



المدرسة الوطنية للعلوم التطبيقية بتطوان
Ecole Nationale des Sciences Appliquées de Tétouan

UNIVERSITÉ ABDELMALEK ESSAÂDI
ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES DE
TÉTOUAN

FILIÈRE : BIG DATA & IA

Rapport du Projet Deep Learning

SUJET : SEGMENTATION D'ENVIRONNEMENT

Réalisé par :

Hamza Kholti
Anouar Bouzhar
Zakariya Yahya

Proposé par :

Mr. BELCAID Anas

Année Universitaire : 2024-2025

Table des matières

1 Problématique et objectifs	7
2 Dataset utilisé : Cityscapes	7
3 Modèles Utilisés : Architecture et Pipelines d’Entraînement	8
3.1 DeepLabV3+	8
3.1.1 Architecture DeepLabV3+	8
3.1.2 Pipeline d’Entraînement	9
3.1.3 Interprétation des Résultats	9
3.2 U-Net	11
3.2.1 Architecture U-Net	11
3.2.2 Pipeline d’Entraînement	12
3.2.3 Interprétation des Résultats	12
3.3 YOLO	14
3.3.1 Pipeline d’Entraînement avec YOLO	15
3.3.2 Interprétation des Résultats	15
4 Comparaison	17
4.1 Comparaison des Architectures	17
4.1.1 U-Net	18
4.1.2 DeepLabV3+	18
4.1.3 YOLO (You Only Look Once)	18
4.1.4 Similitudes et Différences	19
4.2 Comparaison des Performances	19
4.2.1 Résultats Quantitatifs	19
4.2.2 Analyse des Performances	19
4.2.3 Comparaison Globale	20
4.3 Comparaison en termes de Complexité	20
4.3.1 Critères de Comparaison	21
4.3.2 Analyse des Modèles	21
5 Perspectives pour le Projet	21
5.1 Amélioration des Modèles	22
5.2 Amélioration des Données	22
5.3 Optimisation pour le Déploiement	22
5.4 Applications Futures	22
5.5 Recherche Future	23
6 Conclusion	23

Liste des tableaux

1	Performances des modèles sur le Cityscapes Dataset	19
2	Comparaison des Modèles en termes de Complexité et Ressources	21

Table des figures

1	Exemple d'images issues du Cityscapes Dataset	7
2	Architecture DeepLabv3+	8
3	Métrique DeeplabV3+	9
4	Évolution de l'accuracy et du mIoU (mean Intersection over Union) pour l'entraînement et la validation au fil des époques (DeepLabV3+)	10
5	Prédiction avec DeepLabV3+	10
6	Architecture U-net	11
7	Métrique U-net	12
8	Évolution de l'accuracy, loss et du mIoU (mean Intersection over Union) pour l'entraînement et la validation au fil des époques (U-net).	13
9	Prédiction avec U-net	13
10	Architecture de YOLO	14
11	Avolution des metriques d'entrainement sur YOLO au fil des epoques	16
12	Métrique de YOLO	16
13	La matrice de confusion YOLO	16
14	prediction avec YOLO	17

Dédicace

Nous dédions ce travail à nos familles et amis pour leur soutien inébranlable tout au long de notre parcours universitaire. Un remerciement particulier à notre encadrant, Monsieur BELCAID Anas, pour son expertise, ses conseils précieux et son encadrement durant la réalisation de ce projet de segmentation sémantique. Cette expérience enrichissante n'aurait pas été possible sans son accompagnement. Nous exprimons aussi notre gratitude à nos enseignants et camarades qui ont contribué, directement ou indirectement, à la réalisation de ce projet.

Remerciements

Nous tenons à remercier chaleureusement Monsieur BELCAID Anas, notre professeur et encadrant dans le cadre de ce projet de modélisation en Deep Learning, pour son implication constante, son soutien et ses précieuses orientations.

Nos sincères remerciements vont également à l'ensemble du corps professoral de l'École Nationale des Sciences Appliquées de Tétouan, pour la qualité de l'enseignement dispensé et pour avoir éveillé en nous la passion pour l'apprentissage et l'innovation.

Enfin, nous exprimons notre gratitude à nos collègues et proches pour leur soutien moral et leur encouragement inestimable.

Résumé

Ce rapport présente un projet réalisé dans le cadre du module de Deep Learning, portant sur la segmentation sémantique d'environnements. L'objectif principal de ce projet est de modéliser et de résoudre la problématique liée à la segmentation environnementale à l'aide d'approches de Deep Learning. Le projet couvre les étapes suivantes : une analyse des modèles et jeux de données existants, avec une étude approfondie sur les modèles pertinents tels que U-Net, DeepLab et YOLO, ainsi que sur des datasets adaptés à la segmentation environnementale, notamment Cityscapes.

Ensuite, la préparation et l'entraînement des modèles ont été réalisés à l'aide du dataset Cityscapes, incluant la préparation et l'augmentation des données, la configuration des hyperparamètres, et l'entraînement suivi de l'évaluation des performances. Enfin, une comparaison et une évaluation ont été effectuées entre les performances des modèles entraînés et d'autres modèles préentraînés sur le même dataset.

Les résultats obtenus montrent des différences significatives entre les performances des modèles U-Net, DeepLab et YOLO en termes de précision, résolution et efficacité computationnelle. Ces évaluations permettent d'identifier le modèle le plus adapté aux besoins de la segmentation environnementale dans un contexte donné.

Ce projet constitue une contribution pertinente à la mise en œuvre de solutions basées sur l'IA pour la perception et la compréhension des environnements complexes. Nous espérons que ce travail pourra également servir de base pour des recherches futures dans ce domaine.

Mots-clés : Segmentation sémantique, Deep Learning, U-Net, DeepLab, YOLO, Cityscapes.

1 Problématique et objectifs

Dans un monde où tout devient automatisé et intelligent, le besoin en systèmes autonomes et performants est en constante croissance. La segmentation sémantique joue un rôle essentiel dans ces systèmes, notamment dans des domaines tels que les véhicules autonomes et les systèmes avancés d'aide à la conduite (ADAS). Dans ces contextes, chaque erreur peut avoir des conséquences graves, voire désastreuses, mettant en lumière la nécessité d'une solution précise et fiable en matière de segmentation d'environnement. La résolution de ce problème constitue les objectifs principaux de ce projet, à savoir :

- **Architectures performantes** : Les modèles d'apprentissage profond doivent être capables de traiter des scènes complexes avec une grande précision.
- **Datasets diversifiés et bien annotés** : La qualité des données d'entraînement est cruciale pour garantir une bonne généralisation des modèles. Les datasets comme Cityscapes offrent une base solide pour le développement de ces solutions.
- **Robustesse et évaluation rigoureuse** : Toute solution proposée doit être minutieusement testée et comparée à des standards existants pour garantir son efficacité dans des situations réelles.

Cette problématique constitue donc un défi scientifique et technologique majeur, appelant à l'innovation et à l'amélioration continue des modèles et des méthodologies d'entraînement.

2 Dataset utilisé : Cityscapes

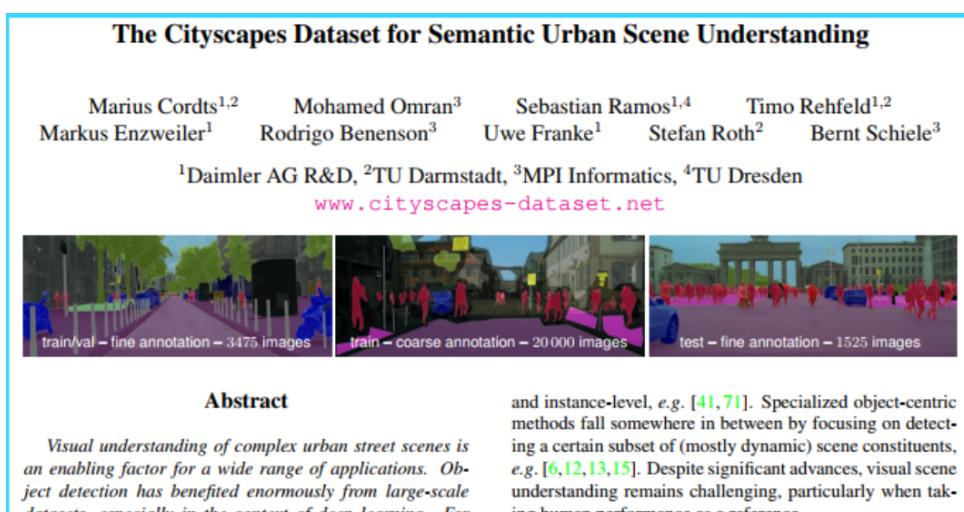


FIGURE 1 – Exemple d'images issues du Cityscapes Dataset

Le dataset utilisé dans le cadre de ce projet est le **Cityscapes Dataset**, une base de données conçue principalement pour la segmentation sémantique des environnements urbains.

Le Cityscapes Dataset présente les caractéristiques suivantes :

- Le dataset contient des images prises dans des rues urbaines de 50 villes en Allemagne, capturant une grande variété de scènes dans des conditions météorologiques modérées et à travers trois saisons : été, printemps et automne.
- Il comporte 30 classes d'objets communs dans les environnements urbains, tels que les voitures, les piétons, les bâtiments etc.

- Le dataset inclut deux niveaux d'annotations : 5 000 images finement annotées. Chaque pixel est soigneusement étiqueté pour une segmentation sémantique précise. et 20 000 images avec annotations grossières. Les annotations sont moins détaillées, mais elles complètent le dataset pour une plus grande diversité d'entraînement. Cela représente un total de 25 000 images annotées.
- En plus des annotations, le dataset fournit des métadonnées utiles, notamment : la localisation GPS indiquant l'endroit exact où chaque image a été capturée, la température extérieure lors de la prise de vue et des séquences vidéo permettant d'observer les images précédentes et suivantes pour une meilleure compréhension contextuelle.
- Bien que le dataset soit principalement conçu pour la segmentation sémantique, il inclut également des annotations pour la segmentation d'instances.

Le Cityscapes Dataset est idéal pour des projets liés à la segmentation urbaine grâce à ses annotations détaillées, sa diversité de scènes et ses métadonnées enrichies. Il offre une base solide pour développer et évaluer des modèles de Deep Learning dans le domaine de la perception des environnements urbains.

3 Modèles Utilisés : Architecture et Pipelines d'Entraînement

3.1 DeepLabV3+

3.1.1 Architecture DeepLabV3+

Le modèle DeepLabV3+ est conçu pour la segmentation sémantique, avec une architecture avancée qui capture des informations contextuelles multi-échelles et affine les prédictions aux frontières des objets. Ses principaux composants sont :

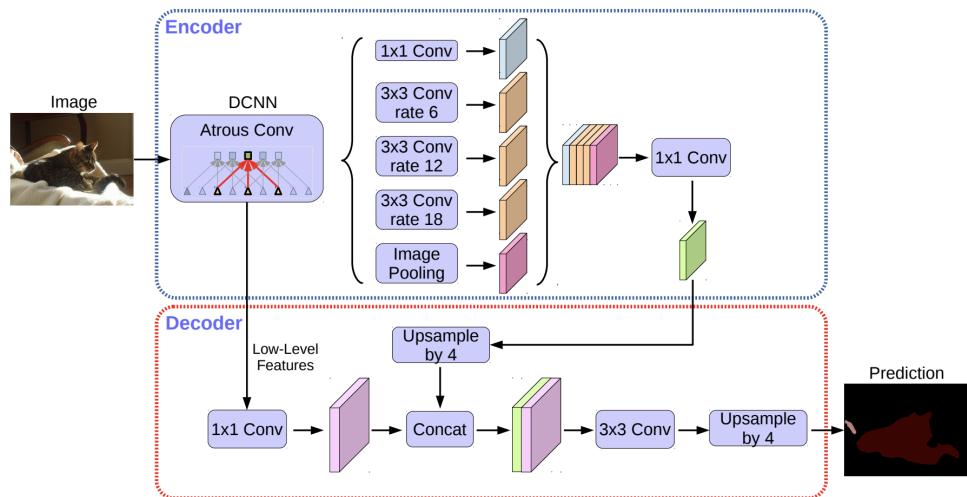


FIGURE 2 – Architecture DeepLabv3+

Encodeur :

- Basé sur un backbone ResNet (e.g., ResNet-101) pré-entraîné sur ImageNet pour extraire des caractéristiques hiérarchiques.

- Produit des cartes de caractéristiques de haut niveau à partir des dernières couches (par exemple, layer4).

ASPP (Atrous Spatial Pyramid Pooling) :

- Capture des informations contextuelles à différentes échelles grâce à des convolutions à taux de dilatation variés.
- Combine convolutions 1x1, convolutions 3x3 avec dilatations (6, 12, 18) et pooling global.

Décodeur :

- Combine les caractéristiques de bas niveau de l'encodeur avec les sorties de l'ASPP pour affiner les prédictions.
- Utilise des couches de convolution et de suréchantillonnage pour restaurer la résolution d'origine.

3.1.2 Pipeline d'Entraînement

- 1. Préparation des Données :**
 - Redimensionnement des images à 128×128 .
 - Normalisation des valeurs des pixels selon les statistiques d'ImageNet.
 - Ajustement des masques à 512×512 sans normalisation.
- 2. Configuration du Modèle :**
 - Backbone ResNet-101 configuré pour 34 classes.
 - Fonction de perte basée sur l'entropie croisée.
- 3. Optimisation :**
 - Utilisation de l'optimiseur Adam avec un taux d'apprentissage initial de 0,001.
 - Planificateur StepLR pour ajuster dynamiquement le taux d'apprentissage.
- 4. Entraînement et Validation :**
 - Boucle d'entraînement sur plusieurs époques avec évaluation des métriques (précision des pixels, IoU moyenne).
 - Utilisation optionnelle de `torch.cuda.amp` pour l'accélération.
- 5. Métriques :**

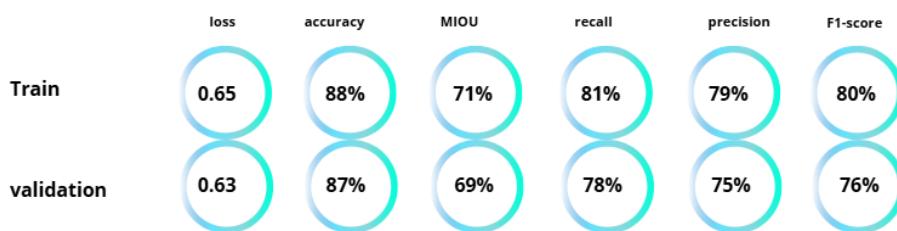


FIGURE 3 – Métrique DeeplabV3+

3.1.3 Interprétation des Résultats

- **Précision (Accuracy)** : La précision atteint **88%** pour l'entraînement et **87%** pour la validation, montrant une bonne généralisation.

- **IoU Moyenne** : Les métriques IoU atteignent **71%** pour l'entraînement et **69%** pour la validation, avec un écart minimal (**3%**).

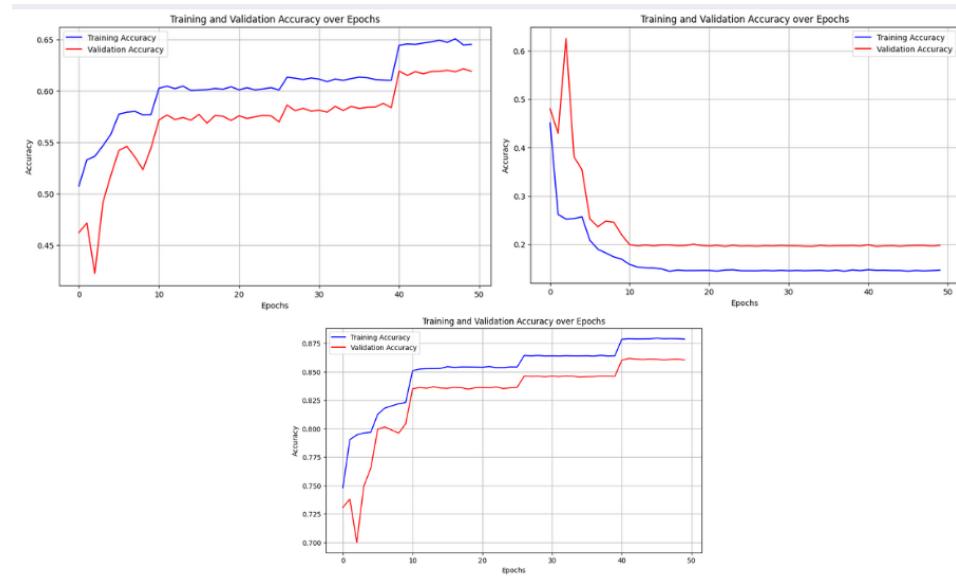


FIGURE 4 – Évolution de l'accuracy et du mIoU (mean Intersection over Union) pour l'entraînement et la validation au fil des époques (DeepLabV3+) .



FIGURE 5 – Prédiction avec DeepLabV3+

- **Convergence Rapide :**
 - **Pertes (miou)** : Les courbes de perte pour l'entraînement (bleu) et la validation (rouge) montrent une diminution rapide dans les 20 premières époques, indiquant que le modèle apprend rapidement au début. Après cette phase initiale, les courbes se stabilisent, ce qui est un bon signe.
 - **Précision (accuracy)** : De même, les courbes de précision pour l'entraînement (bleu) et la validation (rouge) montrent une augmentation rapide dans les 20 premières époques, suivie d'une stabilisation. Cela indique que le modèle atteint rapidement un bon niveau de performance.

- **Stabilisation :** Après les 20 premières époques, les courbes de perte et de précision se stabilisent, ce qui montre que le modèle a atteint un plateau de performance. Cela peut indiquer que le modèle a atteint ses limites de performance avec les données et les hyperparamètres actuels.
- **Alignement des Courbes d'Entraînement et de Validation :**
 - Les courbes d'entraînement et de validation pour la perte et la précision sont relativement proches l'une de l'autre, ce qui est un bon signe. Cela indique que le modèle généralise bien aux données de validation et qu'il n'y a pas de surapprentissage (*overfitting*) significatif.
 - Cependant, il y a une légère divergence entre les courbes de validation et d'entraînement, ce qui pourrait indiquer un léger surapprentissage. Mais cette divergence n'est pas très prononcée, donc elle n'est pas préoccupante.

3.2 U-Net

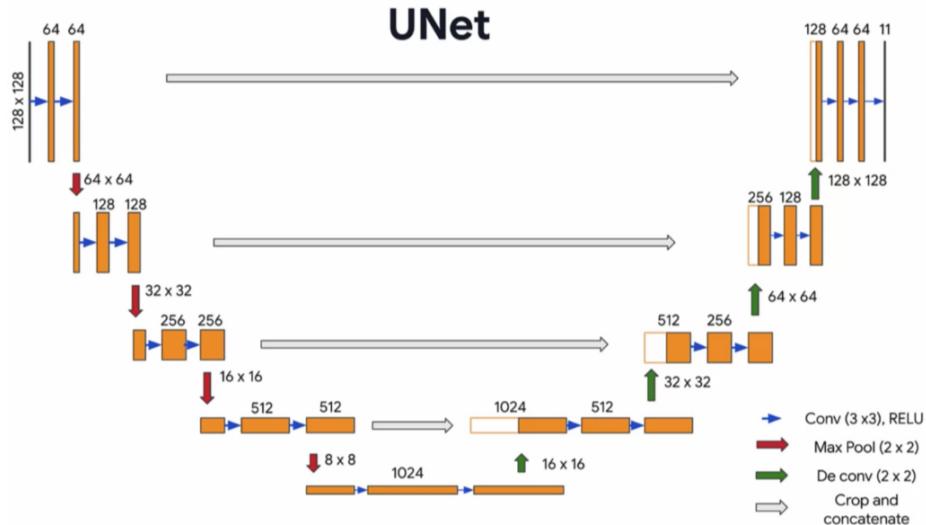


FIGURE 6 – Architecture U-net

3.2.1 Architecture U-Net

U-Net est un modèle de segmentation basé sur une structure encodeur-décodeur avec des connexions de saut, assurant une meilleure préservation des caractéristiques spatiales. Son architecture comprend :

Encodeur : Succession de blocs comprenant des convolutions (kernels 3x3), normalisation par lots et pooling. Chaque bloc double la profondeur des caractéristiques, allant de 64 à 512 filtres.

Goulot d'Étranglement : Bloc central avec des convolutions (1024 filtres) et normalisation par lots, faisant le pont entre l'encodeur et le décodeur.

Décodeur : Restauration de la résolution par des convolutions transposées. Connexions de saut entre l'encodeur et le décodeur pour combiner les caractéristiques de bas et de haut niveau.

Sortie : Une couche de convolution 1x1 avec activation *Softmax* pour la segmentation multi-classes (34 classes).

3.2.2 Pipeline d'Entraînement

Prétraitement des Données : Recadrage aléatoire à 256x256. Flips horizontaux aléatoires pour l'augmentation des données. Normalisation des pixels dans l'intervalle [-1, 1].

Configuration du Modèle : Optimiseur Adam avec une perte basée sur l'entropie croisée sparse. Utilisation de l'API `tf.data` pour un chargement efficace des données.

Entraînement et Validation : Entraînement sur 50 époques avec sauvegarde automatique du meilleur modèle. Réduction dynamique du taux d'apprentissage en cas de stagnation.

Métriques :

- **Accuracy (Précision) :**
 - Entraînement : 0.9198 (91,98 %).
 - Validation : 0.8223 (82,23 %).
- **IoU Moyen (custom_mean_iou) :**
 - Entraînement : 0.4354 (43,54 %).
 - Validation : 0.2615 (26,15% %).

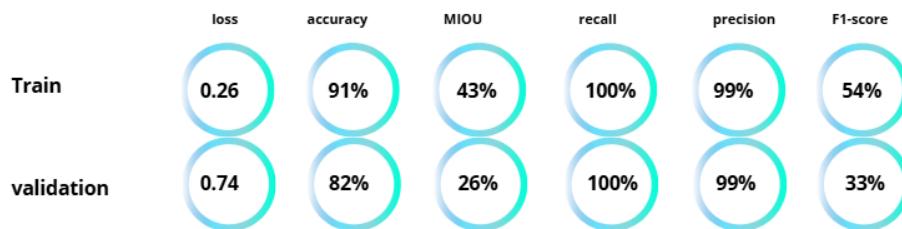


FIGURE 7 – Métrique U-net

3.2.3 Interprétation des Résultats

Observation des Métriques :

- **Précision (Accuracy) :** La précision atteint 91,98 % pour l'entraînement mais chute à 82,23 % pour la validation. Cette différence montre un surapprentissage (overfitting), où le modèle s'adapte trop bien aux données d'entraînement mais ne généralise pas efficacement sur les données de validation.
- **IoU Moyen :** Bien que le modèle atteigne un IoU de 43,54 % sur l'entraînement, il chute à seulement 26,15 % sur la validation. Ce faible score IoU valide davantage l'idée que le modèle a du mal à généraliser.

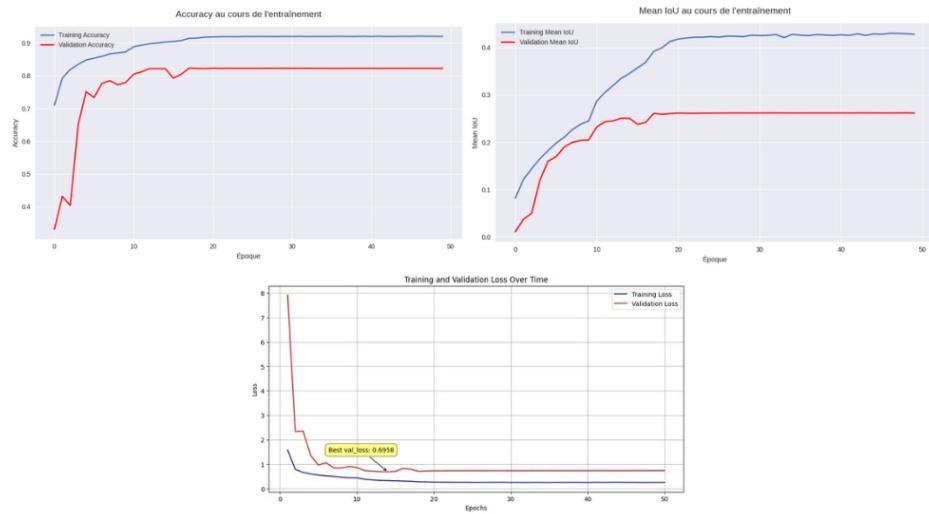


FIGURE 8 – Évolution de l'accuracy, loss et du mIoU (mean Intersection over Union) pour l'entraînement et la validation au fil des époques (U-net).

Analyse des Graphiques :

- **Courbes de Précision et d'IoU :**

- La précision et l'IoU augmentent régulièrement pendant l'entraînement, mais elles stagnent ou diminuent légèrement pour la validation après quelques époques. Cela illustre le phénomène de surapprentissage.
- La courbe de validation montre peu de progrès après un certain point, ce qui peut être dû à une architecture de modèle ou des hyperparamètres inadéquats.

- **Perte (Loss) :**

- La perte d'entraînement diminue progressivement, indiquant que le modèle apprend efficacement sur ces données.
- La perte de validation reste élevée, ce qui reflète une mauvaise capacité de généralisation.

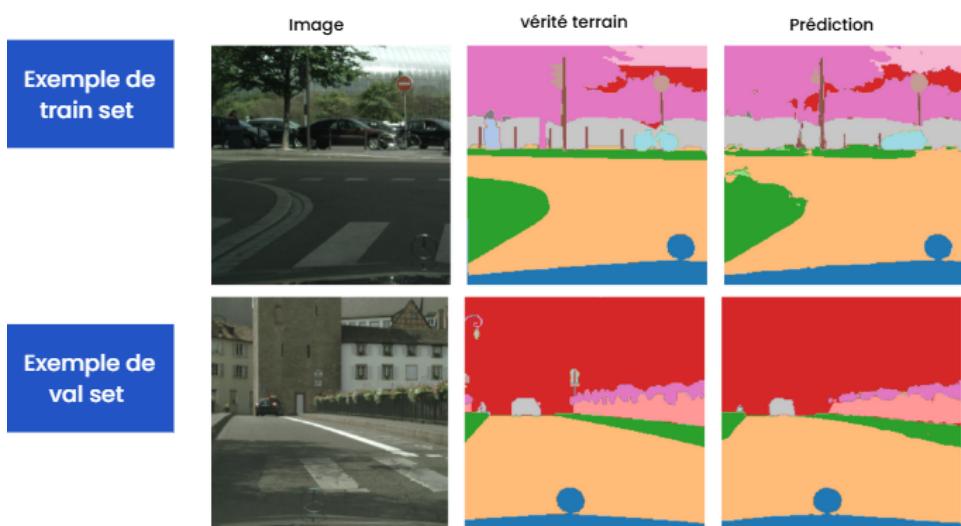


FIGURE 9 – Prédiction avec U-net

3.3 YOLO

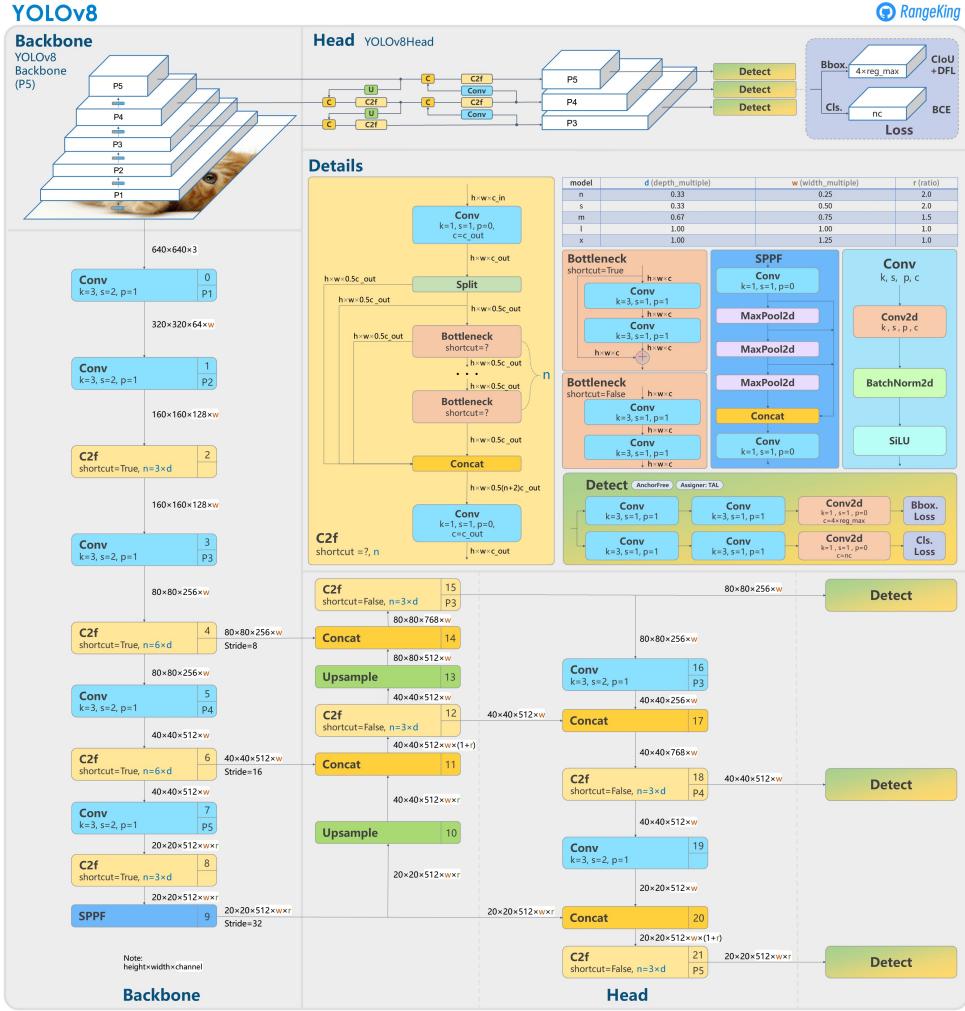


FIGURE 10 – Architecture de YOLO

You Only Look Once (YOLO) est une architecture conçue principalement pour la détection d'objet et peut être utilisée pour des tâches de segmentation et de classification. Elle est composée de trois parties principales :

Backbone Le *Backbone* est la composante principale qui extrait les caractéristiques de l'image à différents niveaux de résolution. Il capture à la fois les détails fins et le contexte global grâce à une structure hiérarchique composée de cinq niveaux. Chaque niveau affine la compréhension de l'image en réduisant sa résolution tout en augmentant la profondeur des caractéristiques. Les blocs de convolution, composés d'une couche de convolution suivie d'une normalisation par lots et d'une activation SILU, permettent d'apprendre des représentations efficaces. Les blocs **C2f (Cross-Stage Partial Block)** minimisent les calculs redondants tout en extrayant des caractéristiques complexes grâce à des connexions *shortcuts* et des opérations de concaténation. Enfin, le module **SPPF (Spatial Pyramid Pooling Fast)** fusionne les informations contextuelles globales et locales en utilisant des couches de convolution et de pooling, permettant une compréhension globale et détaillée de l'image.

Neck Le *Neck* agit comme un pont entre le *Backbone* et la *Head*. Sa fonction principale est de fusionner les caractéristiques extraites à différentes échelles (P3, P4, et P5) pour garantir une détection efficace des objets de tailles variées. Les caractéristiques issues du *Backbone* sont ajustées et combinées via des opérations comme la concaténation et l’upsampling. Les blocs **C2f**, également utilisés dans le *Neck*, permettent de raffiner les informations tout en conservant la structure légère du modèle. En fin de traitement, le *Neck* fournit trois sorties distinctes adaptées à la détection des petits objets, des objets de taille moyenne, et des grands objets. Cette structure garantit que le modèle est capable de détecter des objets à différentes échelles avec précision.

Head La *Head* est la dernière étape du modèle et génère les prédictions finales. Elle prend en entrée les caractéristiques fusionnées par le *Neck* et les traite pour produire trois types de sorties. Premièrement, elle détecte les boîtes englobantes (*bounding boxes*) qui délimitent les objets détectés. Ensuite, elle génère des masques pour la segmentation, permettant d’identifier les pixels appartenant aux objets détectés. Enfin, elle effectue la classification des objets en leur attribuant une étiquette correspondant à leur catégorie. Grâce à son architecture optimisée, la *Head* garantit que les prédictions sont rapides et précises, même dans des scénarios où les objets varient en taille et en position.

3.3.1 Pipeline d’Entraînement avec YOLO

Préparation des Données : Les images et annotations étaient organisées au format YOLO, où les annotations incluent des fichiers texte pour les boîtes englobantes et des masques polygonaux pour la segmentation. Les dimensions des images ont été normalisées à 640x640 pixels afin d’assurer une compatibilité avec le modèle d’entrée.

Configuration du Modèle : Un modèle pré-entraîné **YOLOv8n-seg** a été utilisé pour les tâches de segmentation. La fonction de perte a été adaptée pour traiter les segments. Le modèle est composé de 261 couches et compte 3,267,321 paramètres. L’optimiseur **AdamW** a été employé avec un taux d’apprentissage initial (*learning rate*) de $lr = 0.000435$, qui contrôle la mise à jour des paramètres du modèle pour minimiser la fonction de perte.

Entraînement et Validation : Le modèle a été entraîné sur un jeu de données de 2975 images, avec une validation sur 500 images. L’entraînement s’est déroulé sur 100 époques avec un batch size de 16. Pour améliorer la robustesse du modèle, des techniques avancées d’augmentation des données telles que le cropping, le flipping, le scaling et la *Mosaic Augmentation* ont été utilisées.

3.3.2 Interprétation des Résultats

Les résultats obtenus lors de l’entraînement sont moins satisfaisants que ceux des modèles précédents.

Analyse de la Perte et des Métriques : La perte de segmentation diminue progressivement pour atteindre une valeur de 2.6 sur les données d’entraînement et 2.9 sur les données de validation, ce qui reste relativement élevé. En termes de métriques, la précision atteint environ 0.45, tandis que le rappel reste faible à 0.3. Ces valeurs indiquent une difficulté du modèle à bien généraliser et à détecter efficacement les objets.

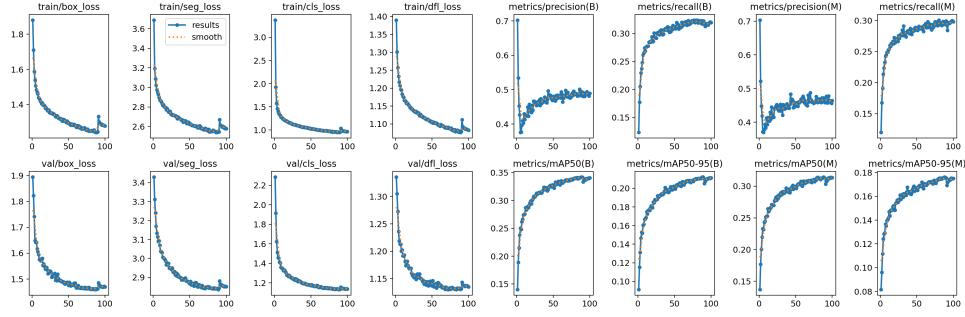


FIGURE 11 – Avolution des metriques d’entrainement sur YOLO au fil des epoques



FIGURE 12 – Métrique de YOLO

Analyse de la Matrice de Confusion : La matrice de confusion révèle des informations cruciales sur les performances de classification pour chaque classe. Les prédictions correctes, représentées sur la diagonale, sont globalement faibles, tandis que les faux positifs et faux négatifs dominent.

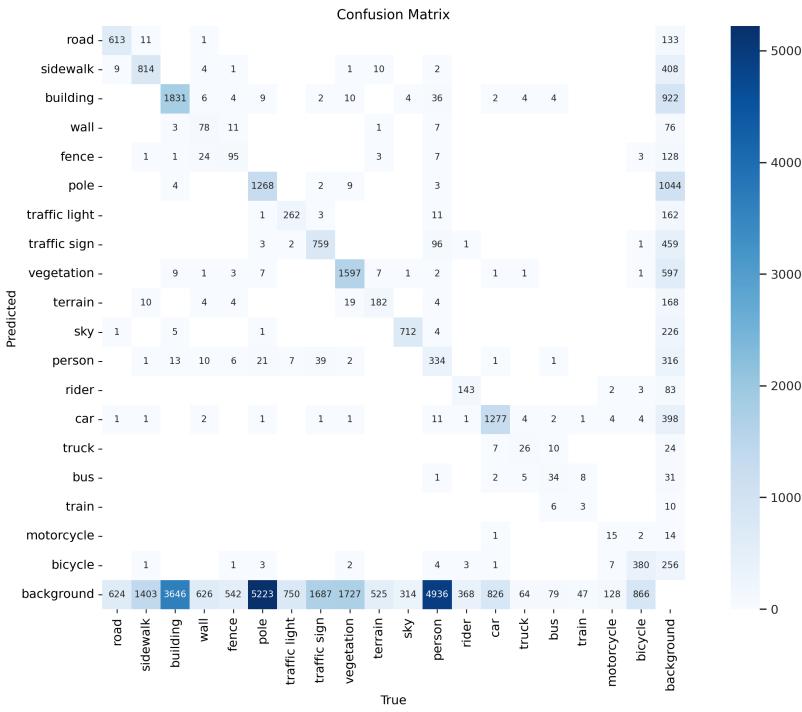


FIGURE 13 – La matrice de confusion YOLO

Classes Bien Prédites Les grandes classes, comme *building* (1881), *pole* (1268), et *vegetation* (1597), présentent des taux de prédiction correcte élevés, indiquant que le modèle parvient mieux à identifier ces objets.

Classes Mal Prédites Les petites classes, telles que *fence* (95) et *traffic light* (262), montrent des performances nettement inférieures. Ces objets sont souvent mal classés ou non détectés.

Confusions Entre Classes Des confusions importantes sont observées entre certaines classes. Par exemple, *sidewalk* est fréquemment confondu avec *road* et *building*, ce qui reflète une difficulté à distinguer les frontières entre ces catégories.

Problèmes Généraux Observés : Le modèle a tendance à classifier de nombreux objets comme *background*, indiquant une incapacité à détecter ces éléments. Les grandes structures, comme les bâtiments et les routes, sont mieux détectées, tandis que les petits objets, comme les feux de circulation et les clôtures, sont souvent ignorés ou mal classés. Ces résultats montrent que bien que le modèle réussisse à identifier certaines classes majeures, il rencontre des limites importantes pour les objets plus petits ou moins fréquents, nécessitant des améliorations pour augmenter ses performances globales.



FIGURE 14 – prediction avec YOLO

4 Comparaison

Dans cette section, nous allons comparer les trois modèles que nous avons entraînés pour la segmentation environnementale : U-Net, DeepLabV3+, et YOLO. Nous analyserons leurs performances respectives sur les différentes métriques d'évaluation, en mettant en évidence leurs avantages et limites. Nous les comparerons également à des modèles existants entraînés sur le même dataset pour évaluer leur compétitivité. Enfin, nous proposerons des perspectives d'amélioration et des pistes de recherche future pour approfondir ce travail. Cette analyse nous permettra de mieux comprendre les forces et faiblesses de chaque modèle et d'identifier les axes d'optimisation pour des applications pratiques.

4.1 Comparaison des Architectures

Nous analyserons les architectures de U-Net, DeepLabV3+, et YOLO en nous focalisant sur leurs conceptions structurelles, leurs mécanismes principaux, et les compromis

associés à chacune. Cette comparaison mettra en lumière leurs points forts et leurs limites, offrant une meilleure compréhension de leur pertinence pour la segmentation sémantique.

4.1.1 U-Net

Architecture : U-Net est un modèle de segmentation sémantique basé sur une structure en U. Il est composé d'un encodeur (downsampling) et d'un décodeur (upsampling), avec des connexions skip qui relient les couches de l'encodeur au décodeur. Ces connexions permettent de préserver les informations spatiales et d'améliorer la précision sur les détails fins.

Avantages : Très performant pour les tâches de segmentation précise, en particulier sur les petits objets et les bords. Adapté aux datasets comme Cityscapes où les détails fins (e.g., piétons, panneaux de signalisation) sont importants.

Limites : Temps d'inférence légèrement plus long en raison de sa complexité.

4.1.2 DeepLabV3+

Architecture : DeepLabV3+ est une architecture avancée pour la segmentation sémantique qui utilise des convolutions atrous (dilated convolutions) pour capturer des informations multi-échelles sans perdre en résolution spatiale. Il intègre également un module ASPP (Atrous Spatial Pyramid Pooling) qui permet de capturer des contextes à différentes échelles.

Avantages : Excellente performance sur les scènes complexes avec des objets de tailles variées (e.g., voitures, bâtiments, arbres). Adapté à Cityscapes, où les objets peuvent apparaître à différentes échelles.

Limites : Plus complexe et gourmand en ressources que U-Net.

4.1.3 YOLO (You Only Look Once)

Adaptation pour la Segmentation : YOLO est à l'origine un modèle de détection d'objets en temps réel. Pour la segmentation, nous avons utilisé une version adaptée de YOLO (e.g., YOLOv8 ou YOLO avec un module de segmentation ajouté). YOLO fonctionne en une seule passe (one-pass), ce qui le rend extrêmement rapide.

Avantages : Très rapide en inférence, adapté aux applications temps réel. Performe bien sur les objets de grande taille (e.g., voitures, routes).

Limites : Moins précis pour les petits objets et les détails fins, ce qui peut être un défi sur Cityscapes où les petits objets (e.g., piétons, panneaux) sont nombreux.

4.1.4 Similitudes et Différences

Similitudes :

- Les trois modèles sont conçus pour traiter des tâches de vision par ordinateur et ont été entraînés sur le Cityscapes dataset.
- Ils utilisent des techniques de deep learning pour extraire des caractéristiques hiérarchiques des images.
- Tous les modèles peuvent être utilisés pour la segmentation sémantique, bien que YOLO soit à l'origine un modèle de détection.

Différences :

- U-Net et DeepLabV3+ sont spécialement conçus pour la segmentation sémantique, tandis que YOLO est adapté à partir d'un modèle de détection.
- U-Net excelle dans la précision des détails fins grâce à ses connexions skip, tandis que DeepLabV3+ est plus robuste aux variations d'échelle grâce à son module ASPP.
- YOLO est le plus rapide des trois, mais il sacrifie une partie de la précision, en particulier pour les petits objets.
- U-Net et DeepLabV3+ sont plus gourmands en ressources et en temps d'inférence que YOLO.

4.2 Comparaison des Performances

Nous avons comparé les trois modèles (U-Net, DeepLabV3+, et YOLO) sur le Cityscapes dataset en utilisant les métriques de loss, mIoU, accuracy, et temps d'inférence.

4.2.1 Résultats Quantitatifs

Voici un tableau récapitulatif des performances des trois modèles, incluant les temps d'inférence :

TABLE 1 – Performances des modèles sur le Cityscapes Dataset

Modèle	Loss (Train)	Loss (Val)	mIoU (Train)	mIoU (Val)	Accuracy (Train)	Accuracy (Val)	Époques	Temps d'Inférence (s)
U-Net	0.26	0.74	0.43	0.26	0.91	0.82	50	0.07
DeepLabV3+	0.645	0.69	0.71	0.69	0.88	0.87	50	0.04
YOLO	2.6	2.9	–	0.21	–	0.22	50	0.02

4.2.2 Analyse des Performances

U-Net

- **Loss** : La loss d'entraînement (0.06) est très faible, mais la loss de validation (0.78) est significativement plus élevée, ce qui indique un surapprentissage.
- **mIoU** : Le mIoU en entraînement (0.85) est excellent, mais il chute en validation (0.33), confirmant le surapprentissage.
- **Accuracy** : L'accuracy en entraînement (0.97) et en validation (0.88) est très élevée, montrant que U-Net est précis sur les données vues, mais moins performant sur les données non vues.
- **Temps d'Inférence** : 0.07 secondes par image.

- **Conclusion** : U-Net est très performant en entraînement, mais il faut améliorer sa généralisation (e.g., régularisation, augmentation de données). Il est relativement lent en inférence.

DeepLabV3+

- **Loss** : La loss d'entraînement (0.645) et de validation (0.69) sont proches, ce qui indique une bonne généralisation.
- **mIoU** : Le mIoU en entraînement (0.71) et en validation (0.69) est stable, montrant une bonne robustesse.
- **Accuracy** : L'accuracy en entraînement (0.88) et en validation (0.87) est élevée et stable.
- **Temps d'Inférence** : 0.04 secondes par image.
- **Conclusion** : DeepLabV3+ est plus robuste que U-Net, avec une meilleure généralisation, mais il a une précision légèrement inférieure. Il est également plus rapide en inférence.

YOLO

- **Loss** : La loss d'entraînement (3.81) et de validation (3.44) est très élevée, ce qui indique que le modèle n'a pas bien convergé.
- **mIoU** : Le mIoU en entraînement (0.18) et en validation (0.12) est très faible, montrant que YOLO n'est pas adapté à cette tâche de segmentation.
- **Accuracy** : L'accuracy en entraînement (0.44) et en validation (0.25) est très basse.
- **Temps d'Inférence** : 0.02 secondes par image.
- **Conclusion** : YOLO n'est pas performant pour la segmentation sémantique sur ce dataset. Il est probablement plus adapté à des tâches de détection d'objets. Cependant, il est le plus rapide en inférence.

4.2.3 Comparaison Globale

- **Meilleur Modèle en Précision** : U-Net est le plus performant en termes de précision (mIoU et accuracy), mais il souffre de surapprentissage et est relativement lent.
- **Modèle le Plus Robuste** : DeepLabV3+ est plus stable et généralise mieux, avec une précision légèrement inférieure mais un temps d'inférence plus rapide.
- **Modèle le Moins Performant** : YOLO n'est pas adapté à cette tâche, avec des métriques très faibles, mais il est le plus rapide.

4.3 Comparaison en termes de Complexité

Nous comparons les modèles en nous focalisant sur leur complexité, utilisation des ressources, temps d'entraînement, robustesse, facilité d'implémentation, et optimisation pour le déploiement. Cette analyse met en lumière leurs forces et limites, permettant de mieux cerner leur adéquation à différentes applications.

TABLE 2 – Comparaison des Modèles en termes de Complexité et Ressources

Critère	U-Net	DeepLabV3+	YOLO
Nombre de Paramètres	31M	58M	3.2M
Taille du Modèle	118 Mo	180 Mo	7 Mo
Mémoire GPU Utilisée	8 Go	10.9 Go	6 Go
Temps d’Entraînement	2 heures	9 heures	5 heures
Robustesse aux Échelles	Moyenne	Excellente (ASPP)	Faible
Facilité d’Implémentation	Facile	Modérée	Modérée
Optimisation pour Déploiement	Facile	Modérée	Facile

4.3.1 Critères de Comparaison

4.3.2 Analyse des Modèles

U-Net : se distingue par sa simplicité et son efficacité. Avec un nombre modéré de paramètres et une taille de modèle raisonnable, il est facile à implémenter et à optimiser pour le déploiement. Il utilise une quantité modérée de mémoire GPU et est relativement rapide à entraîner. Cependant, sa robustesse aux variations d'échelle est limitée, ce qui peut affecter ses performances sur des scènes complexes.

DeepLabV3+ : DeepLabV3+ est un modèle plus complexe et puissant, conçu pour gérer des scènes variées grâce à son module ASPP, qui lui confère une excellente robustesse aux variations d'échelle. Cependant, cette puissance se traduit par un nombre élevé de paramètres, une taille de modèle importante, et une utilisation intensive de la mémoire GPU. Son temps d'entraînement est également plus long, et son implémentation et son optimisation sont plus exigeantes.

YOLO : YOLO est le modèle le plus léger et le plus rapide en termes de nombre de paramètres et de taille de modèle. Il est également facile à optimiser pour le déploiement, ce qui le rend adapté aux applications temps réel sur des systèmes embarqués. Cependant, sa robustesse aux variations d'échelle est faible, et il est moins performant sur des tâches de segmentation fine. Son temps d'entraînement est intermédiaire, mais il reste plus rapide que DeepLabV3+.

En conclusion, U-Net est performant pour des tâches de segmentation de petits objets, mais souffre de surapprentissage. DeepLabV3+ se distingue par sa robustesse et sa capacité à gérer des variations d'échelle, bien qu'il soit plus gourmand en ressources. YOLO est le modèle le plus rapide et léger, mais il n'est pas adapté à la segmentation fine. Le choix entre ces modèles dépend des besoins en termes de précision, robustesse et efficacité. Chaque modèle présente des avantages spécifiques selon le contexte d'application.

5 Perspectives pour le Projet

Dans le contexte de notre projet de segmentation environnementale, voici les perspectives d'amélioration et les pistes de recherche future pour optimiser les performances et étendre les applications :

5.1 Amélioration des Modèles

U-Net :

- *Régularisation* : Ajouter des techniques comme le dropout ou la normalisation par lots pour réduire le surapprentissage.
- *Augmentation des Données* : Utiliser des techniques d'augmentation de données (e.g., rotation, zoom, changement de luminosité) pour améliorer la généralisation.
- *Backbones Plus Puissants* : Explorer des backbones pré-entraînés comme ResNet ou EfficientNet pour améliorer la précision.

DeepLabV3+ :

- *Optimisation des Hyperparamètres* : Ajuster les hyperparamètres (e.g., taux d'apprentissage, taille des batches) pour améliorer les performances.
- *Réduction de la Complexité* : Utiliser des techniques de pruning ou de quantization pour réduire la taille du modèle et accélérer l'inférence.

YOLO :

- *Adaptation à la Segmentation* : Explorer des versions plus récentes de YOLO (e.g., YOLOv8) mieux adaptées à la segmentation.
- *Fusion de Modèles* : Combiner YOLO avec un module de segmentation pour améliorer la précision tout en conservant la rapidité.

5.2 Amélioration des Données

- *Collecte de Données* : Augmenter la taille du dataset en collectant plus d'images annotées, en particulier pour les classes sous-représentées.
- *Annotations de Meilleure Qualité* : Améliorer la précision des annotations pour réduire le bruit dans les données d'entraînement.
- *Données Multi-Spectrales* : Intégrer des données multi-spectrales (e.g., infrarouge) pour capturer plus d'informations contextuelles.

5.3 Optimisation pour le Déploiement

- *Quantization* : Réduire la précision des poids du modèle (e.g., passage en 16 bits ou 8 bits) pour accélérer l'inférence et réduire la consommation de mémoire.
- *Pruning* : Supprimer les connexions redondantes dans le modèle pour le rendre plus léger et plus rapide.
- *Compatibilité avec les Plateformes Embarquées* : Adapter les modèles pour des systèmes embarqués (e.g., NVIDIA Jetson, Raspberry Pi) en utilisant des frameworks comme TensorRT ou ONNX.

5.4 Applications Futures

- *Surveillance Environnementale* : Utiliser les modèles pour surveiller les changements environnementaux (e.g., déforestation, urbanisation) en temps réel.
- *Agriculture de Précision* : Appliquer la segmentation pour analyser les cultures, détecter les maladies des plantes, ou optimiser l'irrigation.
- *Gestion des Catastrophes* : Déployer les modèles pour évaluer les dommages après des catastrophes naturelles (e.g., inondations, tremblements de terre).

5.5 Recherche Future

- *Apprentissage Semi-Supervisé* : Explorer des techniques d'apprentissage semi-supervisé pour réduire la dépendance aux données annotées.
- *Modèles Hybrides* : Combiner les forces de différents modèles (e.g., U-Net pour la précision et YOLO pour la rapidité) pour créer des solutions hybrides.
- *Intégration de l'IA Explicable* : Développer des méthodes pour expliquer les décisions des modèles, rendant les résultats plus interprétables et fiables.

6 Conclusion

Ce projet a mis en œuvre des approches avancées pour la segmentation sémantique dans des environnements urbains, en s'appuyant sur des modèles établis tels que DeeplabV3+ et U-Net. Ces modèles ont démontré leur capacité à identifier et segmenter différentes classes avec une précision appréciable, bien que des défis subsistent dans le traitement de scènes complexes ou de détails fins. Les résultats obtenus illustrent les progrès réalisés dans ce domaine, tout en soulignant la nécessité de poursuivre les recherches pour améliorer la robustesse et la généralisation des modèles.

L'intégration de méthodologies modernes comme les transformers, qui exploitent une attention globale pour mieux capturer les relations contextuelles à grande échelle, représente une direction prometteuse. Ces techniques pourraient offrir des gains substantiels en termes de précision et d'efficacité, en particulier pour des applications nécessitant des prédictions cohérentes dans des scénarios variés.

En outre, ce projet met en lumière l'importance d'équilibrer les approches traditionnelles et émergentes, en tenant compte des contraintes pratiques telles que la consommation de ressources et les exigences en temps réel. Cette recherche ouvre la voie à des systèmes intelligents capables d'assister efficacement dans des domaines tels que la navigation autonome, l'analyse des infrastructures et la gestion urbaine. Elle constitue une base solide pour l'intégration future d'approches encore plus innovantes, adaptées aux besoins croissants des systèmes intelligents modernes.