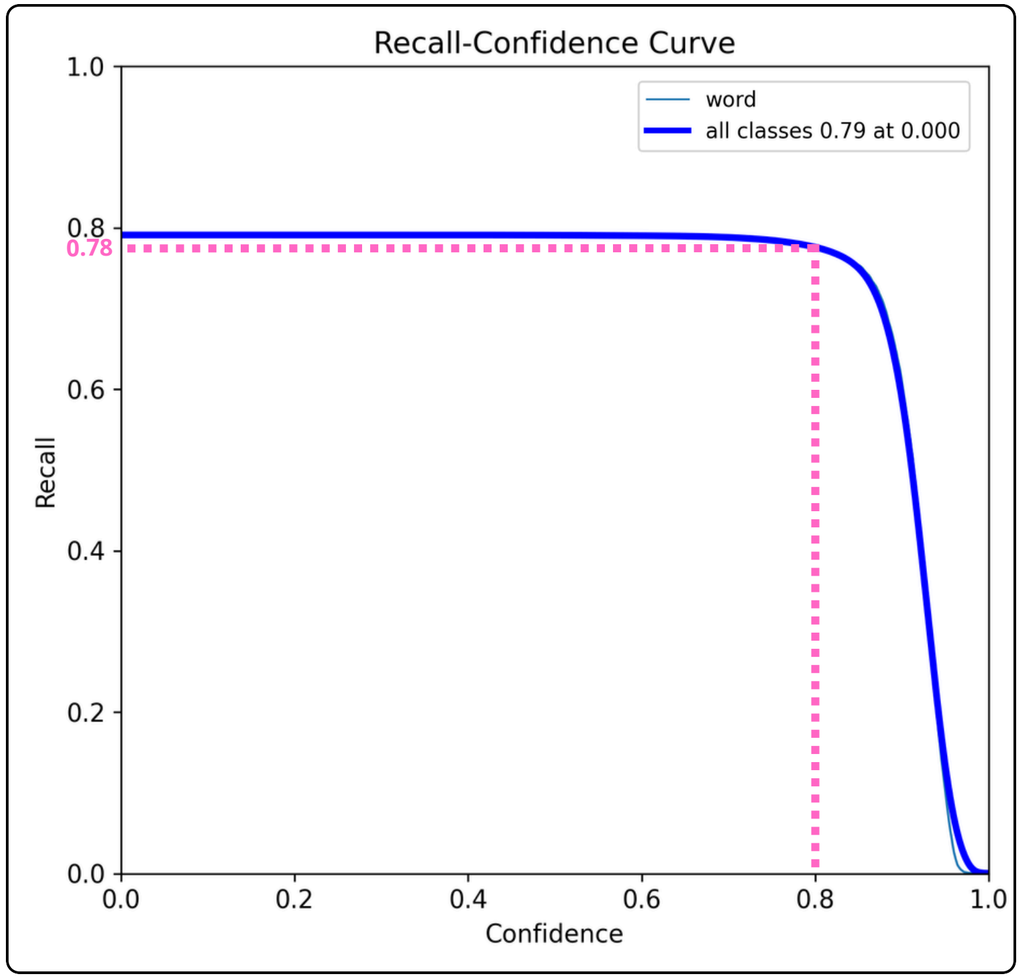
Une fois tous les points placés, on les relie.

À l'inverse, comment la lire ?

Par exemple, pour un seuil de confiance de 0.8, on lit un rappel de 0.78, on en déduit que pour un tel seuil on a beaucoup de faux négatifs.

À retenir : plus la courbe est haute et met du temps à descendre, meilleur est le modèle.

RAPPEL-CONFIANCE



précision = $\frac{\text{vrai positif}}{\text{vrai positif + } \mathbf{\text{faux négatif}}}$

Pour aller plus loin (vous pouvez passer si vous le souhaitez)

La courbe finira nécessairement à 0 car plus on augmente le seuil de confiance, plus le modèle est “strict”, il n’accepte que des prédiction dont il est sur, donc beaucoup d’images posititives finissent négatives donc beaucoup de faux négatifs apparaissent ce qui annule la fraction.

LES MESURES D'ÉVALUATION

La précision et le rappel sont deux métriques intéressantes mais sont complémentaires, c'est à dire que pour avoir une vision globale du modèle **il faut regarder les deux en même temps** mais surtout voir comment l'une évolue par rapport à l'autre.

Pour éviter de faire des va-et-viens entre les deux graphiques, **des métriques les combinent directement**, donc on introduit :

Précision-rappel

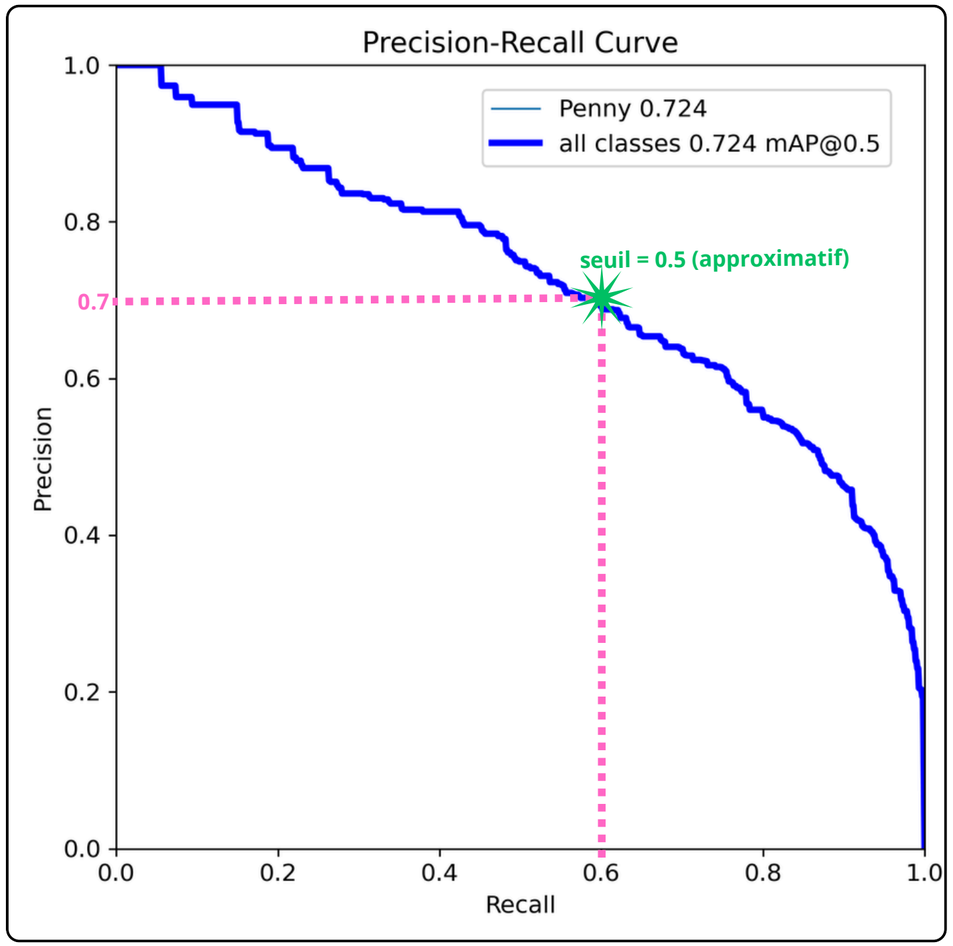
Cette courbe permet de voir l'évolution simultanément de la précison et du rappel selon la confiance. Elle est construite de manière très spéciale, on verra ça juste après.

F1-score

Le F1-score est la moyenne harmonique de la précision et du rappel. Cette courbe est utilisée pour trouver le seuil de confiance optimal (le pic de la courbe) pour déployer votre modèle.

$$\textbf{F1-score} = 2 \times \frac{\text{précision x rappel}}{\text{précision + rappel}} = \frac{2 \times (\text{vrai positif})}{2 \times (\text{vrai positif}) + \textbf{faux positif + faux négatif}}$$

PRÉCISION-RAPPEL



Comment la construire ?

Cette courbe est construite point par point car une troisième dimension est cachée, la **confiance**.

On commence par le point le plus à gauche : ce point correspond à la valeur de la précision et du rappel lorsque le seuil de confiance est à 1.

On trace ensuite celui juste à sa droite : on prend le seuil 0.99 et on calcule la précison et le rappel depuis la matrice de confusion.

Et ainsi de suite.

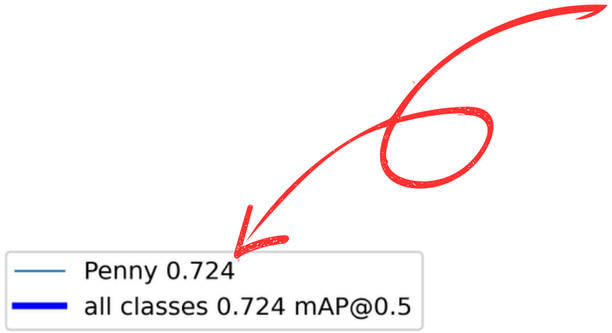
À l'inverse, comment la lire ?

Prenons le point où le seuil de confiance est à 0.5 (si on prend la courbe à ses extrémités avec nos doigts et qu'on la tend, ce sera le milieu), on peut lire qu'à ce seuil, la précision est de 0.7 et le rappel à 0.6.

À retenir : plus la courbe monte dans le coin en haut à droite, meilleur est le modèle.

AVERAGE PRECISION

Pour aller plus loin (vous pouvez passer si vous le souhaitez)



Cette courbe est utilisée pour calculer une autre métrique que vous retrouverez plus loin dans les résultats : **Average Precision (AP)**. Cette métrique correspond à l'**aire sous la courbe de précision-rappel**. C'est la valeur décimal que l'on peut lire à côté du nom de la classe dans la légende.

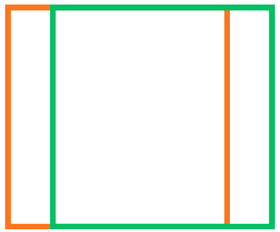
Par extension la **mAP correspond à la *mean* Average Precision** donc à la moyenne des AP quand on a plusieurs classes (on verra après).

Juste après "mAP", vous pouvez lire "@0.5" ce qui correspond à la valeur seuil pour "l'**Intersection over Union**" (IoU).

Pour comprendre rapidement ce que c'est, il faut le voir comme l'**outil de vérification du modèle** : le modèle prédit une boite pour englober un objet et d'autre part, il a la boite de vérité terrain. Mais il a besoin de **quantifier si sa boite à lui est bien placée par rapport à l'endroit exacte où elle devait être** donc on introduit un nouvel outil que l'on va expliquer : IoU.

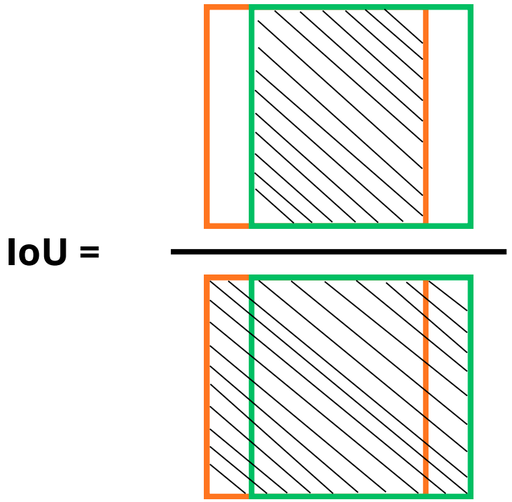
INTERSECTION OVER UNION

Pour aller plus loin (vous pouvez passer si vous le souhaitez)



Prenons ces deux boites : **la boite verte** correspond à la vérité terrain et **la boite orange** a la prédiction du modèle. Elle n'est pas placée idéalement pourtant elle semble quand même englober le même objet.

Donc pour avoir une valeur attestant de la précision du placement de la boite, on va calculer le rapport entre l'intersection des deux boites et leur union, comme vous pouvez le visualiser ci-contre :



Maintenant, ce rapport est à titre indicatif et donc comme la confiance, il faut choisir un seuil qui permet de rejeter ou garder une prédiction. Dans le cas précédent où on avait un seuil à 0.5, cela revient à dire que si une boîte de prédiction ne recouvre pas la boite vérité terrain à au moins 50%, c'est qu'elle est mal placée donc non acceptée.

CONFIANCE - IOU - MAP : LE LIEN ?

Pour aller plus loin (vous pouvez passer si vous le souhaitez)

Il y a plusieurs notion qui semble s'entrecroiser donc on va faire une frise chronologique pour bien comprendre comme ça marche.

Étape 1 : choix des valeurs des seuils de confiance et de IoU donc si on prend les valeurs par défault de YOLOv26, on a :

- seuil de confiance = 0.25
- seuil d'IoU = 0.5

Étape 2 : le modèle commence à s'entrainer, donc il regarde une image, puis fait des prédictions qui sont accompagnées chacune d'une confiance.

Étape 3 : d'après le seuil de confiance donné (0.25), **le modèle accepte toutes les prédictions dont la confiance est supérieure et rejette les autres.**

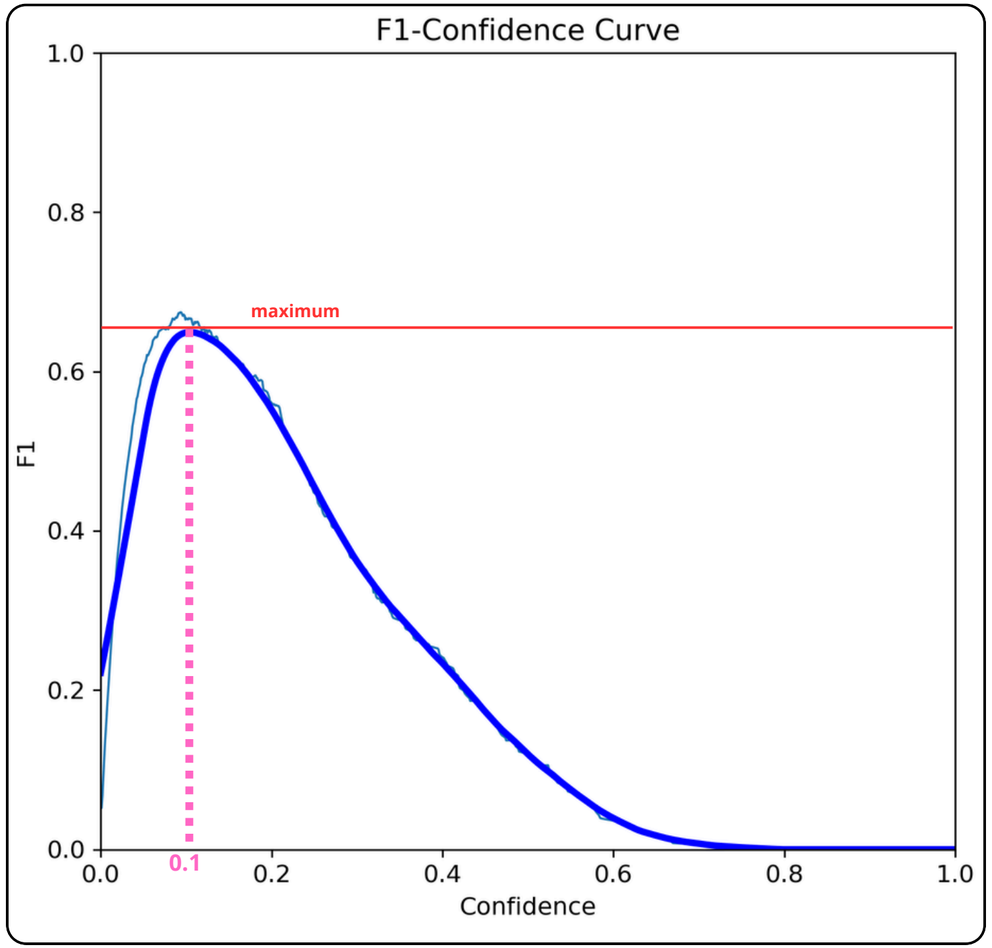
Étape 4 : vient la vérification des boîtes, donc comme pour le jeu des 7 erreurs de tout à l'heure, le modèle prend l'image avec les prédiction d'un côté et l'image avec la vérité terrain de l'autre puis compare. Chacune des boites prédites est comparée via l'IoU aux boites de vérité terrain, et **si le ratio dépasse le seuil (0.5), alors la boite est validée** en tant que "vrai positif". Sinon, **si l'IoU est inférieur, la boite est rejeté** mais comme **elle avait passé le test de confiance**, alors c'est une fausse alerte donc "faux positif".

Étape 5 : enfin, une fois que tout est vérifié, les boites de la vérité terrain qui n'ont pas trouvé détection à leur pied se retrouve dans les "faux négatifs".

Donc une **matrice de confusion dépend du seuil de confiance mais aussi du seuil de l'IoU.**

Enfin, il existe une dernière métrique que vous pourrez observer plus tard : la **mAP50-95**. Elle consiste à calculer la mAP pour des valeurs de seuils d'IoU entre 0.5 et 0.95 avec un pas de 0.05 (0.5, 0.55, 0.6, ..., 0.9, 0.95) et à en faire la moyenne.
Cette métrique est particulièrement intéressante car elle est exigeante en termes de précision de placement de la boite donc en regardant l'évolution de cette valeur, on peut voir si notre modèle englobe correctement. C'est une métrique très commune dans l'industrie.

F1SCORE-CONFIANCE



F1-score = $2 \times \frac{\text{précision x rappel}}{\text{précision + rappel}}$

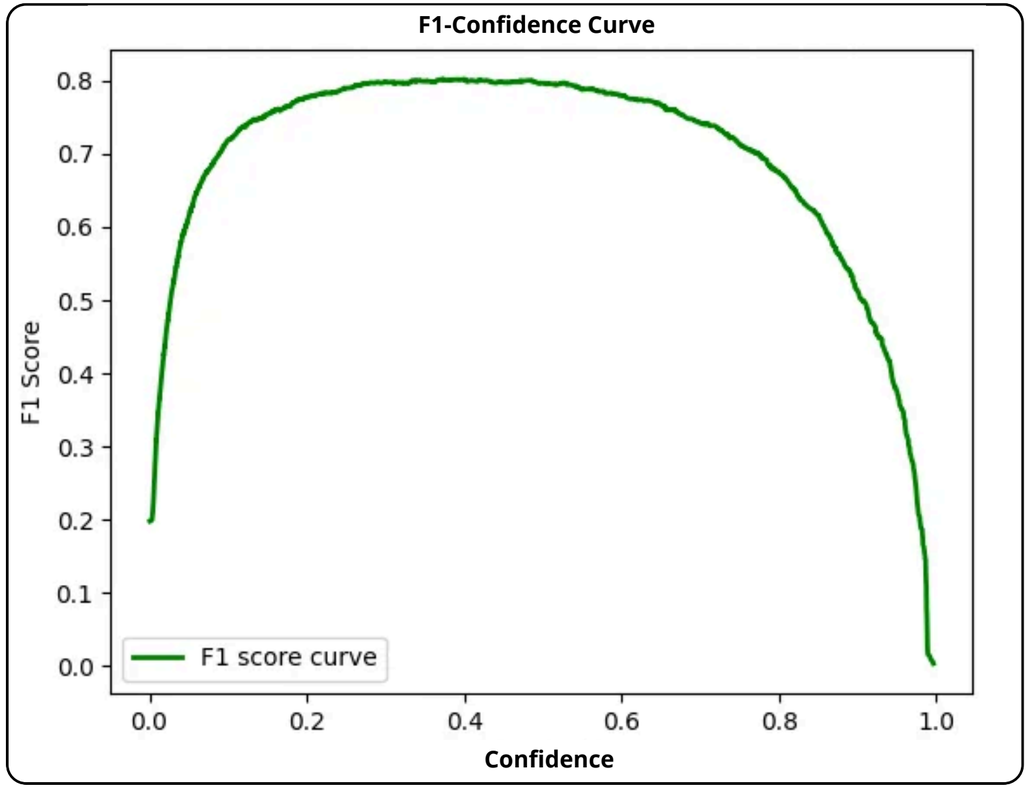
Cette courbe est la moyenne harmonique de la précision et du rappel en fonction du seuil de confiance.

Cette courbe sert à trouver quel seuil optimal, c'est-à-dire celui qui permet de maximiser la précision et le rappel en même temps. **Ce seuil est l'abcisse du maximum de la courbe** comme dessiné sur la courbe ci-contre.

On trouve donc ici qu'un seuil de 0.1 de confiance serait optimal pour maximiser rappel et précision en même temps. (C'est un très mauvais modèle car c'est juste pour l'exemple!)

À retenir : prendre le point le plus haut, meilleur sera le seuil de confiance, meilleur sera le modèle.

F1SCORE-CONFIANCE



Lorsque la courbe F1 présente un plateau plutôt qu'un pic unique, le choix du seuil de confiance idéal dépend de vos objectifs spécifiques :

Priorité à la précision (éviter les faux positifs) : choisissez la valeur à l'extrémité droite du plateau (seuil plus élevé). Cela garantit que le modèle ne prédit que lorsqu'il est très sûr, minimisant les erreurs de détection.

Priorité au rappel (éviter les faux négatifs) : choisissez la valeur à l'extrémité gauche du plateau (seuil plus bas). Cela permet de capturer un maximum d'objets réels, quitte à accepter quelques fausses alertes.

Équilibre : Si aucune contrainte ne prévaut, le centre du plateau est souvent un choix robuste.

MATRICE DE CONFUSION MULTICLASSE

On a vu toute la théorie, maintenant quelques subtilités qui peuvent arriver dans vos entrainements, notamment le multiclasse.

Qu’est-ce que le multiclasse ?

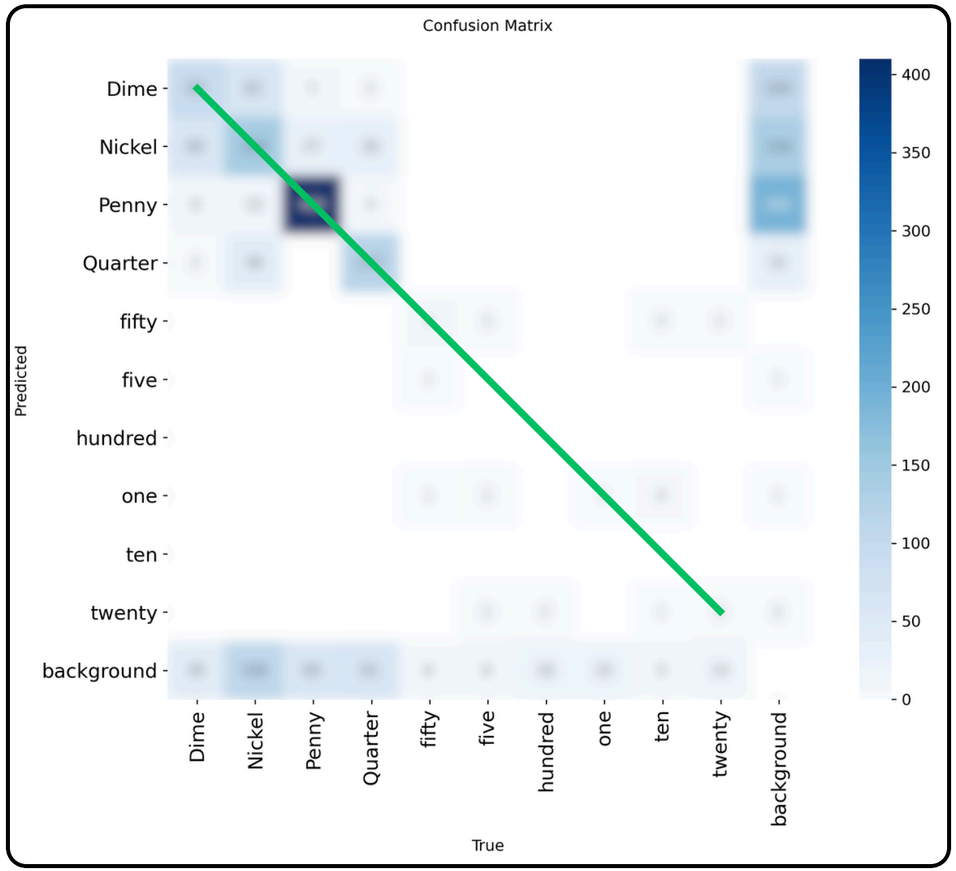
C’est quand vous avez plus d’une classe à détecter, par exemple vous avez chat, chien et lapin.

Dans votre matrice de confusion, vous aurez donc **autant de ligne et de colonne que de classe**, auxquelles on rajoute la ligne et la colonne “negative” (”background”) qui correspond à l’absence de classe.

Voyons à quoi elle ressemble et comment s’en servir pour calculer précision et rappel :

MATRICE DE CONFUSION

La subtilité du multiclasse



Les lignes correspondent aux prédictions et les colonnes à la vérité terrain.

On retrouve bien la *bonne diagonale* qui correspond toujours au fait que la classe de la prédiction est la même classe que la vérité terrain.

Mais il n'y a plus de *mauvaise diagonale.* Les mauvaises prédictions couvrent maintenant le reste des cases de la matrice.

La colonne tout à droite correspond aux objets qui ont été détecté mais faisaient en fait partie du fond. Inversement, la dernière ligne correspond aux choses non détectés qui faisaient en fait partie d'une classe.

MATRICE DE CONFUSION

La subtilité du multiclasse

Avec la nouvelle forme de matrice de confusion que nous venons de voir, il va être nécessaire de **définir** qui sont les **vrais positifs, vrais négatifs, faux positifs** et **faux négatifs** comme vu précédemment, pour pouvoir en sortir les calculs de **précision, rappel et F1-score** pour trouver le **meilleur seuil** pour le déploiement de notre modèle.

Pour ramener une matrice multiclasse à une matrice 2x2, **on va fonctionner classe par classe**. À la fin on devrait avoir autant de matrice de confusion 2x2 que de classe à détecter.

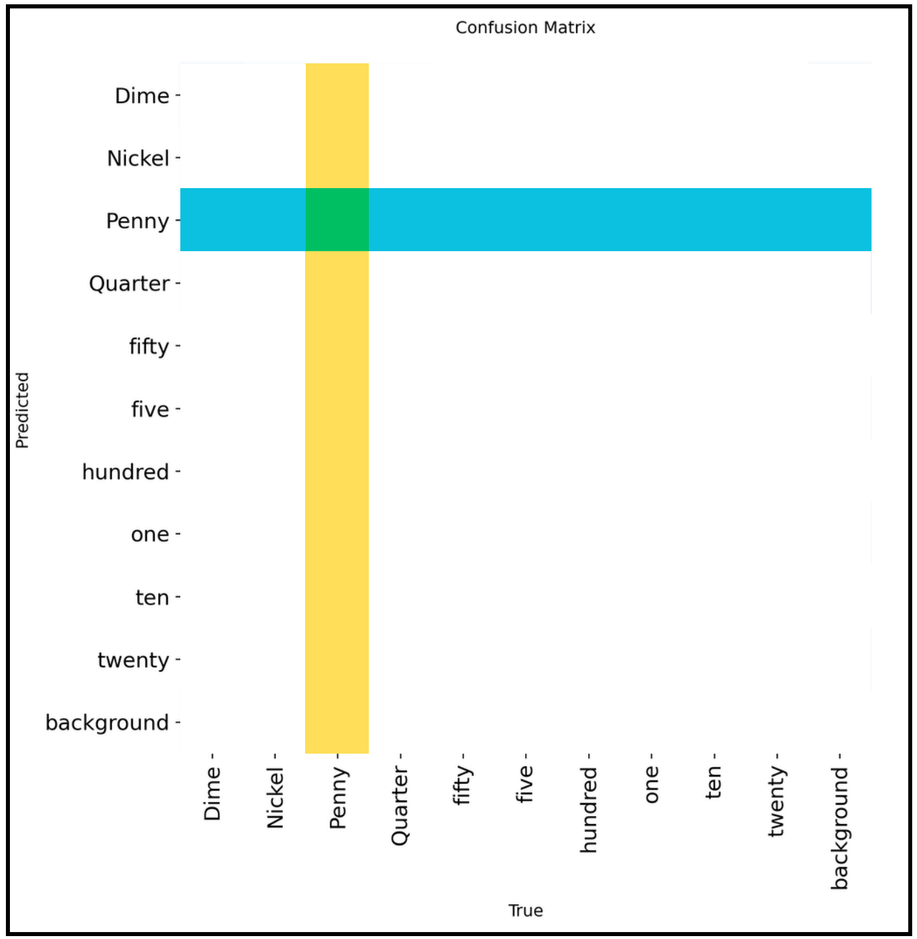
Pour les remplir, on va voir ensemble comment déterminer qui est quoi.

Déjà, nous n'avons pas besoin des vrais négatifs. En effet, vous l'aurez peut-être remarqué (ou non), mais les vrais négatifs (true negative) ne sont dans aucune formule.
En effet, pour de la détection, un vrai négatif n'a pas vraiment de sens, ça revient à dire : il n'y avait rien à détecter et rien n'a été détecté, mais à l'échelle d'une image, cela se transforme vite en *bruit*, c'est-à-dire que chaque pixel pourrait être un vrai négatif avec cette définition, donc on ne les prend pas en compte.

Un fois qu'on a dit ça, on a éliminé une case ! Voyons comment trouver les trois restantes page suivante.

MATRICE DE CONFUSION

La subtilité du multiclasse



TP : True Positive **FP : False Positive** **FN : False Negative**

Prenons encore une fois la classe “Penny” comme exemple.

Les **vrais positifs** sont ceux prédit “penny” qui en était bien donc on a qu’une seule case, celle qui est verte.

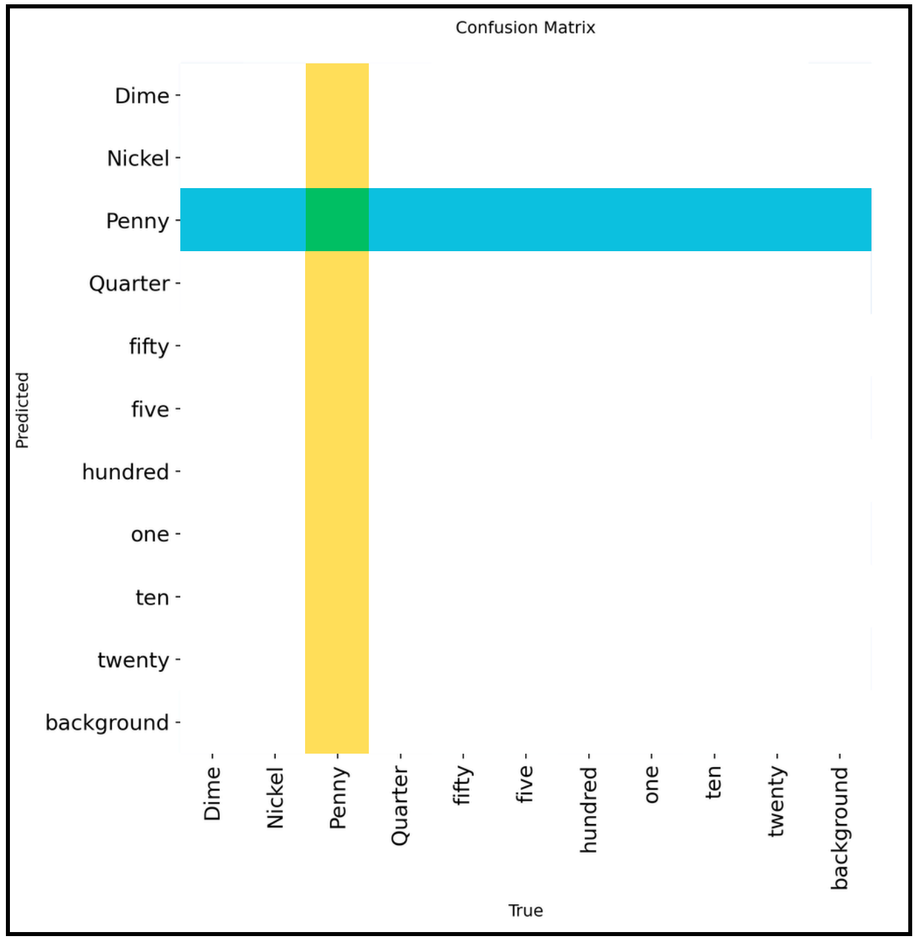
Ensuite, les cases de la même ligne correspondent aux objets que le modèle a prédit comme “penny” mais qui n’en était pas donc les **faux positifs** (les cases sur la même ligne que la case verte).

Enfin, les cases de la même colonne correspondent aux objets que le modèle a prédit comme tout sauf “penny” mais qui était bien un “penny” donc les **faux négatifs** (les cases sur la même colonne que la case verte).

Pour les faux négatifs et les vrais négatifs, on va sommer les valeurs de toutes les cases concernées.

MATRICE DE CONFUSION

La subtilité du multiclasse



TP : True Positive **FP : False Positive** **FN : False Negative**

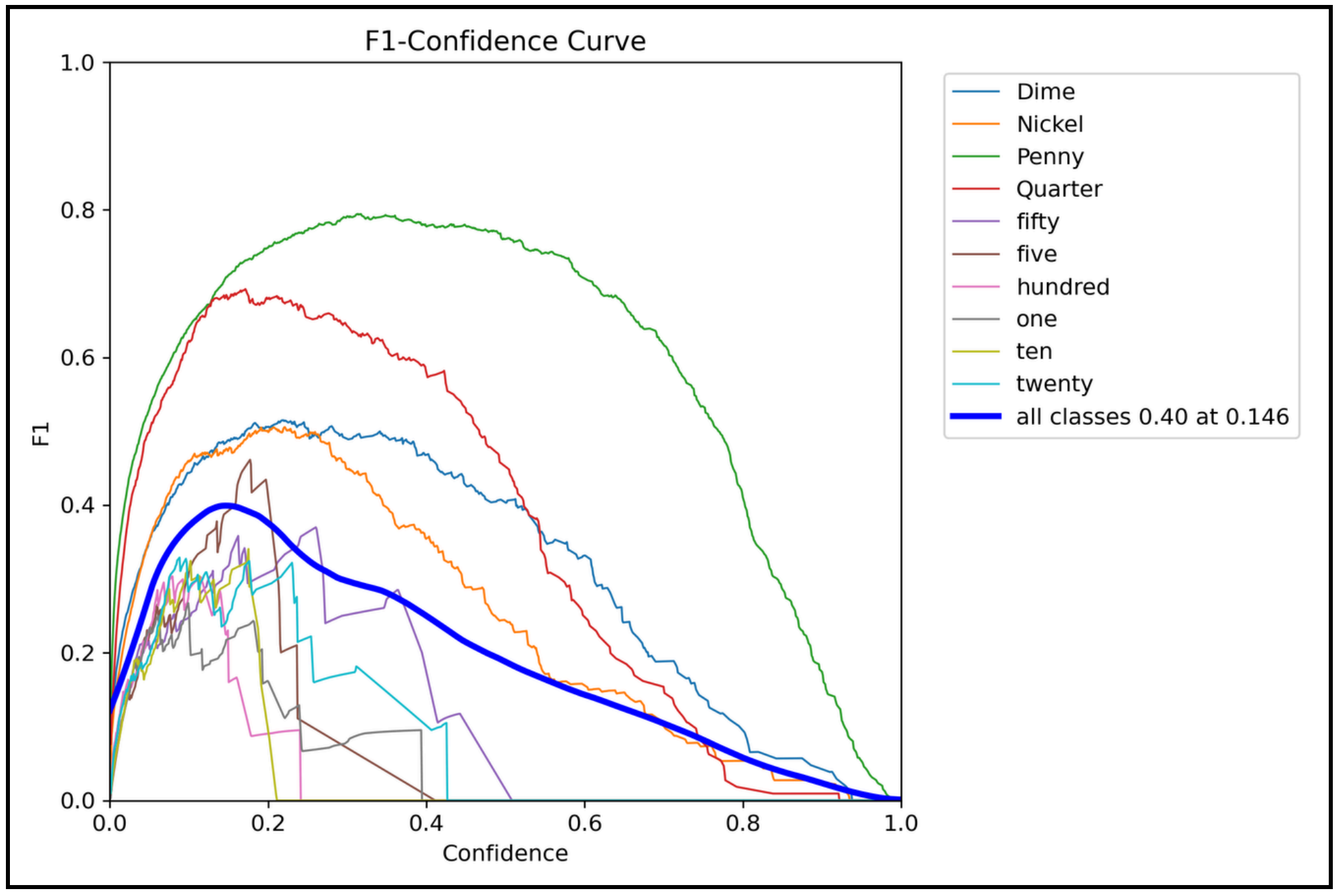
Une fois qu'on a nos 3 valeurs (TP, TN, FP), on va pouvoir calculer précision, rappel. Comme vu précédement, on va refaire ce processus avec **tous les seuils de confiance** pour pouvoir tracer les courbes de précision et de rappel en fonction de ce seuil.

Puis quand les courbes d'une classe sont tracées, on fait avec la classe d'après et ainsi de suite jusqu'à ce qu'elles soient toutes faites. On se retrouve donc avec **autant de courbe que de classe**.

Pour avoir une vue d'ensemble, on fait une **moyenne** de toutes ces courbes, c'est la **courbe en gras** que l'on verra juste après sur les graphiques.

COURBE DU SCORE F1

L’exemple du multiclasse



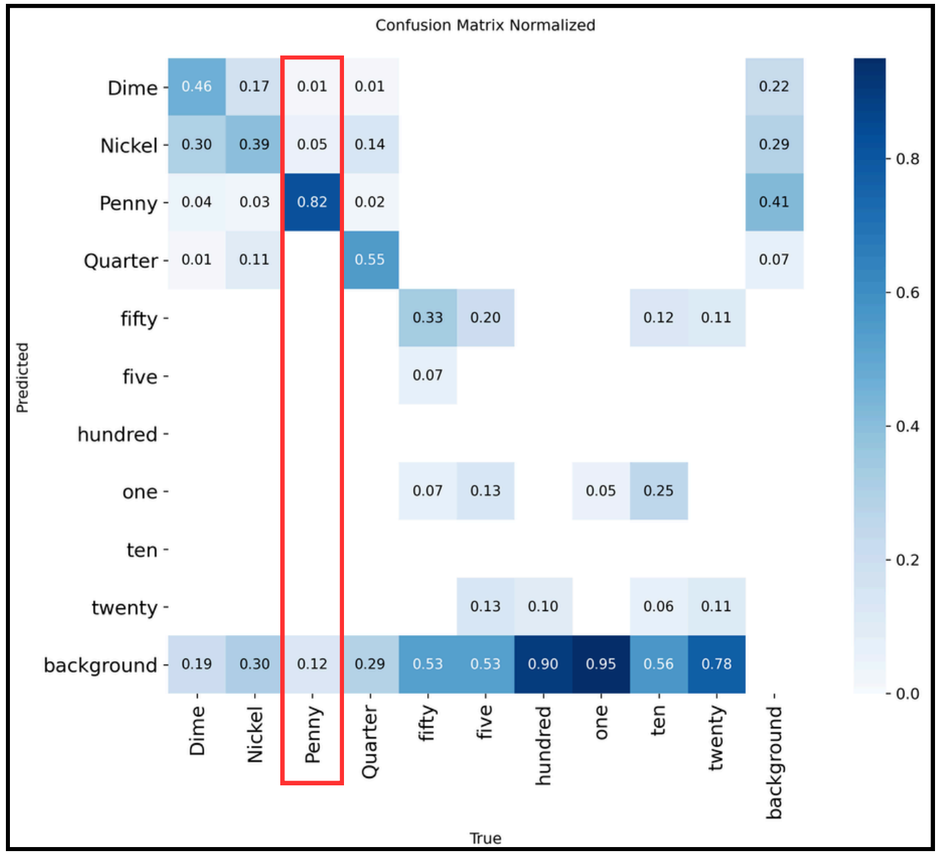
Voici une courbe F1score-confiance issue de détection multiclasse.

Sur la droite, on a la légendre pour savoir quelle courbe appartient à quelle classe.

Et le “0.40 at 0.146” à côté de “all classes” correspond au pic de la courbe (valeur qui nous intéresse pour déterminer le seuil optimal).

MATRICE DE CONFUSION NORMALISÉE

Dans votre dossier, vous avez aussi une matrice de confusion normalisée. Elle permet d'avoir une vision globale car au lieu de donner les chiffres, elle donne des pourcentages.



Comment la lire ?

La matrice de confusion normalisée permet de voir les pourcentages de détection pour chaque classe donc **elle se lit colonne par colonne**. Par exemple, pour les "Penny" de haut en bas :

- 1% ont été pris pour des "Dime"
- 5% ont été confondu avec des "Nickel"
- 82% des "Penny" ont bien été reconnu
- 12% n'ont pas du tout été détecté

On a bien 100% en faisant la somme donc c'est bon.

Voila, **on a finit avec les matrices de confusion et tout ce qui était associé**, on va continuer d'éplucher le dossier de résultats.

DOSSIER DE RÉSULTATS

labels.jpg

labels.jpg est une image avec quatre graphiques similaires à l'image ci-contre :

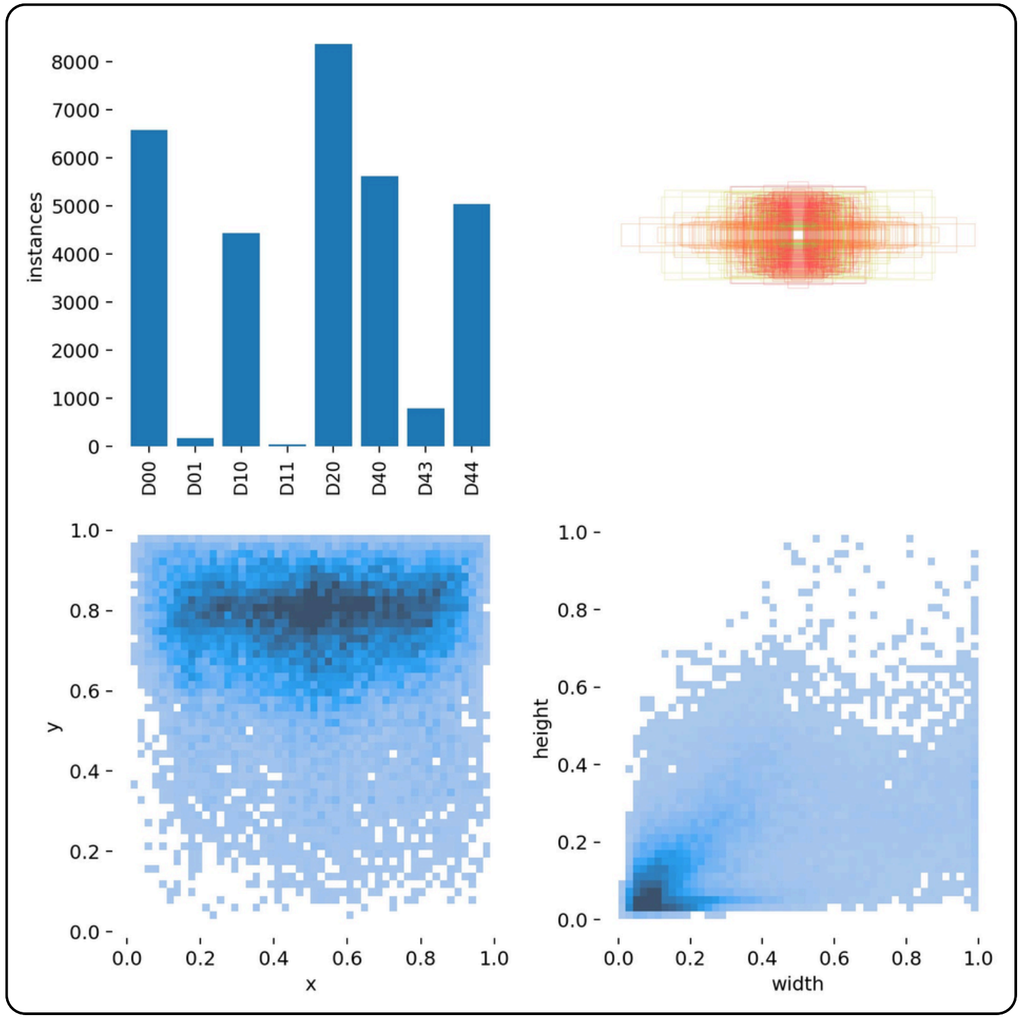
En haut à gauche : la répartition des différentes classes.

En haut à droite : superposition de toutes boites englobantes.

En bas à gauche : les coordonnées du centre des boites englobantes.

En bas à droite : les dimensions des boites englobantes.

Ce sont des statistiques sur les données annotées, **ce fichier est indépendant de l'entrainement**.



DOSSIER DE RÉSULTATS

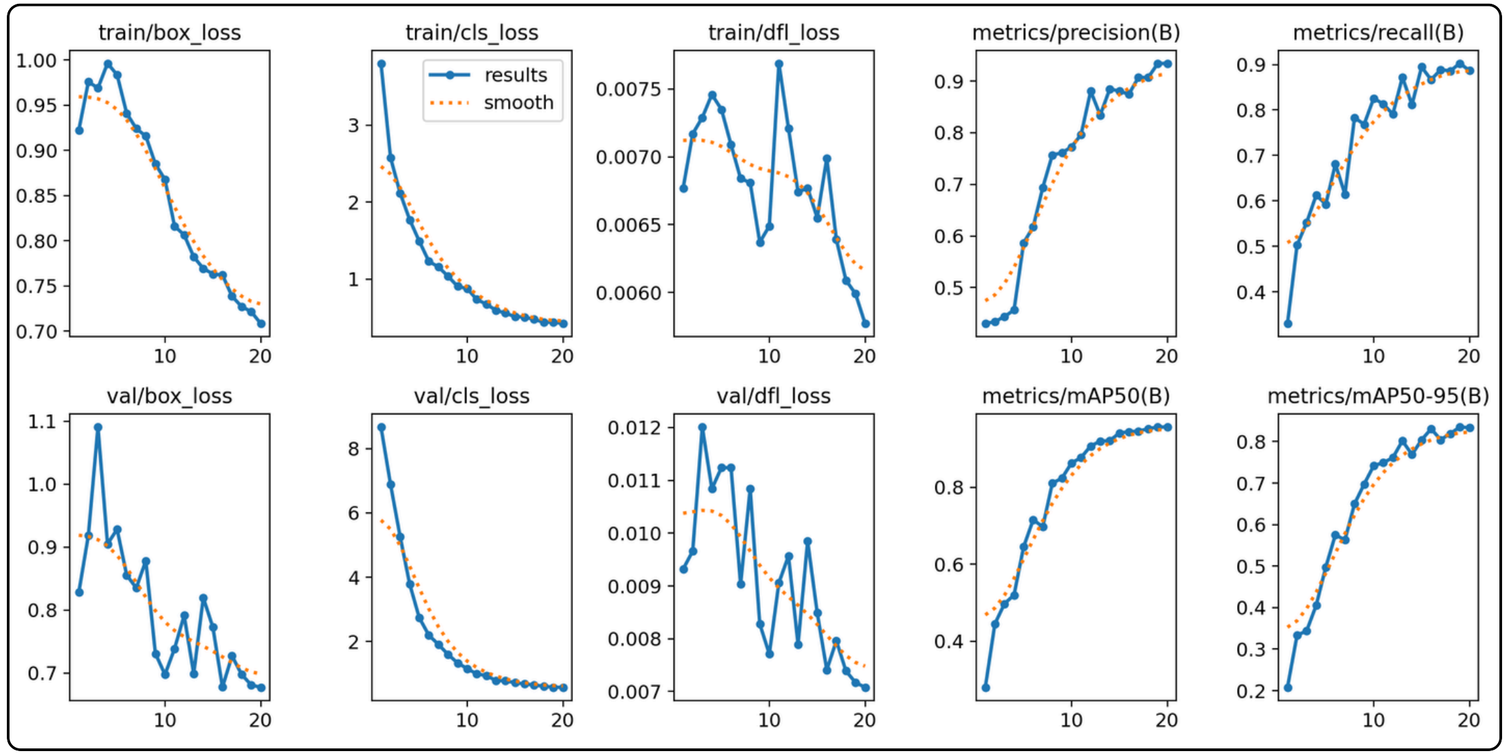
results.csv

results.png

Ces documents contiennent les mêmes métrique, l'un sous forme de tableau csv, l'autre sous forme de graphiques.

Pour comprendre sans rentrer vraiment dans la compréhension de chaque courbe :

- Les courbes qui descendent (loss) : le modèle apprend et fait moins d'erreurs
- Les courbes qui montent (precision, recall, mAP) : le modèle devient plus performant.



DOSSIER DE RÉSULTATS

results.csv results.png

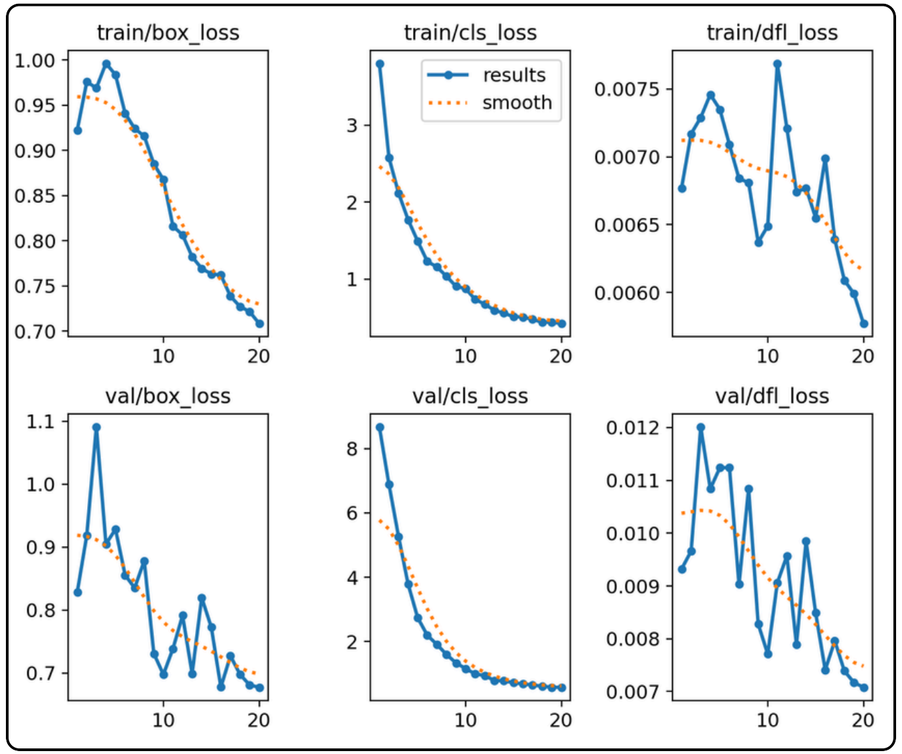
Ces 6 courbes correspondent à l'erreur entre les prédictions et la vérité terrain, en haut pour l'entrainement et en bas pour la validation.

Lors d'un entrainement, il y a un phénomène auquel il faut faire attention : le **surapprentissage** (overfitting). Pour faire simple, cela signifie que le modèle est en train d'apprendre **par coeur** les données d'entrainement.

Pourquoi cela est un problème ? Car le modèle ne saura plus se débrouiller sur des données différentes même si elles sont similaires.

Comment savoir si notre modèle a surappris ? Cela se voit avec les données de validation car le modèle ne surapprend pas sur celles là, donc quand vous regardez les courbes d'erreur, si celles de l'entrainement (celles en haut) continuent de descendre alors que celles de la validation (celles en bas) **remontent** au bout d'un moment, c'est signe que votre modèle surapprend.

Par exemple ci-contre, on voit que les courbes de validation ne font que décroitre, donc c'est un signe que notre modèle **généralise bien** pour l'instant.



DOSSIER DE RÉSULTATS

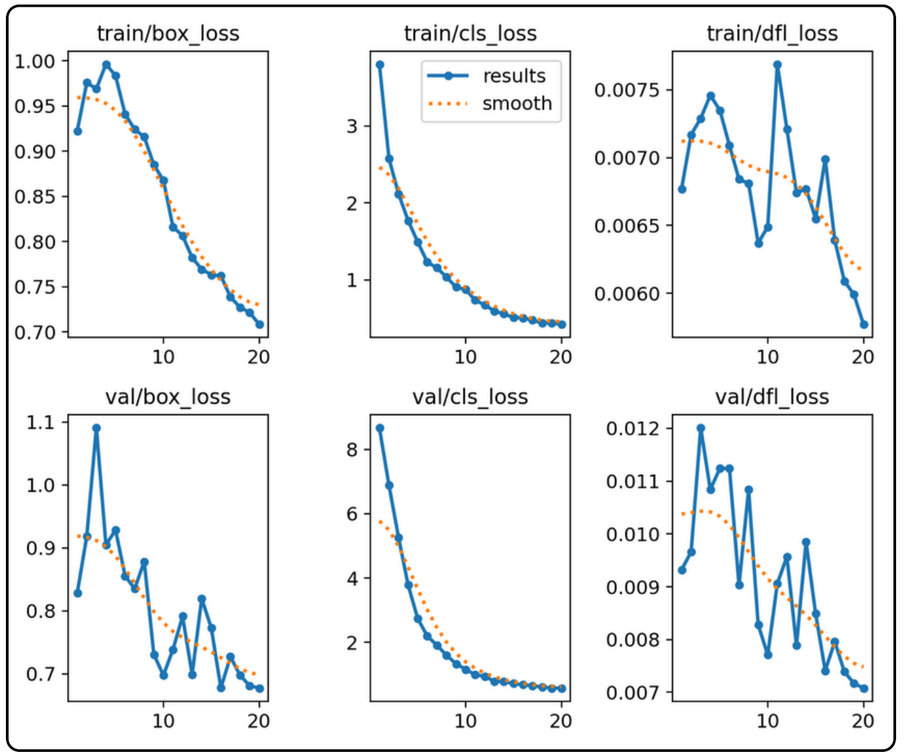
results.csv

results.png

Si on constate du surapprentissage, quelles sont les conséquences ?

Comme vu précédemment, YOLO enregistre deux modèles : best.pt et last.pt. Dans last.pt vous retrouverez votre modèle qui a surappris. En revanche, **dans best.pt, YOLO stocke les poids qui donnaient le meilleur résultat** combiné de l'entrainement et de la validation, donc **avant que le surapprentissage** entre en jeu.

Donc concrètement quel est le problème du surapprentissage ? On peut le voir comme une consommation de ressource inutile car le modèle va *trop loin.* Ainsi, pour remédier à cela, on peut utiliser le paramètre "patience" qui permet de dire au modèle "si après *x* nombre d'époques le modèle ne s'améliore plus (donc les poids de best.pt ne changent pas), alors arrête de t'entrainer" en passant x (un entier postitif) comme paramètre de patience.



ISDM
INSTITUT DE SCIENCE DES DONNÉES
MONTPELLIER

DOSSIER DE RÉSULTATS

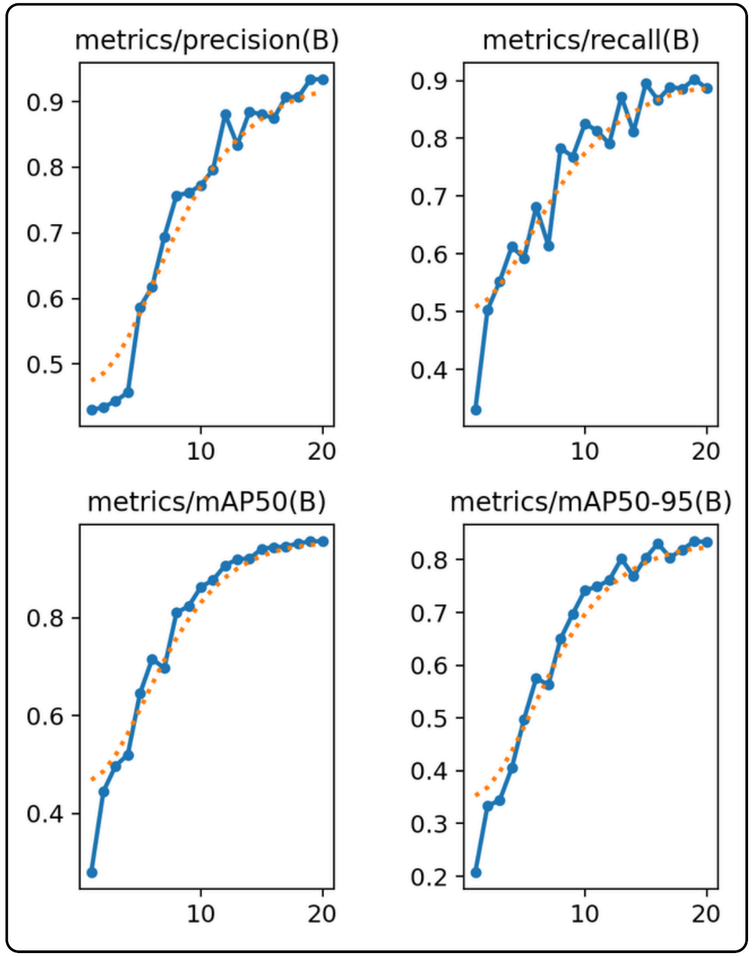
results.csv

results.png

Quant à ces courbes là, elles permettent de voir si notre modèle a atteint son plein potentiel.

Ces courbes permettent de visualiser **si le modèle s'améliore**, donc un bon indicateur que le modèle est arrivé en fin de course est de regarder si ces courbes ont atteint un **plateau**. Dans ce cas là, on saura que notre modèle a atteint son état le plus performant et n'a plus de perspective d'amélioration.

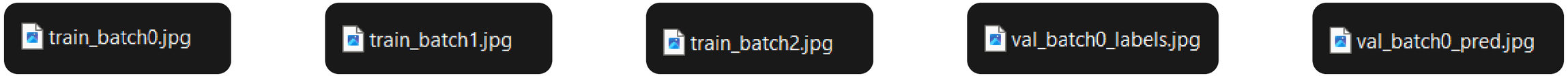
Par exemple, dans le cas ci-contre, on voit qu'après 20 époques, nos courbes sont encore en train de croitre donc on comprend que le modèle aurait besoin de plus d'époque pour se stabiliser.



ISDM
INSTITUT DE SCIENCE DES DONNÉES
MONTPELLIER

DOSSIER DE RÉSULTATS

Enfin, il reste tous les fichiers image ci :



Il peut y en avoir plus en fonction de la taille de votre dataset.

- “**train_batchx.jpg**” sont des exemples des images que le modèle reçoit pour son entraînement, donc des images labellisées
- “**val_batchx_labels.jpg**” c’est la même chose, donc c’est l’entrée du modèle mais cette fois pour faire de la validation.
- “**val_batchx_pred.jpg**” sont les images aves les labels en sortie du modèle donc des prédictions faites par le modèle. Donc vous pouvez y voir des boites pas forcément au bon endroit, avec ou non le bon label de classe etc, permet d’avoir un retour visuel de ce que fait le modèle.

CONCLUSION

Si les fichiers de résultats de votre modèle sont satisfaisant, vous pouvez ensuite **tester votre modèle** sur d'autres données, notamment votre dossier "test". Pour cela, vous devez d'abord charger votre modèle tout juste entrainé. Il se trouve dans le dossier runs/detect/train*x*/weights normalement et c'est le fichier **best.pt** :

```
from ultralytics import YOLO

model = YOLO("chemin/vers/votre/fichier/best.pt")
```

(n'hésitez pas à mettre le chemin absolu sauf si vous êtes sur le cluster)

Puis vous pourrez lancer un **processus de validation** mais en précisant bien que vous voulez que ce soit les données de **test** qui soient utilisée, de la manière suivante :

```
metrics = model.val(split="test")
```

Vous retrouverez ensuite les résultats dans runs/detect/valx, que vous pourrez analyser comme nous venons de le faire.

CONCLUSION

Si ces résultats vous conviennent, parfait. Sinon, il sera sûrement nécessaire d'entrainer à nouveau votre modèle. Peut être il serait judicieux de changer quelques paramètres d'entrainement, mais rajouter des époques peut suffire.
On va donc partir dans cette direction. Pour reprendre un entraînement afin de le prolonger, commencez par **charger les *derniers poids*** et non les meilleurs (très important!) :

```
from ultralytics import YOLO

model = YOLO("chemin/vers/votre/fichier/last.pt")
```

(n'hésitez pas à mettre le chemin absolu)

Lancez ensuite un entraînement comme au début mais avec le paramètre resume=True. Concernant le nombre d'époque, le nombre que vous mettrez là c'est le nombre total d'époques que le modèle fera, donc si votre last.pt est issu d'un entrainement de 100 époques, la commande ci-dessous permettra d'aller jusqu'à 150, donc d'en rajouter 50 et non en rajouter 150 ce qui nous emmenerait à 250.

```
results = model.train(epochs=150, resume=True)
```

ISDM
INSTITUT DE SCIENCE DES DONNÉES
MONTPELLIER

CONCLUSION

Enfin, si les résultats ne vous conviennent toujours pas, vous pouvez songer à **changer quelques paramètres** du modèle, donc pour ça nous avons fait une partie dédiée que vous trouverez juste à la suite de ce diapo.