université de technologie de Compiègne

Rapport de stage TN09

Analyse, recherche et intégration intelligence Artificiel (IA) dans les projets acoustiques

Chenyu SHAO - GI03

<u>Tuteur entreprise :</u> Mahmoud AMOU

<u>Suiveur UTC :</u> Mohamed SALLAK

Février - juillet 2024

Résumé technique

Ce rapport de stage présente une analyse approfondie de l'intégration de l'intelligence artificielle (IA) dans les projets acoustiques. Le stage, réalisé chez Gamba, avait pour objectif de développer et d'améliorer des algorithmes de machine learning, en particulier pour la détection et la classification d'objets dans des images. À travers ce travail, un nouvel outil a été conçu pour réentraîner les modèles existants en utilisant des données supplémentaires, ce qui a permis d'améliorer la précision et la robustesse des classificateurs. Le rapport détaille les méthodes utilisées, les défis rencontrés, les solutions apportées, ainsi que les résultats obtenus. Enfin, il propose des points d'amélioration pour optimiser davantage les modèles et les processus de mesure.

Mots-clés: Intelligence Artificielle (IA), Machine Learning, Détection d'objets, Classification, Python, C#, Vision par ordinateur

Remerciements

Je tiens tout d’abord à exprimer ma gratitude à M. Mahmoud AMOU, chef de projets logiciel, mon tuteur, qui m’a donné l’opportunité de travailler chez Gamba durant ces 6 derniers mois, ainsi que pour son accompagnement précieux tout au long de ce stage.

Je remercie également Mme Marion LORIN, assistante commerciale, pour avoir facilité mon intégration au sein de l’entreprise et m'avoir soutenu dans les tâches administratives.

Mes remerciements vont aussi à Mme Maryam Ly, coordinatrice de stages Génie Informatique, pour son soutien durant ma recherche de stage, ainsi qu'à M. Mohamed SALLAK, mon suiveur de stage, pour avoir pris le temps de s’assurer du bon déroulement de mon expérience.

Je souhaite également adresser un grand merci à ma famille et à mes amis pour leur soutien et leurs précieux conseils tout au long de ce projet.

Enfin, je tiens à remercier sincèrement l’ensemble des employés de Gamba pour leur accueil chaleureux et leur gentillesse durant ces 6 mois.

Sommaire

Acronymes

AP: Average Precision

CNN: Convolutional Neural Networks, les réseaux de neurones convolutifs

IoU: Intersection over Union

mAP: mean Average Precision

RCNN: Regions with CNN

YOLO: You Only Look Once

Liste des Figures

1 Introduction

La détection d'objets est une technique de vision par ordinateur permettant de localiser des objets dans une image ou une vidéo. La détection d'objets nous fournit des informations sur la boîte englobante de l'objet ainsi que sur la classification de cet objet.

La détection d'objets est une technologie fondamentale en vision par ordinateur, avec des applications variées telles que la surveillance, la conduite autonome, et bien d'autres domaines nécessitant une analyse visuelle détaillée. Les progrès continus dans ce domaine sont rendus possibles grâce à des algorithmes innovants et des méthodologies de plus en plus efficaces, permettant d'améliorer constamment la précision et la vitesse des systèmes de détection d'objets.

Dans ce rapport, une première partie est consacrée au contexte de la détection d’objets, notamment les technologie récentes, les différentes méthodes de détection et de classification existantes. Une autre partie est consacrée à la création d'un nouvel outil. Les résultats sont traités dans la dernière partie de ce rapport. Les pointes à améliorer de l'outil sont détaillés et des perspectives futures sont données dans la conclusion.

2 Détection d'objet et classification: historique

2.1 Définition et applications

Il existe des différences entre la classification d'objets, la détection d'objets et la segmentation d'objets. La classification d'objets est un type de reconnaissance d'image qui identifie le type d'objet présent dans une image. Ici, l'image complète est envoyée pour la classification, donc la sortie est une seule classe. La détection d'objets est un type de reconnaissance d'image utilisé pour identifier et localiser la présence d'un objet dans une image. Cela nous donne des informations sur la boîte englobante et la classe de l'objet. Ici, les résultats peuvent inclure plusieurs boîtes englobantes et classes. La segmentation d'objets est le type de reconnaissance d'image utilisé pour identifier et séparer les objets distincts dans une image au niveau des pixels. Ici, on obtient la forme exacte de l'objet détecté plutôt que seulement la boîte englobante. Bien que la segmentation d'image fournisse plus d'informations, la détection d'image reste le premier choix pour la plupart des applications de vision par ordinateur, car elle est moins coûteuse en termes de calcul.

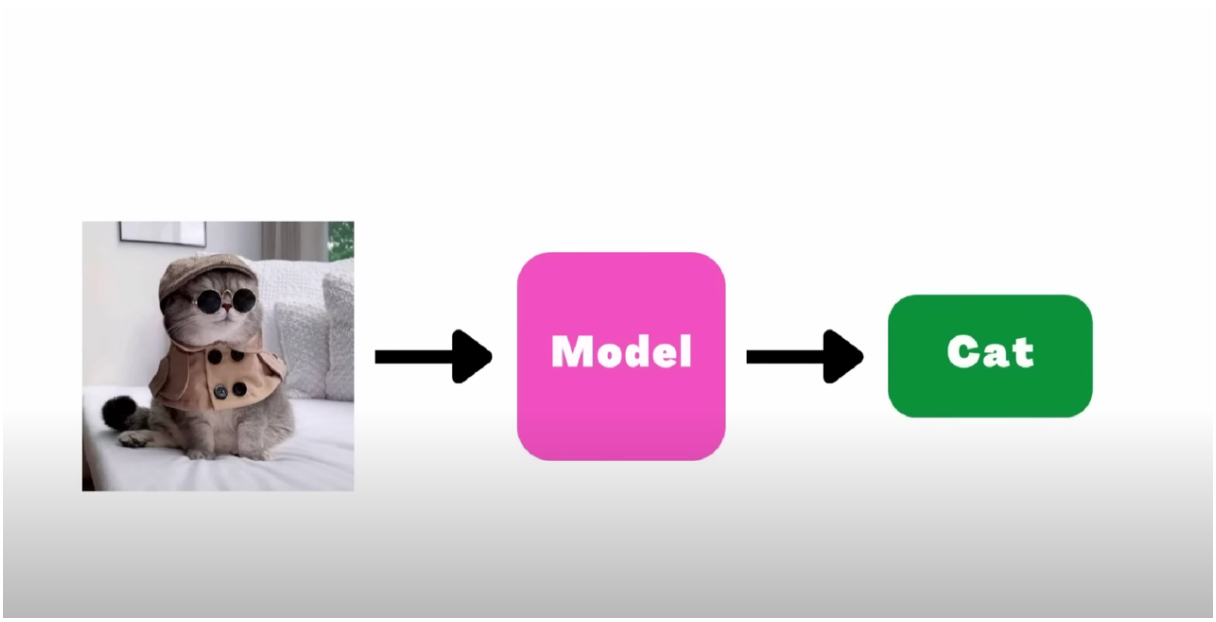

Figure 1 : la classification d'objets

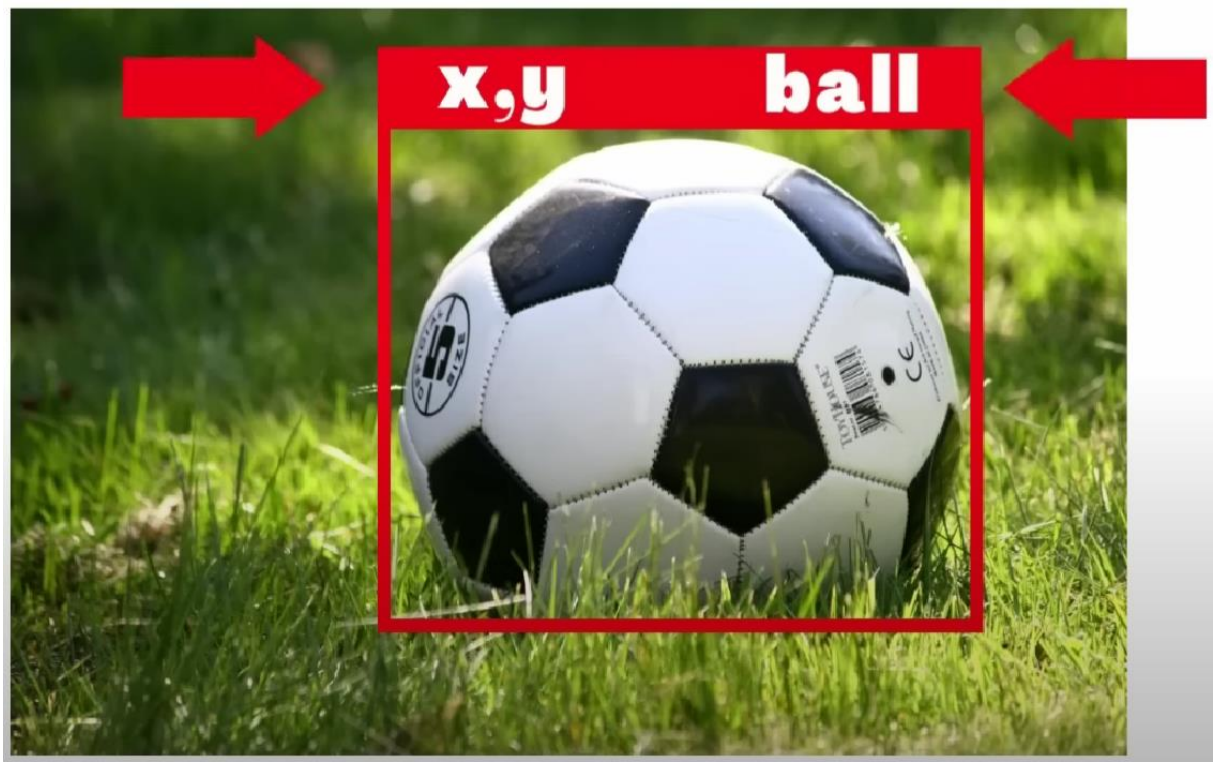

Figure 2 : la détection d'objets

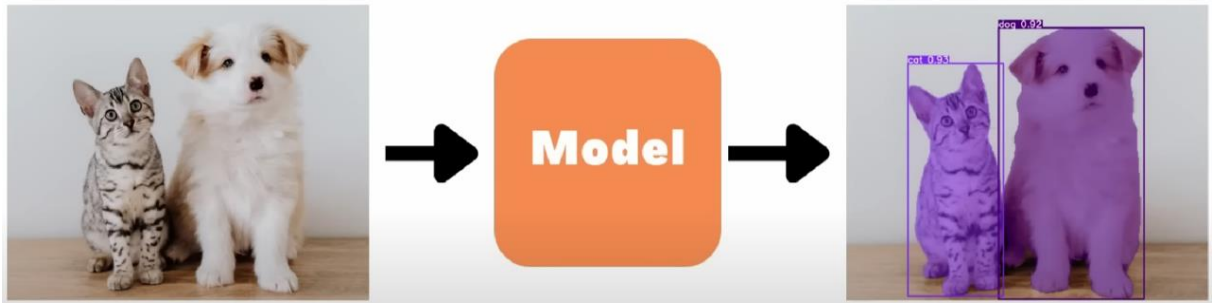

Figure 3 : la segmentation d'objets

2.2 Méthodes de détection

La détection d'objets a commencé dans les années 1970. Les chercheurs ont commencé à développer des méthodes automatisées pour la détection d'objets. Les premiers algorithmes de détection d'objets automatisés étaient basés sur des caractéristiques simples telles que les arêtes et les coins. La première véritable détection d'objets automatisée a été développée par Viola Jones en 2001. Cette méthode utilisait des fenêtres glissantes pour rechercher des caractéristiques de Haar, qui sont simplement des caractéristiques

rectangulaires. Cette méthode est devenue populaire car elle permettait une détection en temps réel. Pendant longtemps, cette méthode a été utilisée pour la détection des visages dans les smartphones et les appareils photo.

Le Histogramme des Gradients Orientés (Histogram of Oriented Gradients, HOG) était une autre méthode publiée en 2005, qui se concentrait sur la forme de l'objet. Elle fonctionnait en extrayant le gradient et l'orientation des arêtes. Cette méthode était principalement utilisée pour détecter les humains dans une image.

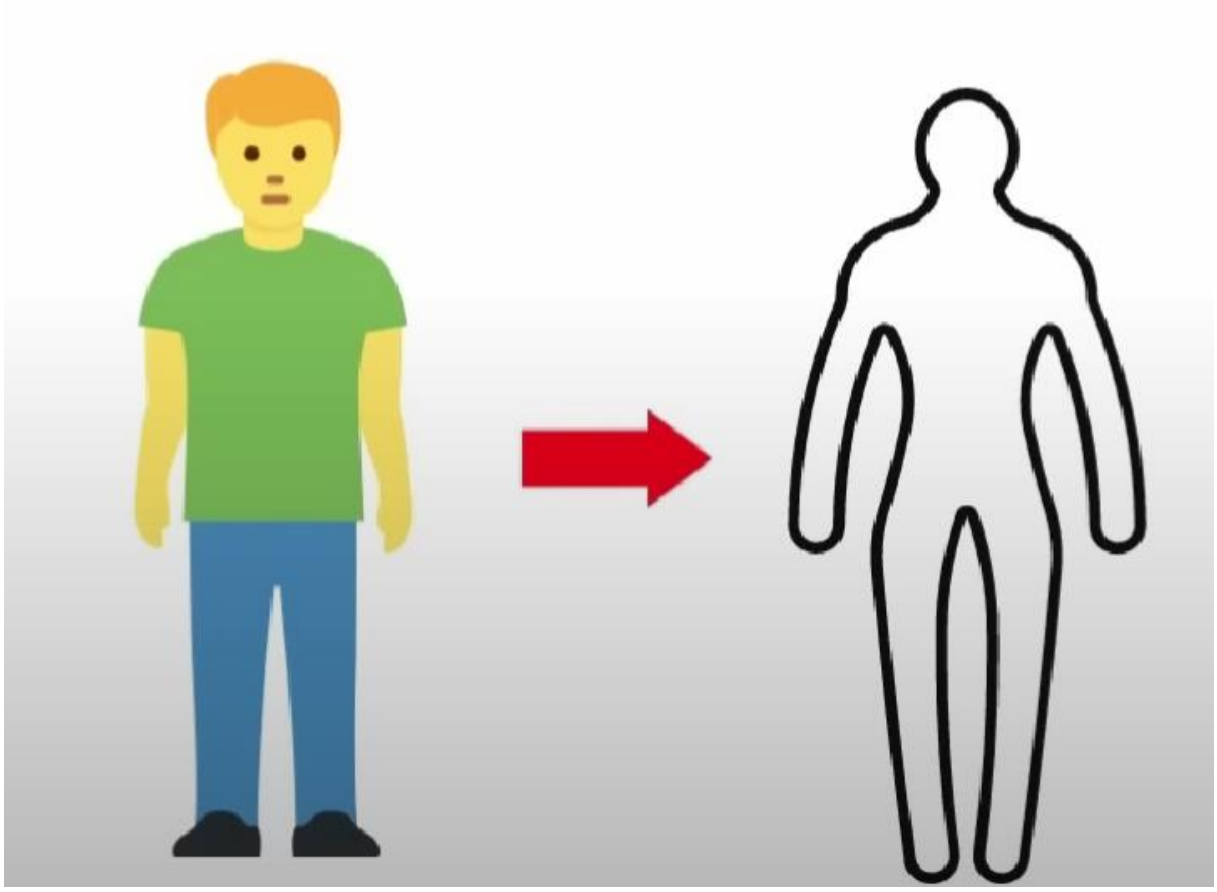

Figure 4 : Histogramme des Gradients Orientés

Ensuite est venue la révolution de l'IA avec la popularité croissante des réseaux neuronaux. Le véritable potentiel a été découvert avec les réseaux de neurones convolutifs (CNN). Cependant, cela a résolu le problème de classification et non de détection. Peu de temps après, le RCNN (Regions with CNN) est apparu, utilisant des régions sélectives pour appliquer les classificateurs. Cela donnait de bons résultats mais était lent. Ensuite, le Fast

RCNN et le Faster RCNN ont été développés, ils étaient plus rapides mais toujours pas en temps réel.

En 2015, la percée est venue avec le modèle YOLO (You Only Look Once). Celui-ci a surpassé tous les autres modèles et a permis une détection d'objets en temps réel. Cette méthode utilisait une approche différente, en effectuant un seul passage de l'image d'entrée pour faire des prédictions sur les objets. Les méthodes comme le RCNN utilisaient des propositions régionales pour effectuer plusieurs itérations pour la même image, tandis que YOLO y parvenait en une seule itération. Cela rend YOLO plus efficace, même si son architecture est basée sur des couches convolutives, semblable à celle de ses prédécesseurs.

2.3 Métriques d'évaluation des performances

Pour évaluer nos modèles de détection d'objets, il existe deux métriques principales que nous devons comprendre. L'une sert à évaluer la qualité de la localisation, et l'autre la qualité de la classification. La première est l'IoU (Intersection over Union), qui mesure la localisation, et la seconde est la mAP (Mean Average Precision), qui évalue la classification.

IoU (Intersection over Union) indique à quel point la boîte englobante prédite est proche de la boîte de vérité. C'est une valeur comprise entre 0 et 1. Si les boîtes se chevauchent parfaitement, cela signifie une détection parfaite avec un IoU de 1. Si les boîtes ont un certain chevauchement, alors la valeur sera comprise entre 0 et 1, en fonction du degré de chevauchement. Si les boîtes ne se chevauchent pas du tout, l'IoU est de 0. La valeur de l'IoU est calculée en prenant le rapport entre l'aire de l'intersection et l'aire de l'union des boîtes.

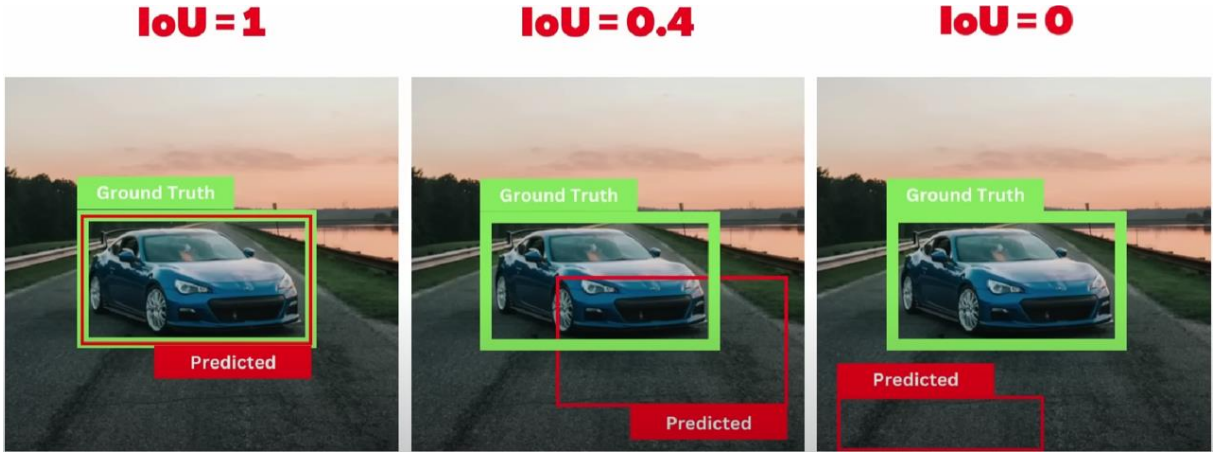

Figure 5 : IoU

mAP (Mean Average Precision) quantifie la précision du modèle en termes de classification. Pour comprendre l'Average Precision (AP), il est important de connaître la Matrice de Confusion, la Précision et le Rappel.

- Matrice de Confusion : C'est un tableau simple qui affiche les classes réelles d'un côté et les classes prédites de l'autre. À partir de cette matrice, on peut calculer la Précision et le Rappel.
- Précision (Precision) : C'est la proportion de vrais positifs parmi toutes les prédictions positives. En termes mathématiques, la Précision est égale aux vrais positifs divisés par la somme des vrais positifs et des faux positifs.
- Rappel (Recall) : C'est la proportion de vrais positifs parmi tous les exemples réellement positifs. Le Rappel est égal aux vrais positifs divisés par la somme des vrais positifs et des faux négatifs.

Étant donné que ces deux métriques fournissent des informations précieuses, on les combine en une seule appelée Average Precision (AP), qui correspond à l'aire sous la courbe Précision-Rappel. Cette courbe optimise les effets des deux métriques et nous donne une meilleure idée de la précision globale du modèle. L'Average Precision est calculée pour une seule classe. Si nous avons plusieurs classes, alors la moyenne de ces valeurs de Précision est appelée mean Average Precision (mAP).

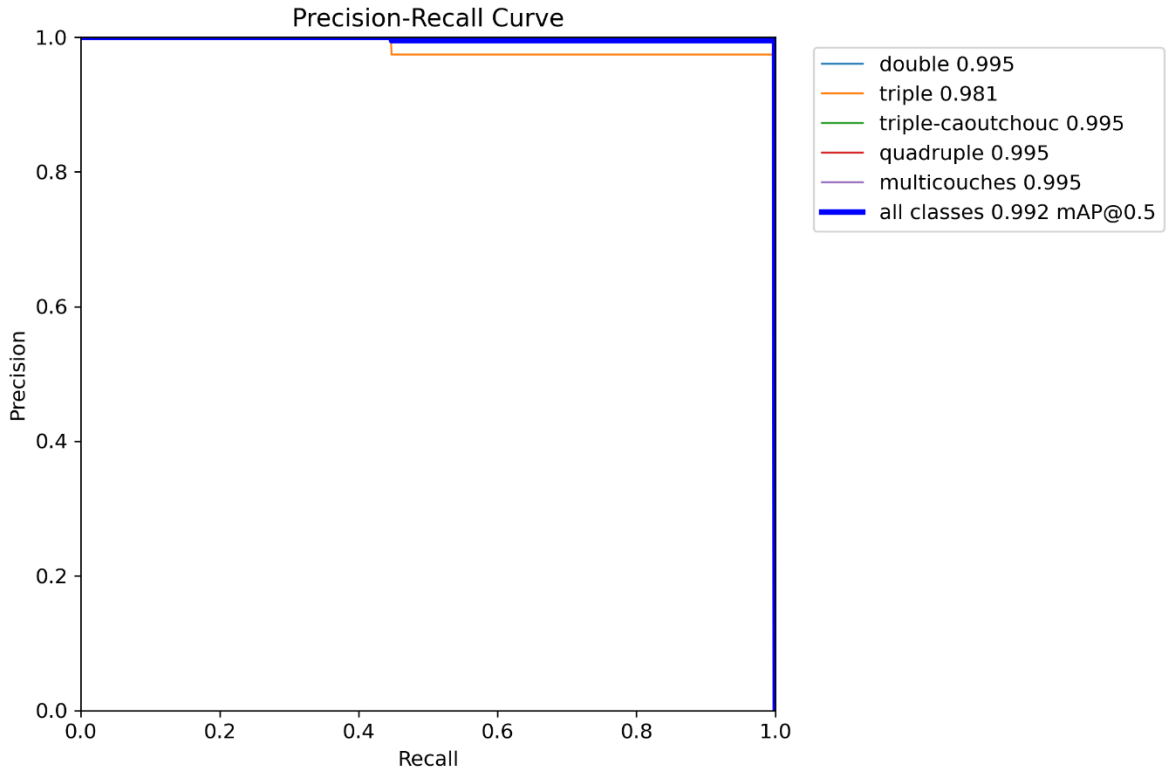

Figure 6 : la courbe Précision-Rappel

3 Développement

Étant donné que l'étiquetage manuel de chaque type de système d'une coupure et la répétition de cette opération prennent du temps, un nouvel outil est nécessaire et présenté dans cette section, y compris les différentes étapes de développement.

3.1 Mission de stage et description des projets

Dans le cadre de ce stage, ma mission principale était de concevoir et de développer un algorithme avancé destiné à améliorer les capacités d'apprentissage automatique en ce qui concerne la classification de divers types de fenêtres. Plus précisément, il s'agissait de créer une solution algorithmique capable de réentraîner les modèles de machine learning existants, en leur fournissant des informations supplémentaires sur les caractéristiques distinctives des différents types de fenêtres. L'objectif était d'améliorer la précision et la robustesse des classificateurs, en optimisant leur capacité à différencier les variations subtiles entre les différentes catégories de fenêtres. Cela impliquait une analyse approfondie des données, le choix des meilleures techniques de machine learning, et la mise en œuvre d'un processus de réentraînement itératif pour garantir que les modèles puissent s'adapter aux nouvelles données tout en conservant une performance optimale.

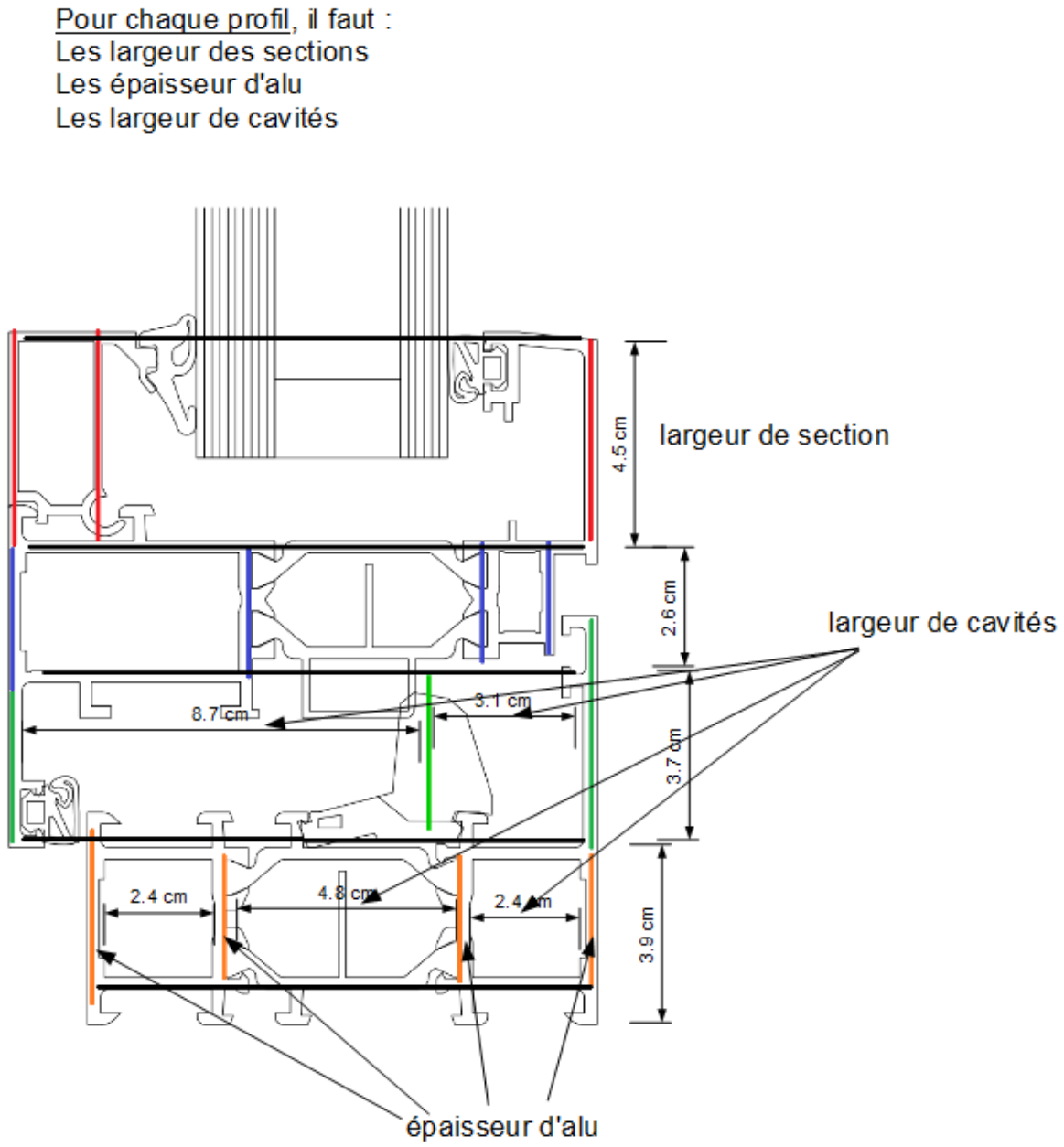

Figure 7 : Mission de stage et description des projets

3.2 Processus de réalisation

La première étape consiste à télécharger Python. Nous pouvons télécharger la dernière version, mais cela n'est pas recommandé, car elle pourrait contenir des erreurs ou nécessiter des corrections de bugs. Nous allons donc accéder à la section des anciennes versions où nous

pouvons voir que Python 3.11 sera pris en charge jusqu'en 2027 et Python 3.10 jusqu'en 2026. Je n'ai pas choisi la version la plus récente, mais plutôt celle juste en dessous, la version 3.10.

Il est possible d'avoir le même projet fonctionnant sur différentes versions de Python, ce qui nous permet de voir laquelle est la plus stable et laquelle nous donne les meilleurs résultats.

Lorsque nous travaillons avec la détection d'objets, il y a de nombreuses bibliothèques qui fonctionnent en arrière-plan. Les installer une par une peut prendre beaucoup de temps. Une méthode simple pour éviter cela est d'utiliser un fichier de dépendances, souvent appelé ‘requirements.txt’. Nous pouvons copier un fichier de ce type déjà existant, ou créer le nôtre.

```
cvzone==1.5.6

ultralytics==8.1.29

hydra-core>=1.2.0
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python==4.5.4.60
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
```

tqdm>=4.64.0

filterpy==1.4.5

scikit-image==0.19.3

lap==0.4.0

Voici comment cela fonctionne :

Nous plaçons simplement le fichier ‘requirements.txt’ dans notre projet. Ce fichier liste toutes les bibliothèques et versions nécessaires pour que notre projet fonctionne correctement. Une fois le fichier en place, nous recevrons une notification indiquant que certains packages requis ne sont pas installés. Il nous suffira de cliquer sur "Installer les dépendances", et toutes les bibliothèques nécessaires seront automatiquement installées. Cette méthode standardise le processus et nous fait gagner du temps, en veillant à ce que toutes les dépendances soient correctement gérées.

3.3 Défis et solutions

Au cours de mon stage, j'ai été confronté à plusieurs défis techniques liés à l'intégration de scripts Python dans un projet C# sous Visual Studio 2019. Mon objectif était d'exécuter des scripts Python via des commandes C# tout en assurant la gestion des dépendances Python nécessaires.

Solution : Exécution de scripts Python à partir de C#

Dans Visual Studio 2019, il est possible d'exécuter un fichier Python (.py) à partir d'un programme C#. Voici les étapes principales que j'ai suivies pour implémenter cette solution :

1. Installation de l'environnement Python :
   J'ai d'abord veillé à ce que Python soit installé sur le système et ajouté au PATH des variables d'environnement.
2. Création d'un projet C# :
   J'ai créé un nouveau projet de console en C# dans Visual Studio 2019.
3. Ajout et exécution d'un script Python :
   Après avoir ajouté le fichier Python au projet, j'ai utilisé la classe System.Diagnostics.Process pour exécuter ce script depuis le programme C#.
   Le code suivant montre comment exécuter un script Python à partir d'un programme C# :

```
static void ExecutePythonScript(string pythonFilePath)
    {
      // Créer des informations de démarrage du processus
      ProcessStartInfo start = new ProcessStartInfo();
      start.FileName = "python";
      start.Arguments = pythonFilePath;
      start.UseShellExecute = false;
      start.RedirectStandardOutput = true;
      start.RedirectStandardError = true;
      start.CreateNoWindow = true;
      // Démarrer le processus
      using (Process process = Process.Start(start))
      {
        // Lire la sortie standard
        using (System.IO.StreamReader reader = process.StandardOutput)
        {
          string result = reader.ReadToEnd();
```

```
            Console.Write(result);
        }
        // Lire l'erreur standard (si disponible)
        using (System.IO.StreamReader reader = process.StandardError)
        {
            string error = reader.ReadToEnd();
            if (!string.IsNullOrEmpty(error))
            {
                Console.Write(error);
            }
        }
    }
}
```

4. Gestion des dépendances Python :
   J'ai également ajouté des commandes C# pour mettre à jour pip et installer les dépendances nécessaires avant l'exécution du script Python. Voici comment cela a été réalisé :

```
static void UpdatePipAndInstallPackages()
    {
        // Mettre à jour l'environnement pip
        RunCommand("python -m pip install --upgrade pip");
        // Dictionnaire des noms et versions des packages
        var packages = new (string Name, string Version)[]
        {
            ("ultralytics", "8.1.29"),
```

```
            ("opencv-python", "4.5.4.60"),
            ("cvzone", "1.5.6"),
            ("numpy", "1.26.4")
        };
        foreach (var package in packages)
        {
            // Construire la commande d'installation avec version
            string command = $"pip install {package.Name}=={package.Version}";
            RunCommand(command);
        }
    }
```

5. Édition du script Python avant exécution :
   Afin d'apporter des modifications au script Python avant son exécution, j'ai implémenté une méthode en C# permettant de spécifier une ligne et de modifier son contenu :

```
static void UpdateFileNameImage(string filePath, string newContent)
    {
        // Lire le contenu du fichier
        string[] lines = File.ReadAllLines(filePath);
        string searchString = "img = ";
        string replacementString = newContent;
        using (StreamWriter writer = new StreamWriter(filePath))
        {
            foreach (string line in lines)
            {
```

```
            if (line.StartsWith(searchString))
            {
                writer.WriteLine(replacementString);
            }
            else
            {
                writer.WriteLine(line);
            }
        }
    }
}
```

6. Utilisation de chemins relatifs :
   J'ai découvert que les fichiers utilisés par le programme doivent être placés dans des dossiers spécifiques du projet, comme un dossier nommé Resources. Pour lire ces fichiers dans le code, j'ai utilisé des chemins relatifs, en m'assurant que les fichiers étaient correctement copiés dans le répertoire de sortie à l'exécution.
   Exemple de code pour accéder à un fichier avec un chemin relatif :

```
// Construire un chemin relatif
string relativePath = Path.Combine("Ressources", "Yolo.py");
// Obtenir le chemin absolu
string pythonFilePath = Path.GetFullPath(relativePath);
// Mettre à jour l'environnement pip et installer les packages Python requis
UpdatePipAndInstallPackages();
// Màj nom fichier image en modifiant le script Python
// Le nom du répertoire doit répondre aux exigences de la ligne de commande
// "img = cv2.imread(\"chemin/absolu/image.xxx\")"
UpdateFileNameImage(pythonFilePath, "img =
cv2.imread(\"D:/ChenyuSHAO/Image4.png\")");
```

Grâce à l'utilisation des techniques mentionnées ci-dessus, j'ai pu intégrer efficacement l'exécution de scripts Python dans un projet C# sous Visual Studio 2019. La gestion des dépendances, l'édition de scripts à la volée et l'utilisation de chemins relatifs ont été des solutions clés pour surmonter les défis rencontrés lors de ce projet.

4 Résultats et analyse

4.1 Performance de détection et évaluation de classification

Dans cette section, je présente une analyse détaillée des performances du modèle de classification à travers plusieurs mesures d'évaluation, incluant la matrice de confusion normalisée, la courbe de précision-confiance, la courbe de précision-rappel, et la courbe de rappel-confiance.

1. Matrice de Confusion Normalisée

La matrice de confusion normalisée (Figure 8) illustre la répartition des prédictions correctes et incorrectes à travers les différentes classes. Les résultats montrent que le modèle a une excellente capacité à distinguer certaines classes comme "double", "triple" et "multicouches" avec une précision de 100%. Cependant, pour la classe "quadruple", nous observons une certaine confusion avec "quadruple-caoutchouc-1", ce qui est mis en évidence par une faible proportion de prédictions correctes (3%).

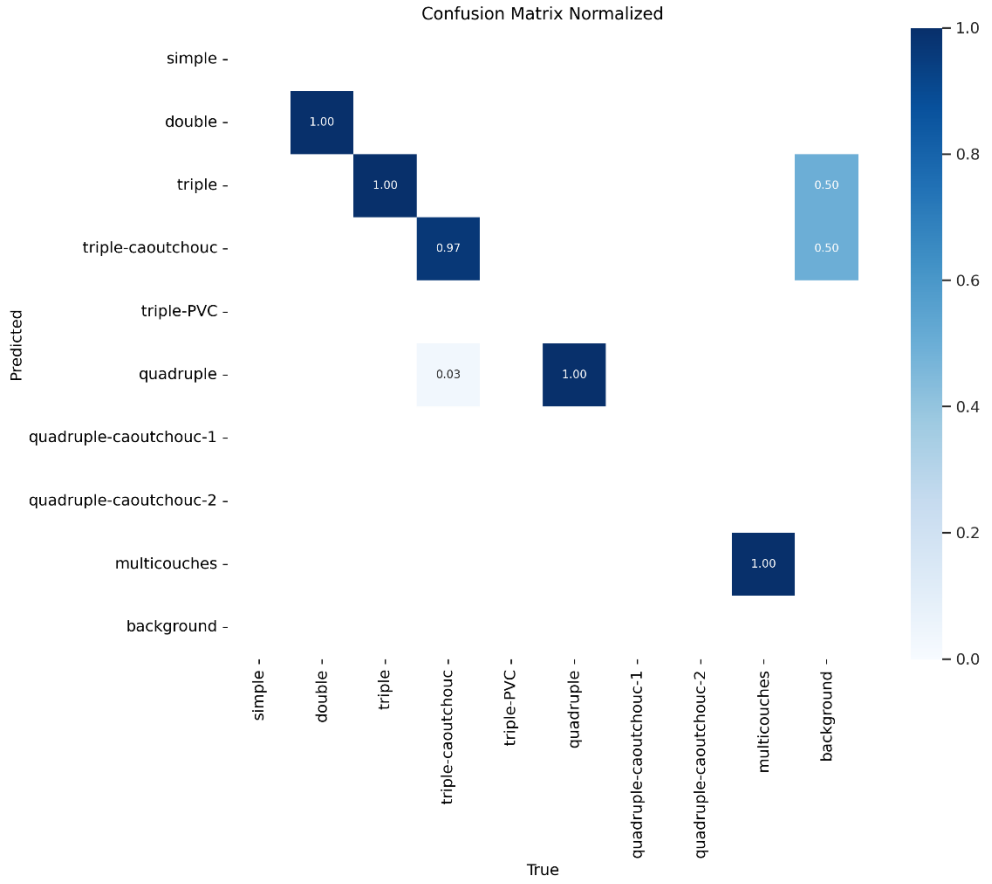

Figure 8 : La matrice de confusion normalisée

2. Courbe de Précision-Confiance

La courbe de précision-confiance (Figure 9) évalue la précision du modèle en fonction de la confiance des prédictions. Les courbes montrent que pour la plupart des classes, la précision est élevée même pour des niveaux de confiance inférieurs, avec une précision de 0.977 pour l'ensemble des classes. La courbe "triple-caoutchouc" se distingue légèrement en nécessitant une plus grande confiance pour atteindre des niveaux de précision élevés, indiquant une légère incertitude dans la classification de cette classe spécifique.

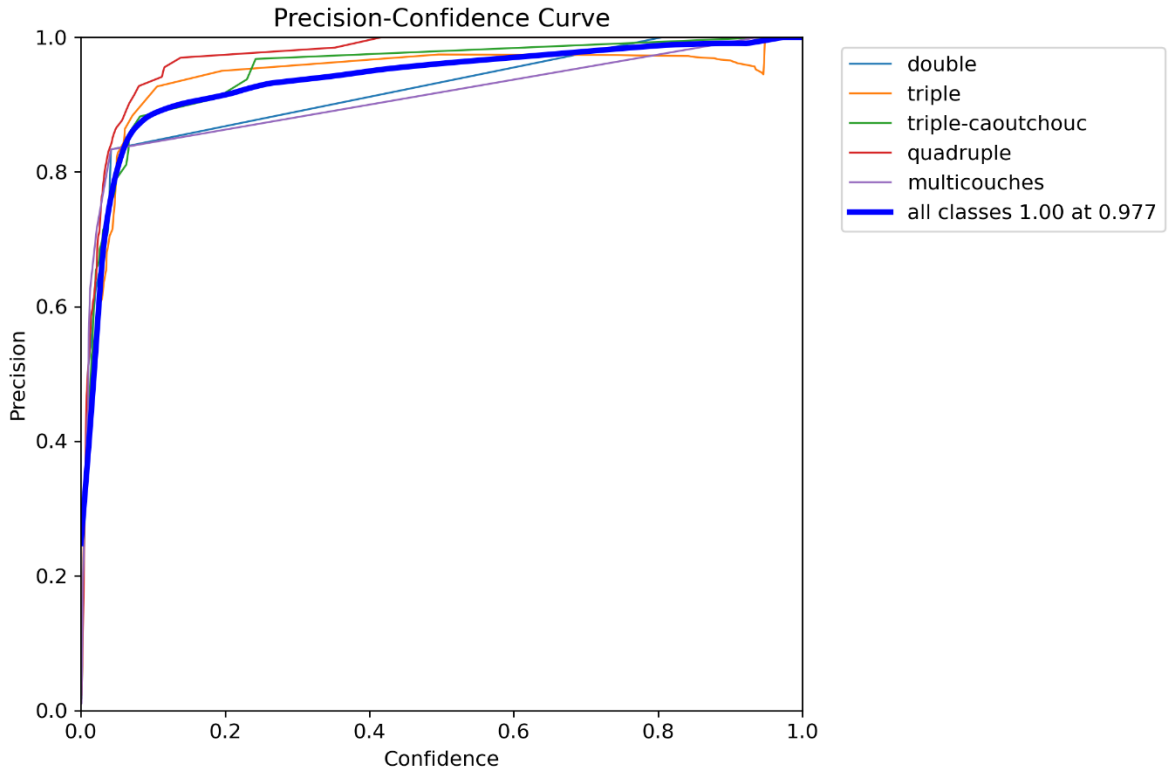

Figure 9 : La courbe de précision-confiance

3. Courbe de Précision-Rappel

La courbe de précision-rappel (Figure 10) montre que le modèle maintient une précision presque parfaite pour toutes les classes lorsque le rappel augmente, avec une moyenne de 0.992 pour l'ensemble des classes. Toutefois, on note une légère diminution de précision pour la classe "triple" (0.981), ce qui pourrait indiquer des erreurs potentielles lorsque le modèle tente de rappeler toutes les instances positives de cette classe.

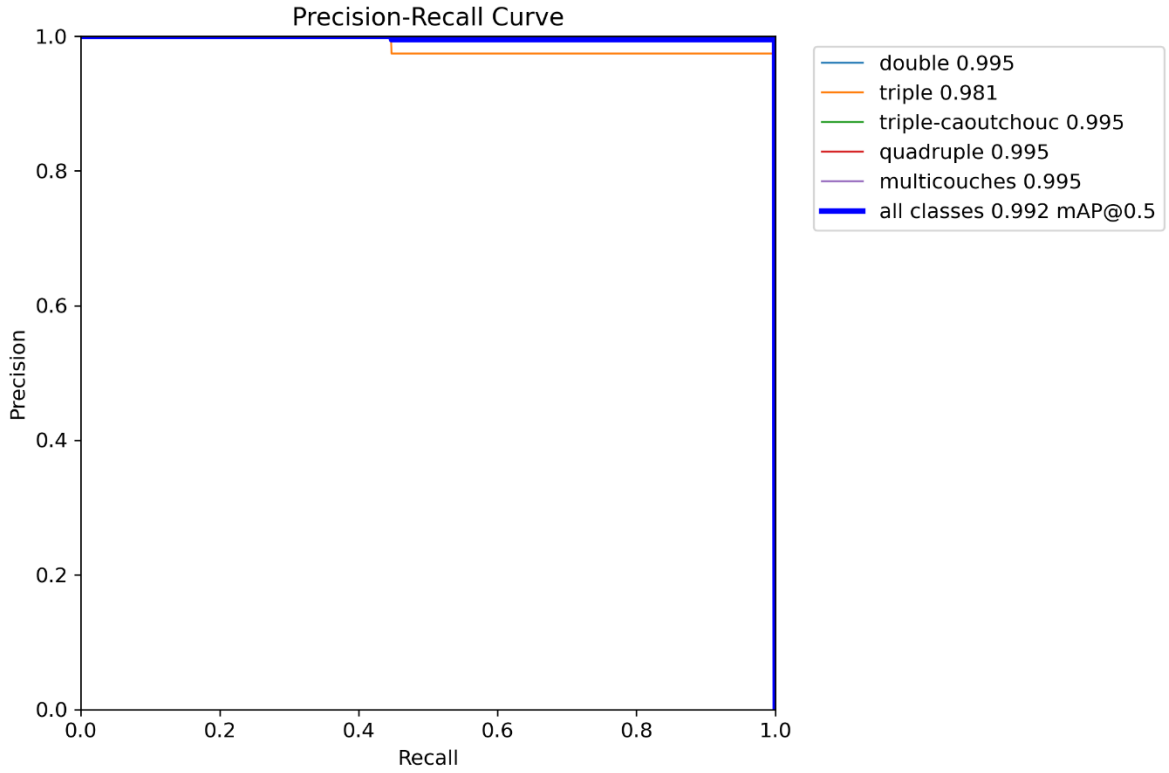

Figure 10 : La courbe de précision-rappel

4. Courbe de Rappel-Confiance

La courbe de rappel-confiance (Figure 11) analyse la capacité du modèle à rappeler correctement les instances à différents niveaux de confiance. Les résultats montrent que, pour atteindre un rappel élevé, le modèle doit être extrêmement confiant dans ses prédictions. En particulier, la classe "triple" montre une diminution significative du rappel à des niveaux de confiance plus élevés, suggérant que le modèle peut être prudent dans la classification de cette classe spécifique, sacrifiant le rappel pour éviter les fausses alarmes.

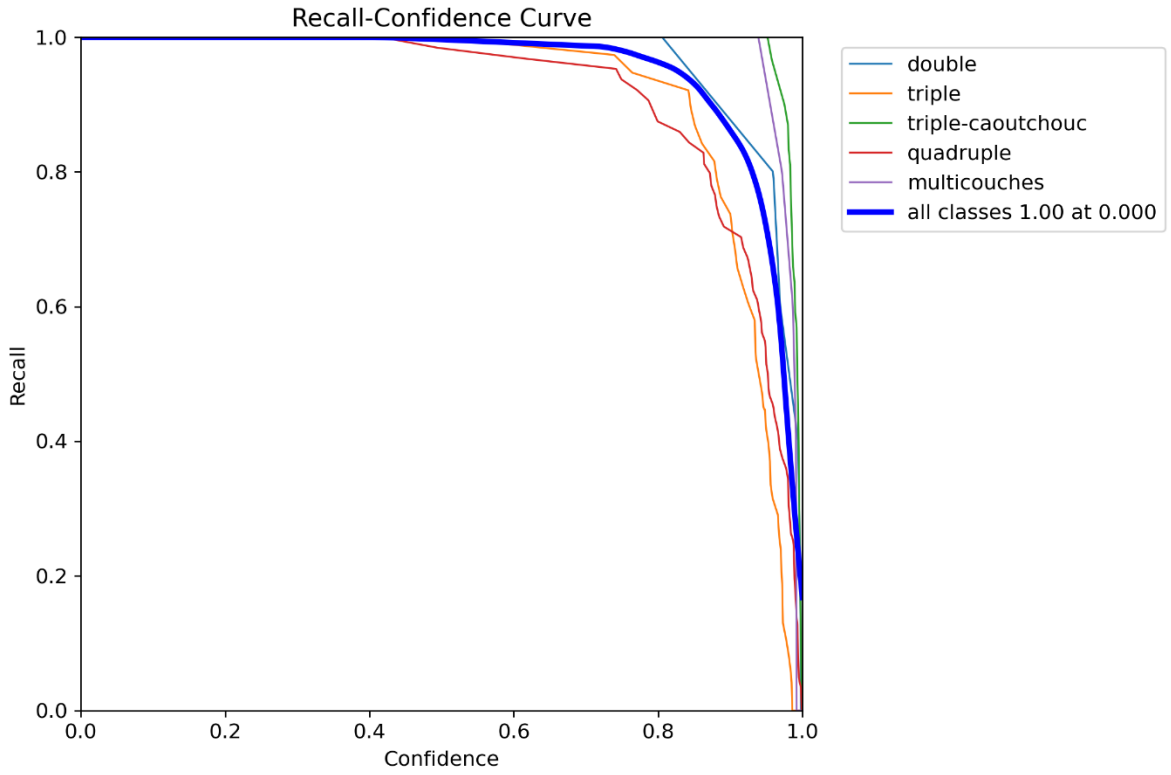

Figure 11 : La courbe de rappel-confiance

4.2 Points à améliorer

Dans le cadre de ce projet, un autre besoin identifié est de mesurer les dimensions exactes de chaque structure après leur reconnaissance. Actuellement, les dimensions obtenues

sont exprimées en pixels. Il est nécessaire de convertir ces dimensions en tailles absolues. Mon idée est d'utiliser la fonctionnalité de mesure d'un logiciel de dessin. Pour cela, j'ajouterais une échelle dans l'image d'entrée et je formerais un modèle supplémentaire pour reconnaître cette échelle. Une fois l'échelle détectée et isolée, le modèle pourrait extraire les chiffres indiqués et les associer aux pixels correspondants. Grâce à ce rapport, il serait possible de calculer toutes les dimensions nécessaires sur l'ensemble du plan. Il est essentiel de développer un processus fiable pour convertir les dimensions exprimées en pixels en tailles absolues. Cela nécessitera l'intégration d'un système de mesure précis dans les images d'entrée, ainsi que la formation d'un modèle performant pour reconnaître et interpréter les échelles présentes sur les dessins.

Par ailleurs, une fois la structure reconnue, il est également souhaitable, dans la mesure du possible, de former un modèle pour identifier les dimensions de chaque couche. Les résultats obtenus avec le modèle que j'ai essayé ne sont pas très satisfaisants. Le modèle actuel présente une précision de seulement 50 %, ce qui est insuffisant pour des applications industrielles. Pour remédier à cela, il serait pertinent d'augmenter le nombre d'exemples de formation, en particulier pour les formes simples qui semblent poser des défis particuliers. En outre, l'exploration de nouvelles techniques de machine learning, telles que des algorithmes de reconnaissance de formes plus avancés ou des architectures de réseaux neuronaux plus adaptées, pourrait offrir des pistes prometteuses pour améliorer la précision du modèle.

5 Conclusion

Au terme de ce stage chez Gamba, j'ai pu développer des compétences techniques avancées dans l'intégration de l'intelligence artificielle, en particulier dans le domaine de la détection et de la classification d'objets. Le projet m'a permis de concevoir un algorithme performant pour améliorer les capacités de machine learning, tout en relevant des défis techniques complexes tels que l'exécution de scripts Python dans un environnement C#. Les résultats obtenus démontrent une amélioration significative de la précision des modèles utilisés, bien que des axes d'amélioration subsistent, notamment en ce qui concerne la conversion des dimensions en tailles absolues et l'amélioration des performances de classification pour certaines catégories d'objets.

Cette expérience a non seulement renforcé mes compétences en développement et en analyse, mais elle m'a également permis de mieux comprendre les enjeux liés à l'application des technologies d'IA dans un cadre industriel. Ce stage a ainsi posé les bases d'une amélioration continue des systèmes, ouvrant la voie à des applications encore plus robustes et précises.

Bibliographie

[1] Kaehler, Adrian, and Gary Bradski. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, 2016.

[2] Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. “Face recognition with local binary patterns.” European conference on computer vision. Springer Berlin Heidelberg, 2004.

[3] LabelImg, https://github.com/HumanSignal/labelImg.

[4] Csurka, Gabriella, et al. “Visual categorization with bags of keypoints.” Workshop on statistical learning in computer vision, ECCV. Vol. 1. No. 1–22. 2004.

[5] Forsyth, D., and J. Ponce. Computer Vision: A Modern Approach. Engle-wood Cliffs, NJ: Prentice-Hall, 2003.

[6] Hinton, G. E., S. Osindero, and Y. Teh. “A fast learning algorithm for deep belief nets,” Neural Computation 18 (2006): 1527–1554.

[8] Johnson, D. H. “Gauss and the history of the fast Fourier transform,” IEEE Acoustics, Speech, and Signal Processing Magazine 1 (1984): 14–21.

[9] Zivkovic, Z. “Improved adaptive Gaussian mixture model for background subtraction,” International Conference Pattern Recognition, UK, August 2004.

Presentation du Groupe Gamba

Le Groupe Gamba est une entreprise spécialisée dans l'ingénierie acoustique et vibratoire, fondée en 1976 à Toulouse par René Gamba, un jeune docteur en mécanique des fluides. Au fil des ans, l'entreprise a évolué et s'est structurée pour devenir un acteur majeur dans son domaine, avec une présence nationale et internationale.

Aujourd'hui, le Groupe Gamba emploie plus de 75 collaborateurs et se distingue par son approche axée sur l'évolution continue, tant sur le plan technique que géographique. Cette dynamique est renforcée par une politique d'ouverture de l'actionnariat vers les salariés, favorisant ainsi l'engagement et l'épanouissement au sein de l'entreprise.

Le groupe est engagé dans une démarche de responsabilité sociale et environnementale, déclinée en quatre axes principaux : les hommes et les territoires, l'écoconception et l'empreinte, la gouvernance et l'éthique, ainsi que la qualité et les certifications. Cette démarche se traduit notamment par la construction de bureaux à énergie positive, comme le siège social situé à Labège.

Avec des bureaux en France et à l'étranger, notamment au Brésil, le Groupe Gamba est en mesure d'accompagner ses clients sur des projets variés, allant de l'acoustique des bâtiments à la gestion des vibrations dans les transports, en passant par l'acoustique des parcs éoliens.

Enfin, le groupe valorise la formation continue de ses collaborateurs et propose également des formations spécialisées dans ses domaines d'expertise, contribuant ainsi à l'évolution des compétences au sein de l'entreprise et de ses partenaires.

Annexes

Annexe 1 : Images d'exemple du projet

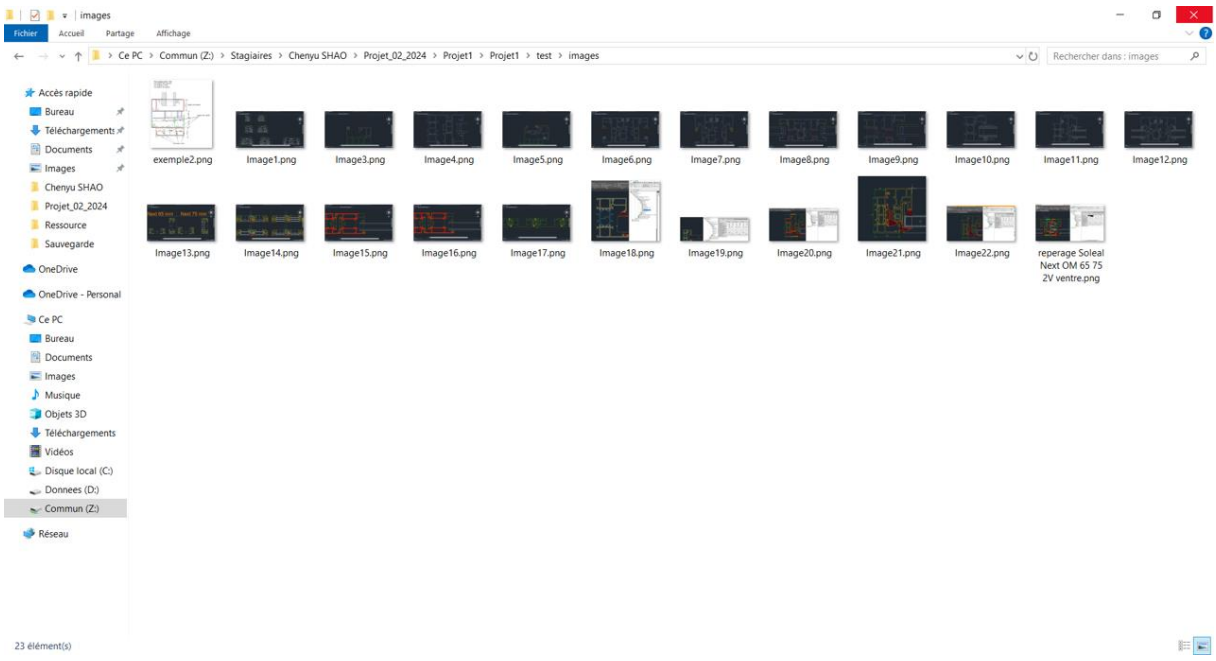

Figure 12 : Les images d'exemple du projet

Annexe 2 : Analyse du modèle

Figure 13 : La première partie du modèle