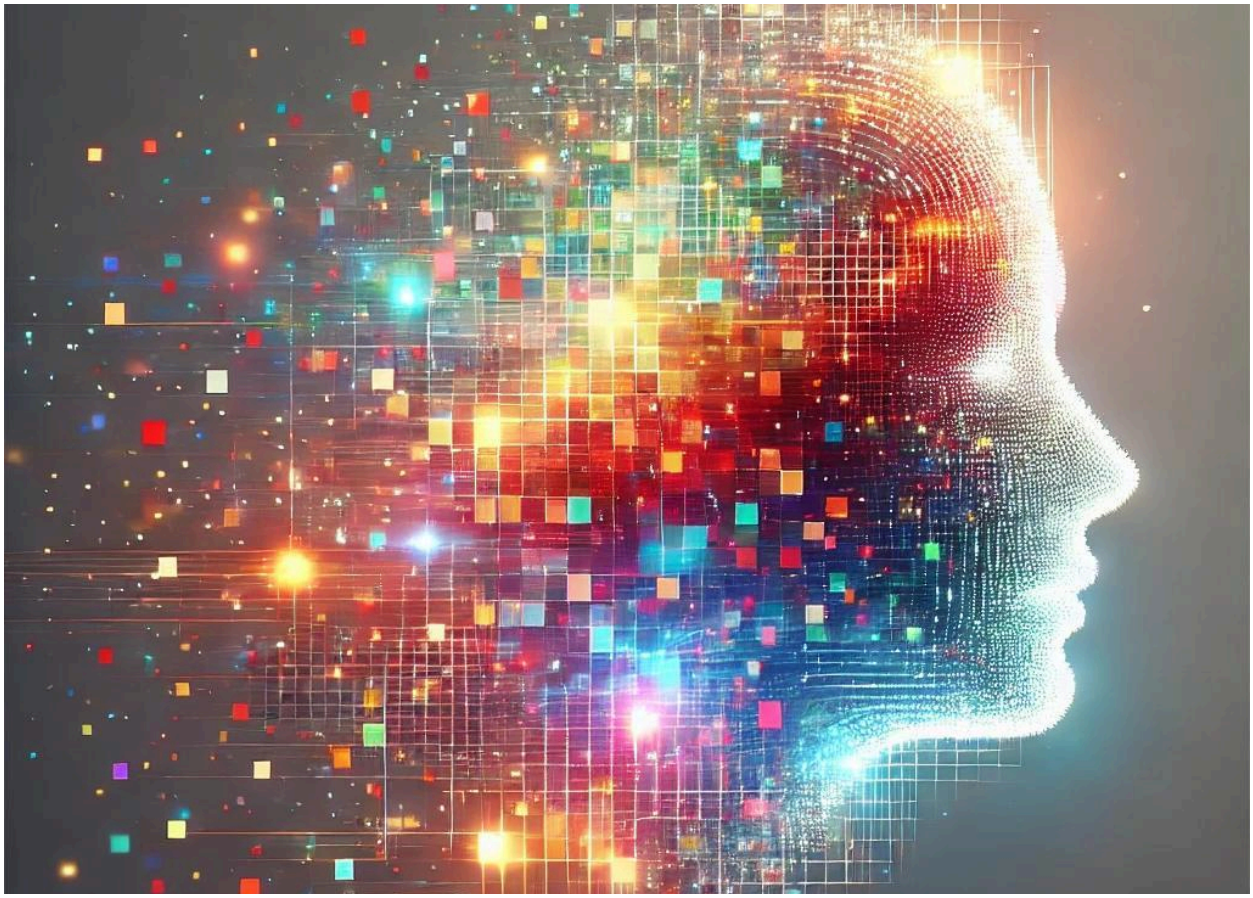
Projet de segmentation d'images
Note technique



Introduction

Cette note technique présente les étapes clés et les résultats obtenus dans le cadre du développement d'un modèle de segmentation d'images. Elle détaille les approches méthodologiques adoptées, les modèles testés, les performances observées et les améliorations envisageables. Ce document vise à synthétiser le travail réalisé, en mettant en avant les choix stratégiques et les apprentissages, tout en ouvrant des perspectives pour de futures optimisations.

I. Synthèse de l'état de l'art

Les approches modernes de segmentation d'images reposent sur des architectures de réseaux neuronaux convolutifs (CNN). Voici un résumé des principales techniques :

- U-Net : Une architecture largement adoptée pour les tâches de segmentation, caractérisée par son équilibre entre extraction de caractéristiques et résolution spatiale.
- DeepLab : Une autre architecture populaire qui utilise des convolutions atrous pour capturer des caractéristiques à différents niveaux de résolution.
- SegNet : Connu pour sa simplicité et son efficacité, il se concentre sur la récupération de la résolution spatiale tout en étant léger.

- Transfer Learning : L’utilisation d’encodeurs pré-entraînés, comme VGG16, ResNet ou EfficientNet, optimise les performances en exploitant des représentations pré-apprises. Cela réduit le temps d’entraînement et améliore les résultats sur de petits ensembles de données.

Nouvelles avancées (2024-2025)

- Segment Anything Model (SAM) : Introduit par Meta AI, ce modèle universel de segmentation peut traiter n'importe quelle image ou vidéo avec des performances remarquables. Il s’appuie sur des bases de données massives et une conception optimisée pour des tâches complexes.
- Vision Transformers (ViT) : Appliqués à la segmentation, les ViTs permettent de capturer des relations globales entre pixels, offrant des performances accrues sur des images haute résolution.
- Modèles allégés pour systèmes embarqués : Des modèles comme MobileNet ou Tiny-ViT sont spécifiquement conçus pour une intégration dans des environnements contraints en ressources.

II. Approches développées

A. Préparation des données

Les images et masques sont regroupés en 8 catégories :

1. Void
2. Flat
3. Construction
4. Objects
5. Nature
6. Sky
7. Human
8. Vehicles

Des vérifications ont été effectuées pour éviter les fuites de données et assurer la cohérence entre les ensembles d'entraînement, validation et test.

B. Modèles implémentés

Pour mieux comprendre les performances de différentes architectures et leur potentiel dans un contexte de segmentation d'images embarquée, nous avons entraîné deux variantes principales :

1. U-Net standard :

   Ce modèle repose sur une architecture classique en forme de "U", composée d'un encodeur pour extraire les caractéristiques, d'un goulot d'étranglement pour capturer des représentations globales, et d'un décodeur pour reconstruire les masques segmentés.

   La simplicité de cette architecture permet une mise en œuvre rapide, tout en offrant une base solide pour les comparaisons.

   Nous avons entraîné ce modèle d'abord sans augmentation de données, pour évaluer ses performances de base. Ensuite, nous avons intégré des techniques d'augmentation de données (flip horizontal, ajustements de luminosité, contraste) pour tester l'impact sur la généralisation.

2. U-Net avec VGG16 :

   Cette variante intègre un encodeur pré-entraîné VGG16 dans la partie descendante de l'U-Net, tirant parti du transfer learning pour capturer des caractéristiques plus riches dès les premières couches.

   Cette approche est particulièrement bénéfique lorsque le jeu de données est limité en volume, car elle réduit le besoin d'un entraînement exhaustif sur les premières couches.

   Comme pour le modèle standard, nous avons comparé les résultats sans et avec augmentation de données afin de quantifier les améliorations.

Ces deux modèles ont été choisis pour leur complémentarité : le premier offrant une architecture basique et rapide, le second explorant les bénéfices du transfer learning.

C. Techniques d'augmentation des données

Les augmentations utilisées dans ce projet incluent :

- Flip horizontal : Permet de simuler les variations dans l'orientation des scènes capturées, utile pour les environnements urbains où les objets sont souvent symétriques.
- Ajustements de luminosité et contraste : Ces transformations rendent le modèle plus robuste aux conditions d'éclairage variables (par exemple, ombres ou changements soudains de luminosité).

Ces choix ont été guidés par les spécificités des données et des contraintes du projet. Les environnements routiers, en particulier, sont sujets à des variations importantes en termes d'éclairage et d'angle de vue. En intégrant ces augmentations, nous avons cherché à maximiser la capacité de généralisation du modèle et à améliorer sa robustesse face à des données complexes et non idéales.

III. Résultats obtenus

A. Les métriques

Dans ce projet, plusieurs métriques clés ont été utilisées pour évaluer les performances des modèles. Le **Dice Coefficient** a été choisi pour mesurer la similarité entre les masques prédits et les masques cibles, en mettant l'accent sur une bonne segmentation des classes, y compris celles moins représentées. L'**IoU** (Intersection over Union) complète cette mesure en évaluant la précision globale des prédictions en fonction du chevauchement entre les zones prédites et les zones réelles. Ces deux métriques permettent une évaluation fine et équilibrée des résultats.

La fonction de perte principale utilisée dans ce projet est une combinaison de la **Binary Crossentropy Loss et du Dice Loss**, appelée dice_bce_loss. Cette fonction de coût est

utilisée pour guider l'entraînement du modèle en optimisant simultanément deux aspects clés : la Binary Crossentropy Loss, qui mesure l'écart entre les prédictions et les valeurs cibles au niveau des pixels, et le Dice Loss, qui favorise une meilleure correspondance structurelle entre les masques prédits et les masques réels. Pendant l'entraînement, la valeur de **val_loss** représente cette fonction de coût appliquée à l'ensemble de validation. Une diminution de la val_loss au fil des époques indique que le modèle parvient à mieux s'adapter aux données tout en maintenant une bonne capacité de généralisation. Ces choix méthodologiques ont été essentiels pour suivre efficacement les progrès du modèle et garantir sa robustesse dans des environnements variés.

B. Performance des modèles

Plusieurs configurations de réseaux de type U-Net ont été évaluées sur le jeu de données, en jouant à la fois sur la présence ou l'absence d'augmentations de données, et sur l'utilisation d'une backbone VGG16 préentraînée. Les principales métriques de segmentation suivies sont le Dice coefficient, l'IoU, ainsi que val_loss.

U-Net sans augmentations :

Après 10 époques, le modèle atteint un Dice Coefficient d'environ 0,658 et un IoU de 0,566 sur l'ensemble de validation, avec une perte validation (val_loss) autour de 0,3905.

Le modèle montre une bonne capacité à segmenter les images, atteignant un Dice Coefficient de 0,658 pour l'entraînement et 0,645 pour la validation. Bien que l'on observe une légère différence entre ces deux valeurs, cela est attendu dans tout processus d'entraînement et reste dans une plage raisonnable. Cette variation pourrait simplement refléter des différences normales entre les données d'entraînement et de validation, sans indiquer un problème majeur d'overfitting. Les résultats suggèrent que le modèle est capable de généraliser correctement tout en exploitant efficacement les données d'entraînement.

Le temps d'entraînement total dépasse 17 400 secondes, dû au nombre important d'itérations et à la taille conséquente des images.

U-Net avec augmentations :

En appliquant des techniques d'augmentation de données, on observe une amélioration notable de la généralisation : le Dice monte jusqu'à ≈0,701 et l'IoU atteint environ 0,610 en validation (val_loss≈0,307).

L'augmentation a manifestement réduit l'overfitting : la performance en validation est plus proche de la performance en entraînement, rendant le modèle plus robuste.

Le temps d'entraînement, autour de 17 200 secondes, est comparable à celui sans augmentation. L'effort supplémentaire de prétraitement est compensé par le fait que le modèle apprend plus vite des exemples plus variés, ce qui conduit souvent à un trade-off neutre ou même légèrement positif sur le temps total.

U-Net avec backbone VGG16, sans augmentations :

L'intégration de VGG16 comme backbone se révèle particulièrement pertinente pour l'extraction de caractéristiques riches, conduisant dès les premières époques à des performances plus élevées. Après 10 époques, on mesure un Dice sur validation de ≈0,710 et un IoU de ≈0,615 (val_loss≈0,2966).

Comparativement au U-Net "classique" sans augmentations, l'architecture VGG16 réduit davantage la perte en validation et obtient une meilleure segmentation globale (Dice et IoU supérieurs), tout en affichant un temps d'entraînement plus court (≈15 270 secondes).

L'effet d'overfitting est encore moins prononcé que dans le cas U-Net simple ; on constate en effet que la courbe de validation suit mieux la courbe d'entraînement, grâce à la plus grande capacité du réseau à généraliser les formes et textures dans les images.

U-Net avec backbone VGG16 et augmentations :

Cette configuration est la plus avancée : elle combine à la fois l'efficacité de la backbone VGG16 et la généralisation offerte par l'augmentation de données.

Les performances en validation atteignent leur niveau le plus élevé, avec un Dice culminant à ≈0,725 et un IoU autour de 0,631 (val_loss≈0,2763), soit la meilleure segmentation parmi toutes les expériences.

Le temps d'entraînement reste lui aussi le plus court parmi toutes les configurations (environ 15 145 secondes), confirmant l'intérêt d'une architecture puissante et préentraînée.

L'overfitting s'avère ici très limité, car la validation suit de près l'apprentissage, ce qui montre une bonne capacité du réseau à s'adapter à la diversité des images, tout en évitant de mémoriser trop spécifiquement l'ensemble d'entraînement.

En résumé, les résultats confirment que la combinaison d'une backbone VGG16 et d'une data augmentation soignée est un levier puissant pour améliorer la segmentation. Les métriques de performance (Dice et IoU) sont significativement plus élevées dans cette configuration finale, et la perte en validation est la plus faible. Sur la question de l'overfitting, on constate que l'augmentation de données permet effectivement de réduire l'écart entre les performances d'entraînement et de validation, rendant le modèle plus robuste. Enfin, malgré l'ajout de la VGG16 et des augmentations, le temps d'entraînement n'augmente pas et peut même se retrouver optimisé grâce aux capacités de généralisation et à l'extraction de caractéristiques plus pertinentes, comme le démontre l'expérience.

C. Analyse comparative

Modèle	IoU moyen	Dice Coeff.	val_loss	Temps d'entraînement
U-Net sans augmentations	0.566	0.658	0.3905	~4h 50min
U-Net avec augmentations	0.610	0.701	0.307	~4h 47min
U-Net VGG16 sans augmentations	0.615	0.710	0.2966	~4h 15min
U-Net VGG16 avec augmentations	0.631	0.725	0.2763	~4h 12min

IV. Architecture retenue

L'architecture U-Net avec VGG16 a été retenue en raison de ses performances remarquables, notamment dans la configuration avec augmentation de données. Ce choix s'explique par plusieurs facteurs mis en avant dans les sections précédentes :

- Efficacité des caractéristiques pré-entraînées : Grâce au backbone VGG16, le modèle bénéficie d'une extraction de caractéristiques plus riche et plus pertinente, ce qui améliore significativement la précision des masques segmentés.
- Robustesse face aux données variées : L'intégration des techniques d'augmentation a permis de réduire l'overfitting, rendant le modèle plus apte à généraliser sur des environnements variés tout en maintenant des performances élevées.
- Rapidité relative : Malgré la puissance supplémentaire apportée par VGG16, le temps d'entraînement est resté compétitif, notamment grâce à une convergence plus rapide due à l'apprentissage préalable de caractéristiques utiles.

Les caractéristiques principales du modèle final incluent :

- Entrées : Images 512x256
- Sorties : Masques de segmentation avec 8 classes
- Métriques suivies : IoU, Dice Coefficient, val_loss

Ce choix d'architecture reflète un compromis optimal entre performance, robustesse et faisabilité dans le cadre des contraintes imposées par l'intégration dans un système embarqué.

V. Conclusion et pistes d'amélioration

A. Conclusion

En conclusion, les travaux réalisés dans ce projet ont permis de démontrer l'efficacité d'une approche combinant une architecture U-Net avec un encodeur VGG16 pré-entraîné et des techniques d'augmentation de données adaptées. Cette configuration s'est avérée particulièrement performante pour répondre aux exigences de segmentation d'images

dans des systèmes embarqués, atteignant des niveaux élevés de Dice Coefficient et d'IoU tout en minimisant la perte en validation.

B. Pistes d'amélioration

Pour aller plus loin, plusieurs axes de recherche et d'optimisation peuvent être explorés :

1. Exploration d'autres architectures avancées : Tester des modèles comme DeepLab ou des variantes légères basées sur MobileNet ou Tiny-ViT pourrait offrir des performances comparables avec des contraintes de calcul réduites, essentielles pour des systèmes embarqués.
2. Optimisation des hyperparamètres : Une recherche systématique des hyperparamètres à l'aide de techniques comme le grid search ou le Bayesian optimization pourrait améliorer les performances globales du modèle.
3. Quantification et compression des modèles : Réduire la taille du modèle via des techniques comme la quantification, la distillation de connaissances ou la compression des poids permettrait de faciliter son déploiement sur des dispositifs aux ressources limitées.
4. Augmentation de la diversité des données : L'ajout de données simulant des conditions extrêmes (brouillard, faible luminosité, occlusions) pourrait rendre le modèle encore plus robuste dans des environnements réels variés.
5. Suivi en continu des avancées technologiques : Intégrer les progrès récents, tels que les modèles universels comme le Segment Anything Model (SAM) ou les Vision Transformers (ViT), pourrait ouvrir de nouvelles perspectives pour améliorer encore la précision et la flexibilité du modèle.

En synthèse, ces pistes offrent un cadre pour continuer à améliorer le modèle tout en répondant aux besoins spécifiques des applications industrielles et embarquées.