Conception et développement d'un système de segmentation d'images pour la vision embarquée des véhicules autonomes

Stéphanie ROULLAND

9 décembre 2024

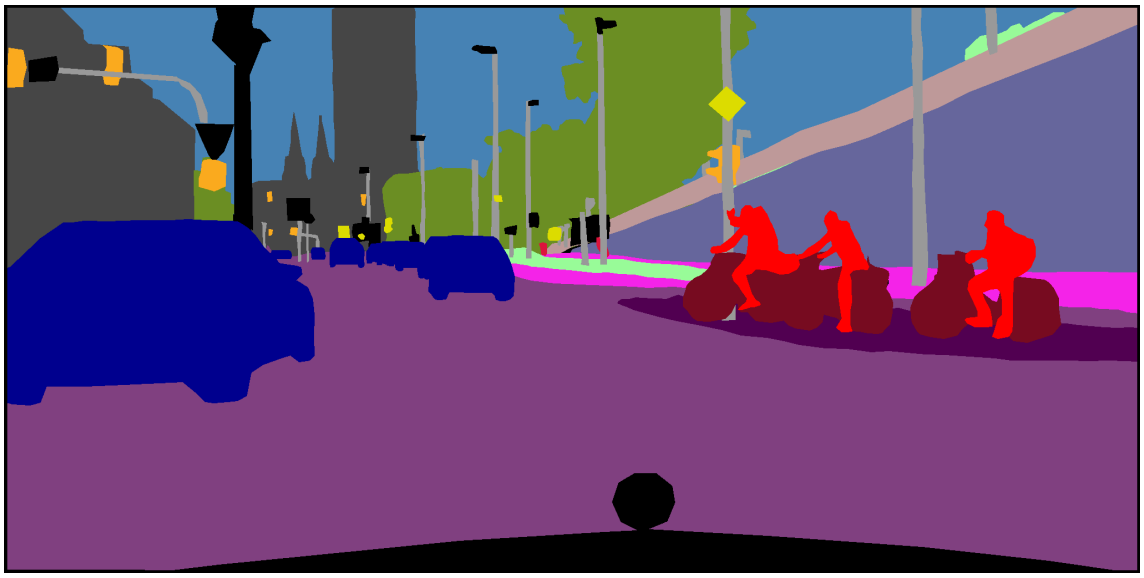


Table des matières

1 Introduction

Dans un contexte où les véhicules autonomes représentent un enjeu majeur pour l'avenir de la mobilité, la vision par ordinateur joue un rôle central dans leur capacité à percevoir et interpréter leur environnement. Future Vision Transport, entreprise spécialisée dans la conception de systèmes embarqués de vision par ordinateur pour véhicules autonomes, développe des solutions innovantes pour répondre à ces défis technologiques.

1.1 Contexte du projet

Au sein du système embarqué de vision par ordinateur de Future Vision Transport, la chaîne de traitement se compose de quatre maillons essentiels : l'acquisition des images en temps réel, leur traitement, leur segmentation et enfin le système de décision. La segmentation d'images constitue une étape critique de cette chaîne, permettant d'identifier et de classifier les différents éléments de la scène urbaine (routes, véhicules, piétons, etc.).

1.2 Rôle de la segmentation dans le système

Positionnée entre le bloc de traitement des images et le système de décision, la segmentation sémantique permet une compréhension fine de l'environnement du véhicule. Elle reçoit en entrée les images prétraitées et fournit au système de décision une carte détaillée des différents éléments de la scène, essentielle pour la prise de décision en temps réel du véhicule autonome.

1.3 Objectifs du projet

Ce projet vise trois objectifs principaux :

1. L'entraînement d'un modèle de segmentation performant sur les 8 catégories principales du dataset Cityscapes, en utilisant le framework Keras, garantissant ainsi la compatibilité avec l'infrastructure existante.
2. La conception et le déploiement d'une API de prédiction permettant l'intégration fluide du modèle dans la chaîne de traitement.
3. Le développement d'une application web de visualisation pour tester et valider les performances du système.

1.4 Structure de la note technique

Cette note technique s'articule autour de plusieurs sections clés : un état de l'art des techniques de segmentation d'images, une présentation détaillée du modèle retenu et de son architecture, une analyse des résultats obtenus avec différentes approches d'augmentation des données, et enfin une discussion des perspectives d'amélioration du système.

2 État de l'art en segmentation d'images

2.1 Introduction aux techniques de segmentation

La segmentation d'images constitue une étape fondamentale dans la compréhension de scènes pour les véhicules autonomes. Elle permet d'identifier et de délimiter précisément les différents éléments présents dans l'environnement urbain. Dans ce domaine, on distingue trois types principaux de segmentation :

- **La segmentation sémantique** qui assigne une classe à chaque pixel de l'image (route, voiture, piéton, etc.).
- **La segmentation d'instance** qui différencie les objets individuels au sein d'une même classe (distingue différentes voitures entre elles).
- **La segmentation panoptique** qui combine les deux approches précédentes.

Dans le contexte des véhicules autonome, la segmentation sémantique revêt une importance particulière car elle permet une compréhension globale de la scène, essentielle pour la navigation et la prise de décision en temps réel.

2.2 Évolution des approches de segmentation

2.2.1 Méthodes traditionnelles

L'histoire de la segmentation d'images trouve ses racines dans des techniques classiques de traitement d'images. Ces approches fondamentales comprennent :

- **Le seuillage** : séparation des objets basée sur l'intensité des pixels.
- **La croissance de régions** : regroupement de pixels selon des critères de similarité.
- **Les contours actifs** : évolution des courbes pour délimiter les objets.
- **Les watershed** : segmentation basée sur la topographie de l'intensité de l'image.

Bien que ces méthodes aient posé les bases théoriques essentielles de la segmentation, leurs limitations sont devenues évidentes face à la complexité des scènes réelles, particulièrement en environnement urbain. Le manque de robustesse face aux variations d'éclairage, aux occlusions et à la diversité des objets a motivé la recherche de solutions plus avancées.

2.2.2 Avènement des réseaux de neurones convolutifs

L'introduction des réseaux de neurones convolutifs (CNN) a marqué un tournant décisif dans le domaine de la vision par ordinateur. Les Fully Convolutional Networks (FCN) ont particulièrement révolutionné l'approche de la segmentation sémantique en introduisant la première architecture entièrement convolutive.

Cette innovation majeure a permis de traiter les images de bout en bout, en combinant l'apprentissage automatique des caractéristiques pertinentes avec une segmentation précise au niveau du pixel. L'architecture FCN a ouvert la voie à de nombreuses évolutions, établissant un nouveau paradigme dans le traitement des images.

2.3 Architecture VGG16 et son adaptation pour la segmentation

2.3.1 Architecture originale de VGG16

Développé par le Visual Geometry Group de l'Université d'Oxford, VGG16 s'est imposé comme une architecture de référence en vision par ordinateur grâce à sa simplicité et son efficacité remarquables.

Le réseau s'articule autour de 13 couches convolutives suivies de 3 couches fully connected, utilisant exclusivement des filtres de convolution 3x3. Cette organisation permet une extraction progressive des caractéristiques, avec une profondeur croissante des features maps. À mesure que l'on avance dans le réseau, les champs réceptifs s'élargissent graduellement, permettant de capturer des motifs de plus en plus complexes et abstraits. La simplicité architecturale de VGG16, combinée à sa capacité à extraire des caractéristiques hiérarchiques pertinentes, en fait un choix particulièrement intéressant pour la segmentation d'images.

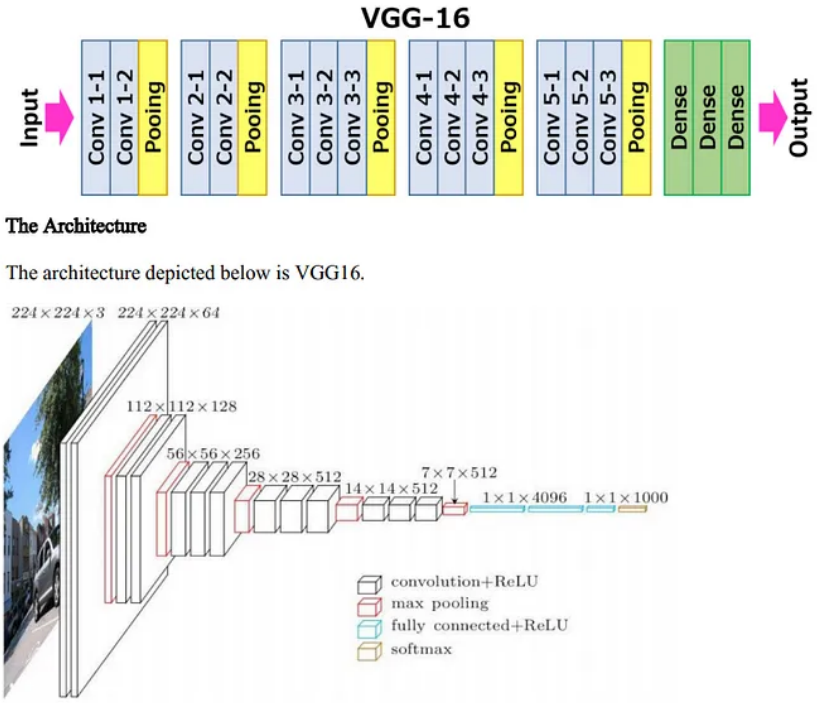


FIGURE 1 – Architecture de VGG16

2.3.2 Adaptation pour la segmentation sémantique

La transformation de VGG16 pour la tâche de segmentation sémantique nécessite plusieurs modifications architecturales stratégiques tout en préservant ses points forts fondamentaux. L'adaptation principale consiste à convertir les couches fully connected en couches convolutives 1x1, une modification qui permet de préserver l'information spatiale cruciale tout au long du réseau. Cette conversion s'accompagne de l'ajout de couches de déconvolution, essentielles pour retrouver la résolution d'origine de l'image.

L'introduction de connexions skip permet également de conserver les détails spatiaux fins qui risqueraient d'être perdus dans les couches profondes du réseau. Ces modifications permettent de bénéficier des poids préentraînés sur ImageNet tout en adaptant le réseau à la tâche spécifique de segmentation.

2.4 Justification du choix de VGG16

2.4.1 Avantages techniques

Sur le plan technique, VGG16 présente des atouts majeurs qui justifient son choix pour notre projet. Sa robustesse, démontrée sur de nombreuses tâches de vision par ordinateur, s'accompagne d'excellentes performances de généralisation et d'une grande stabilité lors de l'entraînement. L'architecture offre un équilibre optimal entre profondeur du réseau et efficacité computationnelle, permettant des temps d'inférence raisonnables tout en maintenant une précision élevée. La gestion efficace de la mémoire et la facilité d'implémentation sous Keras en font une solution particulièrement adaptée aux contraintes de développement.

2.4.2 Adéquation avec le projet

Dans le contexte spécifique de notre projet de segmentation pour véhicules autonomes, VGG16 répond parfaitement aux exigences établies. Son intégration transparente dans l'écosystème Keras existant facilite le développement et la maintenance du système. L'architecture s'adapte naturellement aux huit catégories principales de Cityscapes, permettant une segmentation efficace des éléments critiques des scènes urbaines. La simplicité de son architecture facilite les futures optimisations et évolutions du système, tout en maintenant un niveau de performance élevé.

2.4.3 Considérations pratiques

D'un point de vue pratique, le choix de VGG16 s'appuie sur un écosystème mature et bien documenté. La disponibilité d'une documentation extensive et le support d'une communauté active facilitent grandement le développement et la résolution des problèmes potentiels. Le processus de déploiement est simplifié par l'existence de nombreuses optimisations documentées et la possibilité d'exporter facilement le modèle vers différentes plateformes. Ces aspects pratiques s'avèrent particulièrement précieux pour l'intégration dans une API, répondant ainsi parfaitement aux besoins spécifiques de notre projet tout en facilitant les futures évolutions du système.

3 Présentation du modèle et des catégories de segmentation

3.1 Architecture du modèle

Notre approche repose sur deux architectures disctinctes : VGG16 et U-NEt, toutes deux adaptées à la tâche de segmentation sémantique. VGG16, initialement conçu pour la classification d'images, a été modifié pour la segmentation en transformant ses couches fully connected en couches convolutives. L'architecture U-Net, quant à elle, est nativement conçue pour la segmentation avec son architecture en forme de U caractéristique.

3.1.1 VGG16 adapté pour la segmentation

L'architecture VGG16 modifiée se compose de deux parties principales. L'encodeur conserve la structure originale de VGG16 avec ses cinq blocs de convolution, permettant une extraction progressive des caractéristiques. Chaque bloc augmente la profondeur des feature maps tout en réduisant leur dimension spatiale via le max pooling. Cette organisation permet une extraction hiérarchique des caractéristiques, des motifs simples aux concepts plus abstraits.

La transformation en modèle de segmentation nécessite plusieurs modifications arhictecturales. Les couches fully connected sont converties en couches convolutives 1x1, préservant l'information spatiale tout en réduisant le nombre de paramètres. L'ajout de couches de déconvolution permet ensuite de restaurer la résolution spatiale de l'image d'entrée.

3.1.2 Architecture U-NET

L'architecture U-Net présente une structure symétrique avec un chemin contractant (encodeur) et un chemin expansif (décodeur). L'encodeur capture le contexte via des convolutions et du max pooling successifs, tandis que le décodeur permet une localisation précise via des convolutions transposées. Les connexions skip entre les niveaux correspondants permettent de combiner les informations locales et globales.

3.2 Gestion du déséquilibre des classes

Une caractéristique importante de notre jeu de données est le fort déséquilibre entre les classes. Pour gérer ce problème, nous avons implémenté une pondération des classes basée sur leur fréquence d'apparition :

- **Routes et trottoirs (classe 0)** : 0.0569
- **Bâtiments (classe 1)** : 0.0145
- **Végétation (classe 2)** : 0.0264
- **Ciel (classe 3)** : 0.3206
- **Véhicules (classe 4)** : 0.0377
- **Piétons (classe 5)** : 0.1541
- **Panneaux et feux (classe 6)** : 0.3085
- **Obstacles (classe 7)** : 0.0813

Ces poids sont inversement proportionnels à la fréquence de chaque classe dans le dataset, permettant de donner plus d'importance aux classes sous-représentées lors de l'entraînement. Cette pondération est intégrée directement dans le fonction de perte via weighted_categorical_crossentropy.

3.3 Configuration de l'entraînement

L'entraînement est optimisé grâce à deux mécanismes clés :

- **Early Stopping** : Arrête l'enregistrement lorsque les performances sur l'ensemble de validation cessent de s'améliorer pendant 4 époques consécutives (patience = 4), évitant ainsi le surapprentissage et optimisant la durée d'entraînement.
- **Model Checkpoint** : Sauvegarde les meilleurs poids du modèle basés sur le score de Jaccard (IoU) de validation, assurant la conservation des performances optimales.

La fonction de perte combinée tient compte à la fois de la cross-entropy pondérée et du score de Dice, permettant une optimisation équilibrée entre précision globale et performance par classe.

4 Préparation des données et techniques d'augmentation

4.1 Dataset Cityscapes

Le dataset Cityscapes constitue une ressource précieuse pour notre projet de segmentation d'images pour véhicules autonomes. Ce jeu de données se distingue par sa richesse et sa pertinence, capturant la complexité des environnements urbains à travers des images haute résolution (2048x1024 pixels) prises dans 50 villes différentes. Les images présentent une grande diversité de scènes urbaines, incluant différentes conditions météorologiques, moments de la journée et configurations de trafic. Chaque image est accompagnée d'annotations pixelliques précises, essentielles pour l'entraînement supervisé de notre modèle de segmentation.

4.2 Prétraitement des données

Le prétraitement des données constitue une étape cruciale pour optimiser l'entraînement de notre modèle VGG16. La première étape consiste à redimensionner les images à une résolution plus adaptée au traitement en temps réel (224x224 pixels), compatible avec l'architecture VGG16. Cette réduction de taille permet d'accélérer l'entraînement et l'inférence tout en conservant suffisamment de détails pour une segmentation précise.

La normalisation des images joue également un rôle fondamental dans notre pipeline de prétraitement. Les valeurs des pixels sont normalisées dans l'intervalle [0,1] et standardisées selon les statistiques du jeu de données ImageNet sur lequel VGG16 a été préentraîné Cette standardisation assure une meilleure compatibilité avec les poids préentraînés et favorise une convergence plus stable lors de l'entraînement.

4.3 Techniques d'augmentation des données

L'augmentation des données s'avère essentielle pour améliorer la robustesse et la généralisation de notre modèle. Dans ce projet, nous avons expérimenté deux approches différentes d'augmentation des données : une utilisant la bibliothèque Albumentations et une autre basée sur TensorFlow. Pour chacune de ces approches, nous avons développé et testé une version basique puis une version étendue avec des transformations plus sophistiquées.

4.3.1 Première approche : Albumentations

La première approche testée utilise la bibliothèque Albumentations, spécialisée dans l'augmentation d'images. Dans sa version basique, nous avons implémenté les transformations géométriques fondamentales :

— **Retournement horizontal** avec une probabilité de 50% ;
— **Retournement vertical** avec une probabilité de 20% ;
— **Rotation aléatoire** dans une plage de ±10 degrés ;
— **Mise à l'échelle aléatoire** avec une variation de ±20%.

La version étendue de cette approche enrichit ces transformations avec des modifications plus complexes :

— **Ajustements de luminosité et de contraste** ;
— **Applications de flou gaussien** avec des noyaux de taille variable (3 à 7 pixels) ;
— **Modifications de teinte et de saturation** ;
— **Transformations élastiques** pour simuler des déformations non linéaires ;
— **Distorsions en grille** pour une variabilité géométrique accrue.

4.3.2 Seconde approche : TensorFlow

La seconde approche testée utilise les fonctionnalités natives de TensorFlow. La version basique comprend :

— **Retournement horizontal** avec une probabilité de 50% ;
— **Retournement vertical** avec une probabilité de 20% ;
— **Rotation par multiples de 90 degrés** ;
— **Mise à l'échelle dynamique** entre 80% et 120% de la taille originale.

La version étendue de cette approche ajoute des transformations photométriques :

— **Ajustements de luminosité** avec une variation maximale de 20% ;
— **Modificatiosn du contraste** dans une plage de 80% à 120% ;
— **Ajustements de la saturation** des couleurs (variation de ±20%) ;

— **Modifications de la teinte** avec une variation maximale de 10%.

4.3.3 Gestion des masques de segmentation

Pour les deux approches, un aspect crucial est la préservation de la cohérence entre les images transformées et leurs masques de segmentation correspondants. Chaque transformation appliquée à une image est simultanément appliquée à son masque de segmentation, maintenant ainsi l'alignement spatial essentiel pour l'entraînement du modèle.

Ces deux approches alternatives d'augmentation de données ont été testées séparément, chacune avec sa version basique et étendue, permettant d'évaluer leur impact respectif sur les performances du modèle. Cette expérimentation nous permet d'identifier la stratégie la plus efficace pour notre cas d'usage spécifique de segmentation de scènes urbaines.

5 Entraînement du modèle et évaluation initiale

5.1 Configuration d'entraînement

L'entraînement des modèles utilise Keras comme framework. Le processus s'étend sur 20 époques, avec un batch size de 4 images et l'optimiseur Adam qui permet une adaptation automatique du taux d'apprentissage. La fonction de perte utilisée est la categorical crossentropy, modifiée pour prendre en compte la pondération des classes.

Les mécanismes d'Early Stopping et de Model Checkpoint permettent respectivement d'éviter le surapprentissage et de conserver les meilleurs poids du modèle en se basant sur le score de Jaccard de validation.

5.2 Métriques d'évaluation

L'évaluation précise des performances du modèle de segmentation repose sur plusieurs métriques complémentaires, chacune apportant un éclairage différent sur la qualité de la segmentation :

- **Accuracy** : Mesure la préduction globale de la segmentation en calculant le pourcentage de pixels correctement classifiés.
- **Coefficient de Dice (Dice Coefficient)** : Cette métrique, également connue sous le nom de F1-score, mesure le degré de chevauchement entre les prédictions et les annotations réelles. Elle est particulièrement pertinente pour évaluer la qualité de la segmentation car elle prend en compte à la fois la précision et le rappel.
- **Dice Loss** : dérivée du coefficient de Dice, cette fonction de perte permet d'optimiser directement la similarité entre les masques prédits et réels. Elle est particulièrement adaptée aux problèmes de segmentation où les classes peuvent être déséquilibrées.
- **Total Loss** : Cette métrique combine différentes fonctions de perte pour une évaluation globale de la performance du modèle. Elle offre une vue d'ensemble de la qualité de l'apprentissage.
- **Jaccard Score** : Également connu sous le nom d'Intersection over Union (IoU), il mesure le rapport entre l'intersection et l'union des zones prédites et réelles. Cette métrique est particulièrement stricte car elle pénalise toute divergence entre la prédiction et la réalité..

5.3 Résultats initiaux

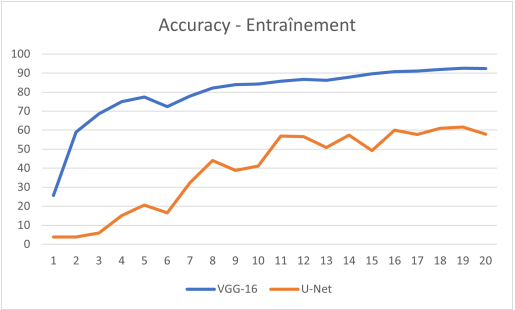
5.3.1 Comparaison VGG16 vs U-Net

Les performances de VGG16 à l'époque 18 (meilleure performance) :

- **Métriques d'entraînement :**
  - **Accuracy** : 91.94%
  - **Dice coefficient** : 89.03%
  - **Jaccard score** : 80.28%
  - **Loss** : 0.0099
- **Métriques de validation :**
  - **Accuracy** : 88.19%
  - **Dice coefficient** : 85.86%
  - **Jaccard score** : 75.24%
  - **Loss** : 0.0256

Les performances de U-Net à l'époque 20 (meilleure performance) :

- **Métriques d'entraînement :**
  - **Accuracy** : 57.94%
  - **Dice coefficient** : 48.53%
  - **Jaccard score** : 32.13%
  - **Loss** : 0.0499
- **Métriques de validation :**
  - **Accuracy** : 64.27%
  - **Dice coefficient** : 55.57%
  - **Jaccard score** : 38.57%
  - **Loss** : 0.0469



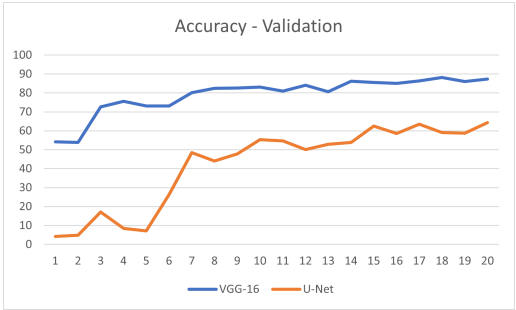
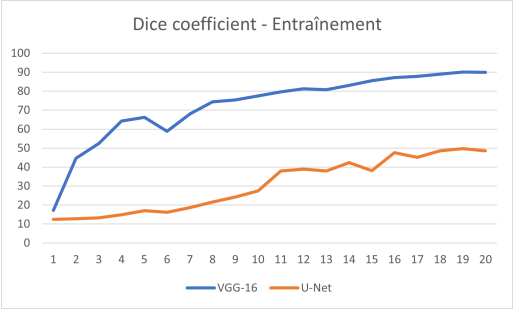


FIGURE 2 – Visualisation de l'accuracy d'entraînement et de validation



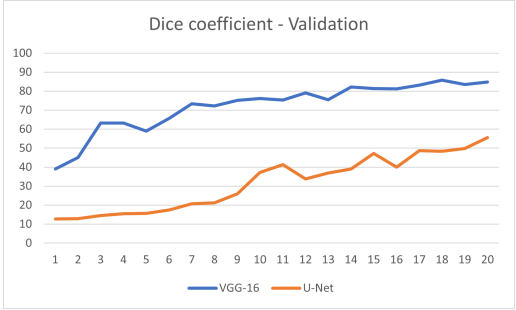
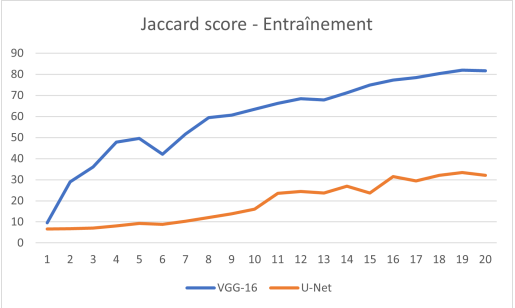


FIGURE 3 – Visualisation du dice coefficient d'entraînement et de validation



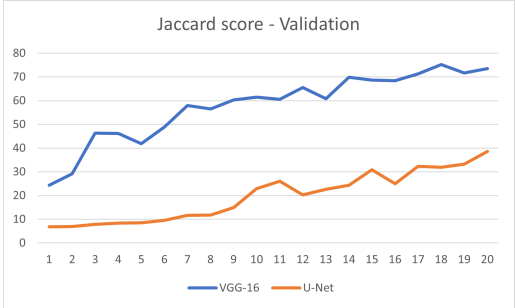
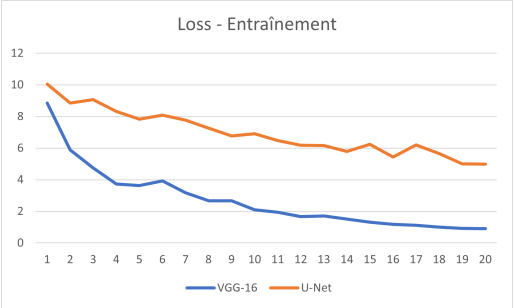


FIGURE 4 – Visualisation du jaccard score d'entraînement et de validation



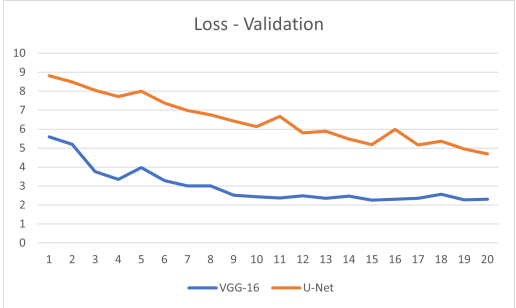


FIGURE 5 – Visualisation du loss d'entraînement et de validation

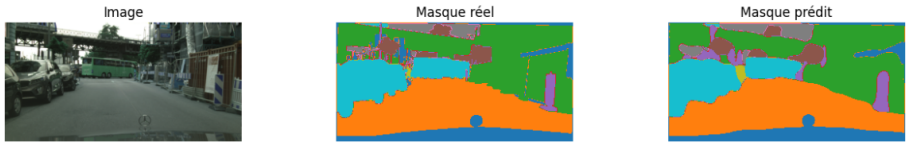


FIGURE 6 – Résultat avec VGG16 sans data augmentation

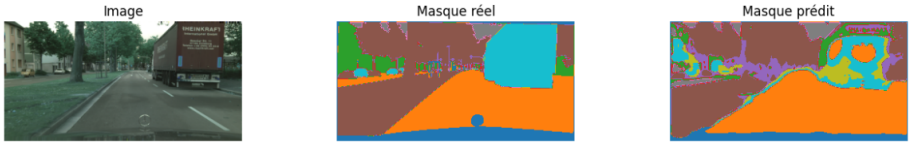


FIGURE 7 – Résultat avec U-Net sans data augmentation

5.3.2 Analyse des performances

VGG16 surpasse significativement U-Net sur toutes les métriques. Cette différence peut s'expliquer par plusieurs facteurs :

— L'architecture VGG16 bénéficie des poids pré-entraînés sur ImageNet ;
— La gestion du déséquilibre des classes via la pondération semble mieux adaptée à VGG16 ;
— La profondeur et la régularité de l'architecture VGG16 permettet une meilleure extraction des caractéristiques.

5.3.3 Impact de la pondération des classes

La pondération des classes a permis d'améliorer la segmentation des classes minoritaires. Les classes fortement pondérées (ciel : 0.3206, panneaux/feux : 0.3085) montrent des performances satisfaisantes malgré leur faible représentation dans le dataset.

6 Impact des techniques d'augmentation de données sur les résultats

6.1 Comparaison des approches d'augmentation

Deux bibliothèques ont été testées pour l'augmentation de données : Albumentations et TensorFlow, chacune avec une version basique et une version étendue.

6.1.1 Performance avec Albumentations

Version basique (meilleures performances à l'époque 20) :
- **Métriques d'entraînement :**
  - **Accuracy** : 89.26%
  - **Dice coefficient** : 85.48%
  - **Jaccard score** : 74.67%
  - **Loss** : 0.0130
- **Métriques de validation :**
  - **Accuracy** : 83.54%
  - **Dice coefficient** : 80.26%
  - **Jaccard score** : 67.07%
  - **Loss** : 0.0305

Version étendue (meilleures performances à l'époque 20) :
- **Métriques d'entraînement :**
  - **Accuracy** : 85.05%
  - **Dice coefficient** : 79.53%
  - **Jaccard score** : 66.13%
  - **Loss** : 0.0182
- **Métriques de validation :**
  - **Accuracy** : 79.38%
  - **Dice coefficient** : 73.72%
  - **Jaccard score** : 58.40%
  - **Loss** : 0.0313

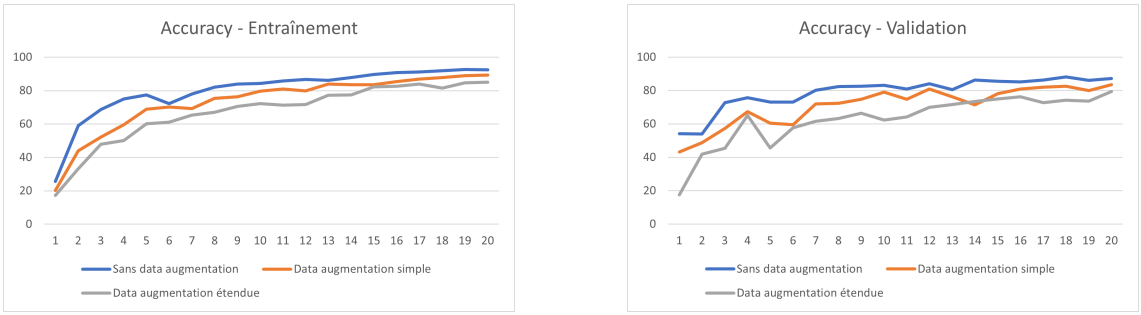


FIGURE 8 – Visualisation de l'accuracy d'entraînement et de validation avec Albumentations

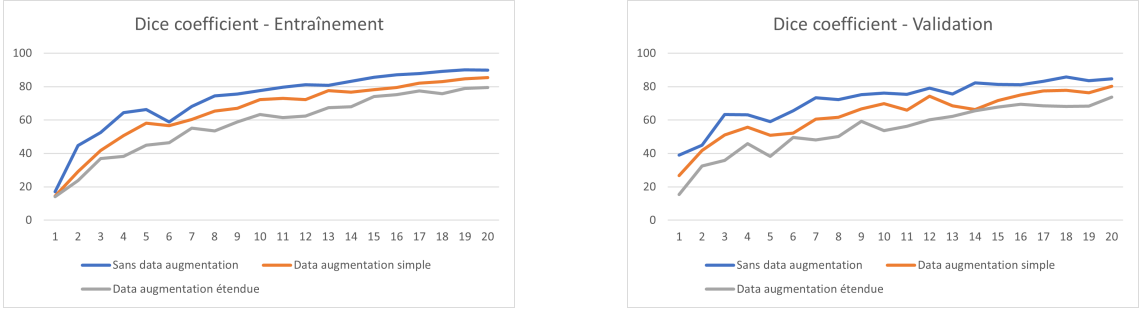


FIGURE 9 – Visualisation du dice coefficient d'entraînement et de validation avec Albumentations

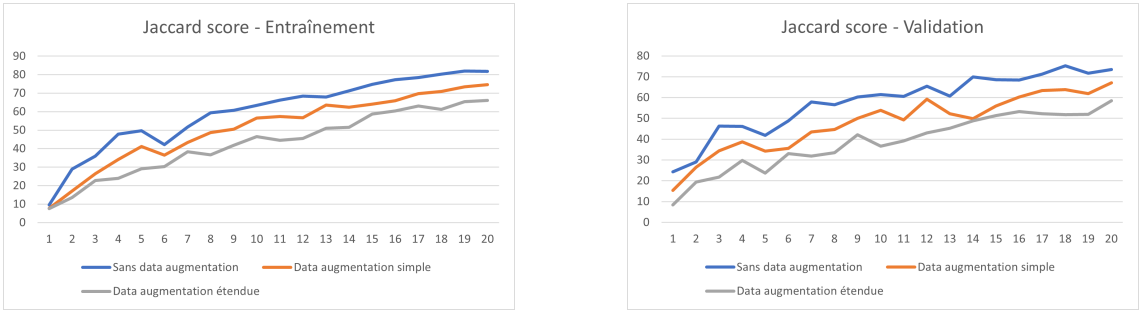


FIGURE 10 – Visualisation du jaccard score d'entraînement et de validation avec Albumentations

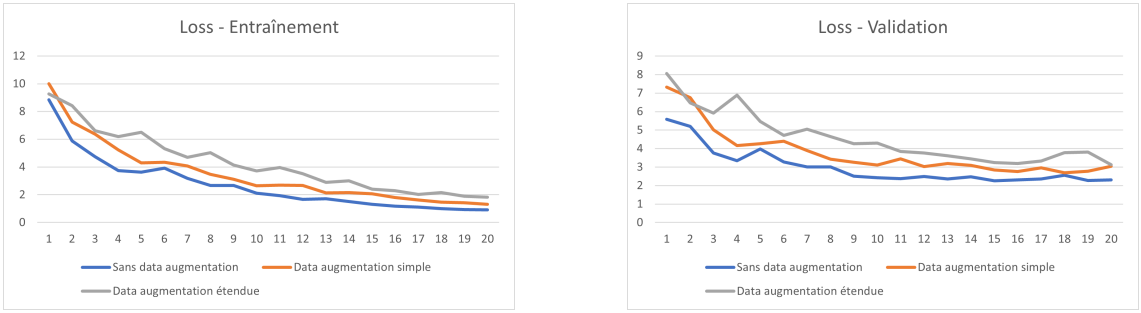


FIGURE 11 – Visualisation du loss d'entraînement et de validation avec Albumentations

L'utilisation d'Albumentations avec des transformations basiques a produit des résultats intéressants mais inférieurs au modèle sans augmentation. En effet, avec une précision de 89.26% et un coefficient de Dice de 85.48% en entraînement, et respectivement 83.54% et 80.26% en validation, ces performances montrent une légère dégradation par rapport au modèle original qui atteignait 91.94% de précision et 89.03% de coefficient de Dice en entraînement.

La version étendue d'Albumentations a montré des performances encore plus faibles, avec une précision de 85.05% et un coefficient de Dice de 79.53% en entraînement. Cette baisse de performance suggère que l'ajout de transformations plus complexes n'a pas bénéficié au modèle, potentiellement en raison d'une trop grande distorsion des caractéristiques importantes des images.

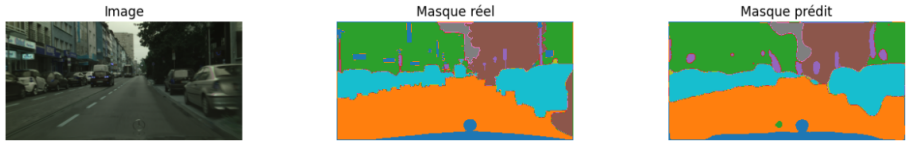


FIGURE 12 – Résultat avec Albumentations - version basique

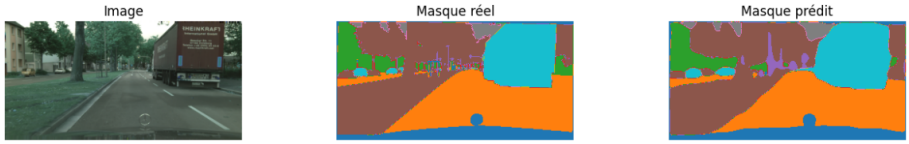


FIGURE 13 – Résultat avec Albumentations - version étendue

6.1.2 Performance avec TensorFlow

Version basique (meilleures performances à l'époque 10) :

- **Métriques d'entraînement :**
  - **Accuracy** : 20.34%
  - **Dice coefficient** : 17.73%
  - **Jaccard score** : 9.73%

    - **Loss** : 0.0848
- **Métriques de validation :**
    - **Accuracy** : 25.14%
    - **Dice coefficient** : 18.35%
    - **Jaccard score** : 10.11%
    - **Loss** : 0.0798

Version étendue (meilleures performances à l'époque 5) :

- **Métriques d'entraînement :**
    - **Accuracy** : 19.41%
    - **Dice coefficient** : 13.77%
    - **Jaccard score** : 7.3%
    - **Loss** : 0.0946
- **Métriques de validation :**
    - **Accuracy** : 17.5%
    - **Dice coefficient** : 14.21%
    - **Jaccard score** : 7.65%
    - **Loss** : 0.0868

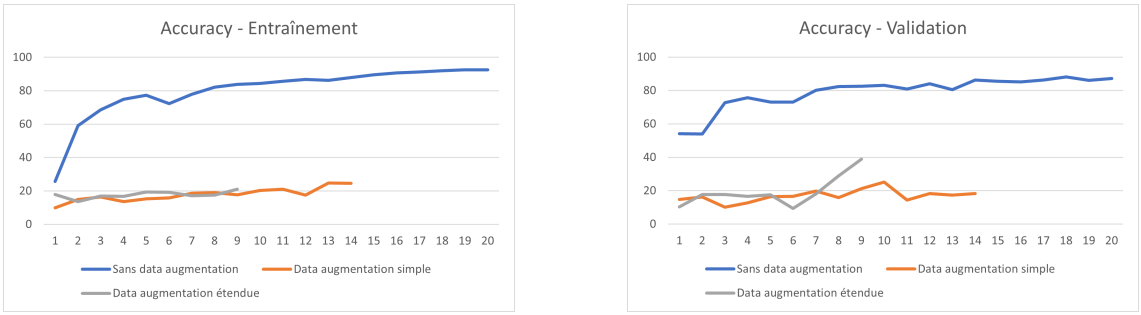


FIGURE 14 – Visualisation de l'accuracy d'entraînement et de validation avec TensorFlow

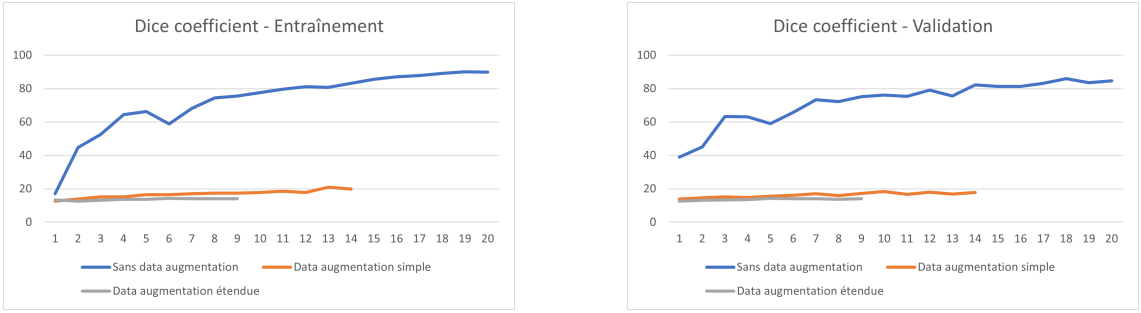


FIGURE 15 – Visualisation du dice coefficient d'entraînement et de validation avec TensorFlow

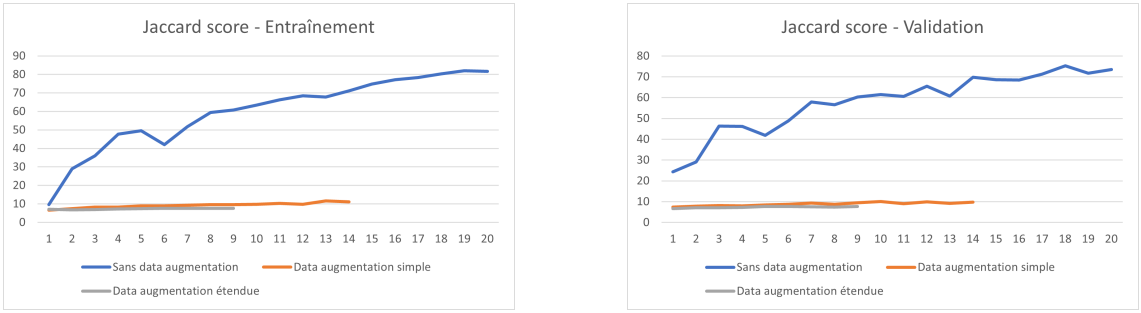


FIGURE 16 – Visualisation du jaccard score d'entraînement et de validation avec TensorFlow

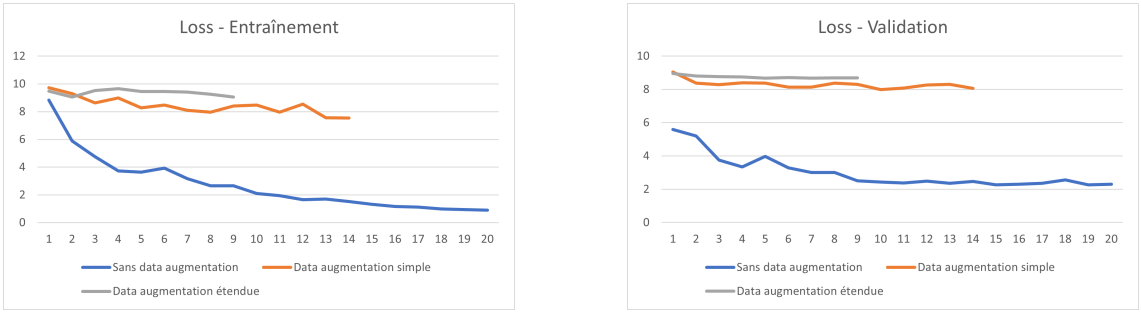


FIGURE 17 – Visualisation du loss d'entraînement et de validation avec TensorFlow

Les résultats obtenus avec TensorFlow sont significativement inférieurs, tant pour la version basique qu'étendue. La version basique atteint seulement 20.34% de précision et 17.73% de coefficient de Dice en entraînement, des résultats très en deçà des performances sans augmentation. La version étendue montre des performances encore plus faibles, avec 19.41% de précision et 13.77% de coefficient de Dice, suggérant que l'approche TensorFlow n'est pas adaptée à notre cas d'usage.

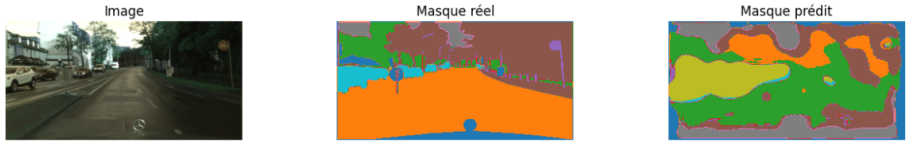


FIGURE 18 – Résultat avec TensorFlow - version basique

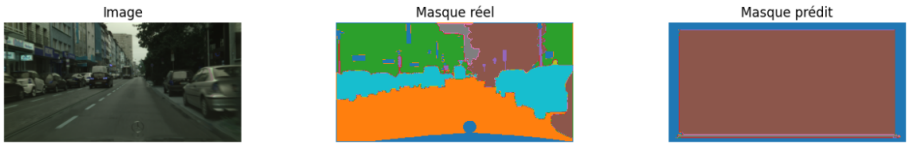


FIGURE 19 – Résultat avec TensorFlow - version étendue

6.2 Analyse des résultats

L'analyse approfondie des résultats révèle plusieurs points importants. Bien qu'Albumentations surpasse clairement TensorFlow dans nos expérimentations, aucune des approches d'augmentation de données n'a permis d'améliorer les performances du modèle initial. Cela pourrait s'expliquer par la qualité déjà élevée de notre dataset original et l'efficacité de notre stratégie de pondération des classes.

Les transformations géométriques et photométriques, plutôt que d'enrichir l'apprentissage, semblent avoir introduit un bruit nuisible dans les données d'entraînement.

6.3 Implications pratiques

Ces résultats ont des implications importantes pour notre projet. Ils démontrent que l'augmentation de données n'est pas toujours bénéfique, particulièrement lorsque le dataset initial est déjà bien construit et que des mécanismes comme la pondération des classes sont en place. La complexité supplémentaire introduite par l'augmentation de données peut même s'avérer contre-productive.

Pour notre cas d'usage spécifique de segmentation d'images urbaines, la solution la plus efficace consiste à utiliser le modèle VGG16 avec la pondération des classes, sans augmentation de données. Cette approche plus simple offre non seulement de meilleures performances mais présente aussi l'avantage d'être plus légère en termes de pipeline de traitement.

7 Résultats et analyse des performances

7.1 Synthèse des résultats

La comparaison des différentes approches testées révèle que VGG16 sans augmentation de données offre les meilleures performances globales. Le modèle atteint 91.94% de précision et un coefficient de Dice de 89.03% en entraînement, avec des performances en validation de 88.19% de précision et 85.86% de coefficient de Dice. L'architecture U-Net montre des performances significativement inférieures, tandis que les tentatives d'augmentation de données, que ce soit via Albumentations ou TensorFlow, n'ont pas permis d'améliorer ces résultats.

7.2 Impact de la pondération des classes

La stratégie de pondération des classes s'est révélée particulièrement efficace. Les classes fortement pondérées comme le ciel (0.3206) et les panneaux/feux (0.3085) montrent une bonne segmentation malgré leur sous-représentation dans le dataset. Les classes avec une pondération plus faible comme les bâtiments (0.0145) et la végétation (0.0264) maintiennent également des performances satisfaisantes, démontrant l'efficacité de cette approche pour gérer le déséquilibre des classes.

7.3 Analyse des forces et faiblesses

7.3.1 Points forts

— Excellente performance globale du modèle VGG16 sans augmentation ;
— Efficacité de la pondération des classes pour gérer le déséquilibre ;
— Bonne généralisation comme en témoignent les scores de validation proches des scores d'entraînement ;
— Convergence rapide avec le mécanisme d'Early Stopping.

7.3.2 Limitations identifiées

— Performances limitées des approches d'augmentation de données ;
— Écart significatif entre VGG16 et U-Net, suggérant une sensibilité à l'architecture choisie ;
— Complexité accrue sans gain de performance lors de l'utilisation de transformations avancées.

7.4 Perspectives pour le projet

Ces résultats suggèrent une voie claire pour l'intégration du système dans la chaîne de traitement de Future Vision Transport. Le modèle VGG16 avec pondération des classes, sans augmentation de données, offre le meilleur compromis entre performance et complexité. Cette approche permet une segmentation précise des scènes urbaines tout en maintenant une architecture relativement simple et efficace.

8 Conclusion et pistes d'amélioration

8.1 Résumé des contributions

Ce projet de segmentation d'images pour Future Vision Transport a permis d'établir une solution robuste pour la compréhension de scènes urbaines. L'utilisation de VGG16, combinée à une stratégie efficace de pondération des classes, a permis d'atteindre des performances remarquables avec une précision de 91.94% et un coefficient de Dice de 89.03%. La comparaison avec U-Net et l'exploration de différentes techniques d'augmentation de données ont confirmé la supériorité de l'approche retenue.

8.2 Limites et défis restants

Malgré ces résultats encourageants, plusieurs limitations ont été identifiées :
— Les tentatives d'augmentation de données n'ont pas amélioré les performances, suggérant des limites dans notre approche de transformation d'images.
— L'écart significatif de performance entre VGG16 et U-Net indique une possible sensibilité à l'architecture choisie.
— Certaines classes minoritaires, malgré la pondération, pourraient bénéficier d'optimisations supplémentaires

8.3 Pistes d'amélioration

8.3.1 Optimisation du modèle

Des optimisations pourraient être envisagées :
— Explorer d'autres architectures modernes comme DeepLabV3+ ou PSPNet ;
— Expérimenter avec différentes stratégies de pondération des classes ;
— Investiguer des techniques d'apprentissage par transfert plus avancées.

8.3.2 Enrichissement des données

Pour améliorer davantage les performances du modèle, plusieurs pistes peuvent être explorées :
— Envisager l'utilisation d'autres jeux de données urbains pour diversifier l'apprentissage ;
— Développer des techniques d'augmentation plus spécifiques aux scènes urbaines ;
— Explorer des approches d'apprentissage semi-supervisé pour exploiter des données non annotées.

8.3.3 Optimisations techniques

Des améliorations techniques pourraient être envisagées :
— Optimiser le modèle pour l'inférence en temps réel ;
— Améliorer la gestion mémoire pour permettre des batchs plus importants ;
— Investiguer des techniques de quantification pour l'optimisation des performances.