



ADVANCED METHODS FOR IMAGE PROCESSING
RAPPORT - PROJET

SEGMENTATION D'IMAGES DE RUES.

Corentin SEUTIN
Iban OYHARCABAL

Janvier 2025

GitHub : <https://github.com/Argitoon/Projet-Am4ip>

Etablissement / Formation : Université de Bordeaux - Master 2 informatique

Table des matières

| | | |
|------------|---|-----------|
| I | Introduction | 2 |
| II | Modèles utilisés | 2 |
| 1 | U-Net | 2 |
| 2 | PSPNet | 3 |
| III | Débruitage | 4 |
| 3 | Iterative Boost Convolutional LSTM Network (IBCLN) | 4 |
| 4 | U-Net | 6 |
| IV | Expérimentations | 8 |
| 5 | Datasets | 9 |
| 6 | Matériel | 9 |
| 7 | Evaluation | 9 |
| 8 | Résultats | 10 |
| 9 | Analyse et Conclusion | 12 |

Première partie

Introduction

La segmentation automatique d'images est une tâche cruciale dans de nombreux domaines. Elle permet en outre de classifier et donc de distinguer les différents éléments d'une image. Cette tâche pourrait aider notamment dans la conduite automatique de véhicules ou encore dans la reconnaissance d'objets dans le domaine de la robotique comme énoncé par Zhao et al.[7]. L'utilisation des Deep Neural Networks (DNNs) et notamment des architectures U-Net[3][8][4] sont largement et majoritairement les plus utilisées pour cette tâche-là. Nous avons ainsi décidé d'utiliser l'architecture U-Net classique que nous avons allégé ainsi que l'architecture PSPNet[7] pour comparer les différents résultats obtenus.

Deuxième partie

Modèles utilisés

Il existe de nombreux modèles de réseaux de neurones pour la segmentation d'images. Pour notre projet, nous avons choisi d'implémenter les modèles suivants : U-Net et PSPNet.

1 U-Net

Durant nos recherches, nous avons déterminé que l'architecture U-Net est l'une des plus utilisées pour la segmentation. Nous avons donc décidé de l'utiliser dans notre projet. Nous avons ainsi implémenté une version simplifiée de cette architecture. Similairement à ce qui est présenté dans la Figure 1, la différence majeure est le nombre de *MaxPooling* et de *UpSampling* effectués qui sont au nombre de deux chacun pour notre modèle.

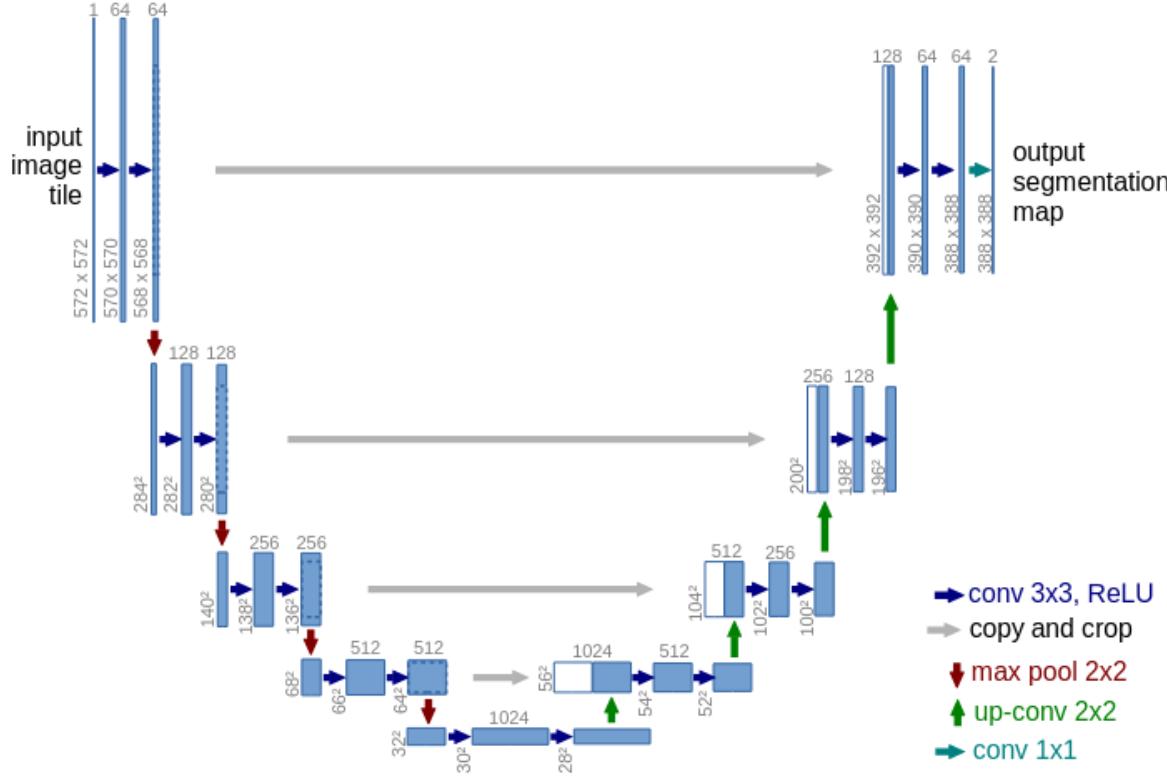


FIGURE 1 – Architecture U-Net, source : [3], Fig. 1.

Le nombre de canaux de sortie de notre modèle est égal au nombre de classes à prédire, dans notre cas **34**. Cela permet en outre d'avoir en sortie les cartes des probabilités pour chaque classe. Il suffit enfin de prendre le *argmax* de ces probabilités, i.e., de choisir pour chaque pixel la classe la plus probable pour obtenir l'image finale prédite.

2 PSPNet

L'architecture PSPNet implémentée est composée de quatre parties dont un réseau de base, un module *Pyramid Pooling*, une concaténation et une convolution finale comme présenté dans la Figure 2.

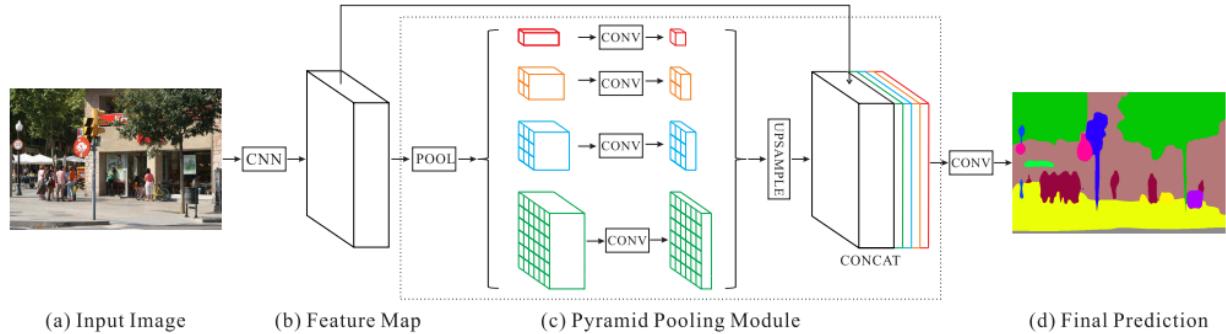


FIGURE 2 – Architecture PSPNet, source : [7], Figure 3.

Cette architecture est simple mais reste différente des architectures U-Nets classiques. Elle nous permettra ainsi de comparer les résultats entre deux modèles structurellement différents.

Troisième partie Débruitage

Dans le but de réduire le bruit présent dans le dataset et notamment dans le dataset *rainy*, nous avons décidé d'essayer deux stratégies différentes. La première est d'utiliser un modèle pré-entraîné pour enlever les reflets d'une image, c'est l'Iterative Boost Convolutional LSTM Network (IBCLN). La seconde est de créer un faux dataset avec des halos lumineux et d'entraîner un modèle type U-Net à débruiter les images de ce faux dataset.

3 Iterative Boost Convolutional LSTM Network (IBCLN)

Li et al.[2] proposent dans leur article l'architecture IBCLN permettant d'enlever les reflets générés par une surface en verre. Elle semble être l'architecture idéale compte tenu du fait que les halos lumineux et autres reflets liés à une surface en verre, le pare-brise, apparaissent dans notre dataset. Comme nous pouvons le constater sur la Figure 3, l'architecture IBCLN se décompose en deux ConvLSTM, un pour la transmission (G_T) et un pour la réflexion (G_R).

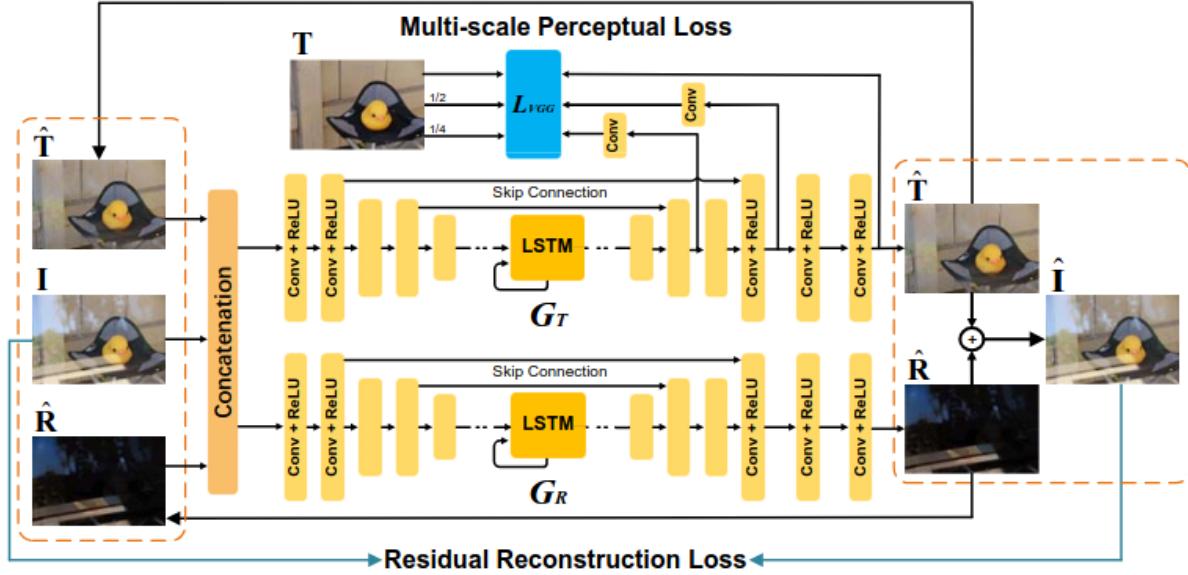


FIGURE 3 – Architecture IBCLN, source : [2], Figure 2.

Ainsi, en sortie de ces deux réseaux nous pouvons récupérer les décompositions de transmission et de réflexion d'une image donnée en entrée. L'image de transmission peut donc être perçue comme l'image débruitée, i.e., l'image pour laquelle nous enlevons la composante de réflexion et donc les reflets relatifs à une vitre.

Comme montré sur la Figure 4, le modèle pré-entraîné et le code fourni par Li et al.[1] permet un débruitage partiel des images du dataset.



FIGURE 4 – De gauche à droite : image d'origine, image débruitée à l'aide du modèle IBCLN pré-entraîné.

On peut alors en effet remarquer que le modèle IBCLN permet de réduire la luminosité des halos lumineux et certains reflets comme les reflets des phares sur la route. Cependant, l'atténuation des reflets reste minime et nous porterons donc une attention particulière par la suite à la comparaison des résultats entre images bruitées et images débruitées (par extrapolation).

4 U-Net

En analysant les images, nous pouvons remarquer que le bruit le plus présent est lié aux sources de lumières qui se reflètent sur le pare-brise comme nous le montre la Figure 5.



FIGURE 5 – Exemple d’image où le bruit induit par les sources lumineuses impacte grandement la qualité de l’image et donc de la segmentation.

Ce qui différencie ce bruit des autres types de bruit, comme les reflets sur la route, est que les pixels aux alentours des sources de lumières, i.e., dans un rayon r donné, possèdent une valeur RGB généralement très élevée voir même saturée et donc très proche du blanc. Cette saturation couplée au fait que les sources lumineuses se situent devant un objet rend le mélange des couleurs avec l’objet en arrière plan compliqué voir impossible. Ainsi, en fonction de l’intensité lumineuse de la source, une partie de l’objet en arrière plan est cachée et le mélange des couleurs devient de plus en plus équitable lorsqu’on s’éloigne de la source. Au contraire, les reflets ont un plus faible impact sur le mélange des couleurs de par leur nature physique. En effet, lorsqu’un faisceau lumineux frappe un objet, une partie de son énergie est réfléchie, une autre est absorbée et une dernière peut être transmise. Autrement dit, l’énergie réfléchie (les reflets) est plus faible que l’énergie incidente (les sources lumineuses et les halos) ce qui explique cette différence d’impact. Ainsi, nous avons décidé de nous intéresser au débruitage de la saturation et des halos induits par les sources lumineuses plutôt qu’aux reflets.

Pour ce faire, nous avons décidé d’utiliser à nouveau la même architecture U-Net que pour la segmentation. La seule différence notable est le nombre de **channels** qui est ici au nombre de trois (RGB) au lieu de 34. Ainsi, nous pouvons directement récupérer l’image débruitée en sortie du modèle.

Le dataset utilisé a été proposé par Wu et al.[6] et regroupe similairement au projet plusieurs sous-datasets : Daytime-Sunny, Daytime-Foggy, Dusk-Rainy, Night-Rainy et Night-Sunny. Nous avons ainsi pu utiliser le sous-dataset Night-Sunny pour créer un nouveau dataset en créant automatiquement des images bruitées, i.e., en ajoutant des sources lumineuses et des halos de lumières les plus réaliste possible. La Figure 6 présente un exemple d’image bruitée générée.

Le bruit généré se décompose en deux parties. La première crée une source lumineuse d’un

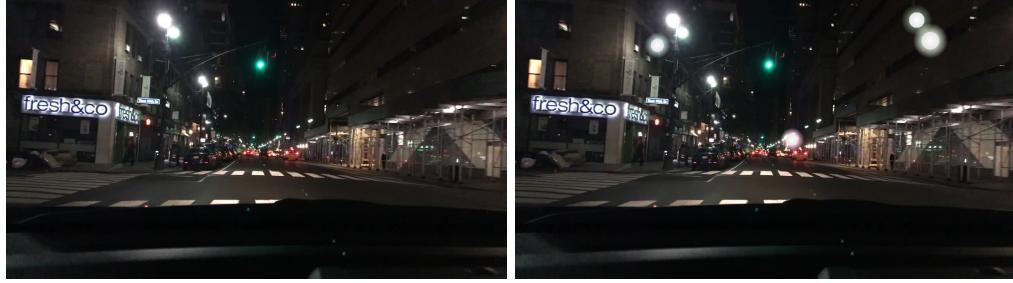


FIGURE 6 – De gauche à droite : image d'origine, image bruitée automatiquement.

certain rayon et d'une certaine couleur sans transparence mais ayant une intensité diminuant en fonction du rayon. La seconde créé le halo lumineux autour de la source en partant du rayon défini précédemment jusqu'à une certaine distance. Ce halo possède contrairement à la source créée une transparence et une intensité beaucoup moins élevée. Ainsi, en entraînant le modèle sur ces images, nous avons essayé de les débruiter. La Figure 7 illustre le débruitage effectué sur une image bruitée automatiquement mais non utilisée pour l'entraînement.

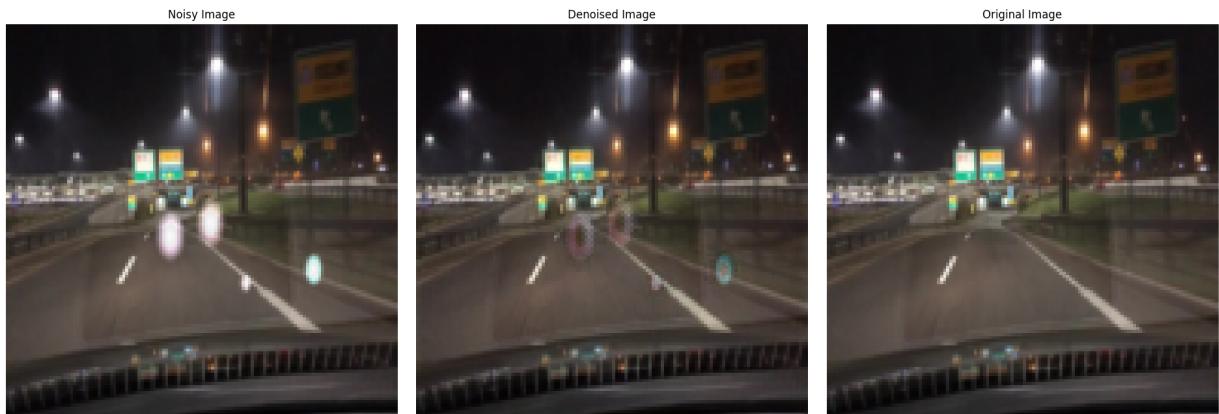


FIGURE 7 – De gauche à droite : image bruitée automatiquement, image en sortie du modèle U-Net utilisé pour le débruitage des halos lumineux et image d'origine.

Comme nous pouvons le constater, le modèle débrute partiellement l'image et semble introduire de la transparence dans les zones bruitées. Nous avons alors essayé de débruiter les images du dataset *rainy* à l'aide de ce modèle. La Figure 8 montre deux des résultats obtenus sur ce dataset.

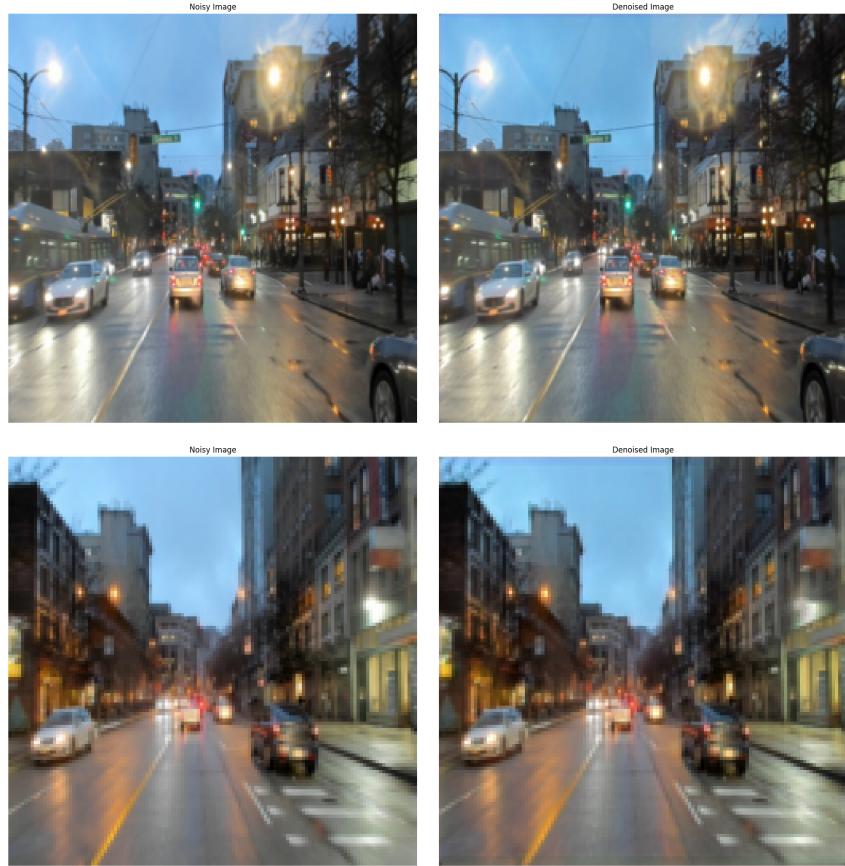


FIGURE 8 – De gauche à droite : image d’origine, image débruitée à l’aide du modèle U-Net.

Nous pouvons voir que le débruitage est présent mais très largement insuffisant ce qui implique que nous n’avons pas continuer d’utiliser le modèle produit au niveau des expérimentations. Il aurait notamment été intéressant de débruiter "manuellement" le dataset proposé par Wu et al., e.g., à l'aide d'un outil de traitement d'image assisté par ordinateur pour ensuite entraîner le modèle sur celui-ci. Ainsi, l'ensemble des bruits auraient été traités ce qui aurait impliqué que le modèle traite l'ensemble de ces bruits et non uniquement les bruits générés par des sources de lumières ponctuelles. Aussi, avec plus de moyens techniques, il aurait été intéressant de construire un nouveau dataset en prenant en photo des lieux derrière un pare-brise et sans pare-brise. Cela implique cependant de nombreux problèmes techniques à résoudre et donc de nombreuses questions auxquelles répondre comme la faisabilité de la mise en place et de la création d'un protocole permettant cela.

Quatrième partie

Expérimentations

5 Datasets

En explorant les images fournies pour ce projet, nous nous sommes rendus compte qu'il s'agissait des images de plusieurs vidéos. De plus, les datasets étant très volumineux, i.e. environ 4000 images pour les datasets *sunny* et *rainy* réunis, il a fallu se poser certaines questions cruciales comme : Faut-il se restreindre à un plus petit nombre d'images pour l'entraînement et l'évaluation de notre modèle ? Comment faire pour éviter l'*overfitting* du modèle sur une scène en particulier ?

Pour ce qui est du nombre d'images à utiliser, nous nous sommes restreints à un nombre variable mais limité d'images, en fonction du matériel utilisé, avec la limite d'un entraînement de deux heures pour une cinquantaine d'epochs. Ainsi, l'utilisateur peut choisir le ratio du dataset à prendre pour l'entraînement et l'évaluation. Le programme choisi alors de manière régulière les images à prendre à l'aide d'un pas en fonction du ratio choisi. Nous avons aussi décidé pour les datasets d'entraînement et de test de créer une limite temporelle entre les deux datasets. E.g., si le dataset d'entraînement prend les images de l'indice 0 à l'indice i , alors le dataset de test prendra les images de l'indice $i+1$ à N .

6 Matériel

| Matériel | Paramètres |
|---------------------------|--------------------------------------|
| CPU | 12th Gen Intel(R) Core(TM) i7-12700F |
| RAM | 32 G |
| GPU | NVIDIA GeForce GTX 1660 SUPER |
| Language de programmation | Python 3.12.0 |
| Deep Learning Framework | PyTorch 2.4.1 |
| CUDA | 11.8 |

7 Evaluation

Le Tableau 1 présente la définition de la matrice de confusion.

| | Predicted Positive | Predicted Negative |
|-----------------|--------------------|--------------------|
| Actual Positive | TP | FN |
| Actual Negative | FP | TN |

TABLE 1 – Matrice de confusion

La matrice de confusion est notamment utilisée pour calculer l'*accuracy* du modèle qui sert de métrique d'évaluation. L'accuracy est calculée comme suit :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Nous avons également utilisé la Mean Intersection over Union (MIoU) comme métrique pour évaluer nos résultats. La formule de la MIoU est donnée par :

$$\text{MIoU} = \frac{1}{N} \sum_{i=1}^N \frac{|A_i \cap B_i|}{|A_i \cup B_i|}$$

Avec A_i la région de la prédiction pour la classe i , B_i la région de la vérité de terrain pour la classe i , $|A_i \cap B_i|$ l'aire de l'intersection de la prédiction et de la vérité de terrain pour la classe i , $|A_i \cup B_i|$ l'aire de l'union de la prédiction et de la vérité de terrain pour la classe i et N est le nombre total de classes.

Contrairement à l'accuracy qui est "pixel-wise", i.e., qui évalue la performance pixel par pixel, la MIoU permet de traiter le problème de "class imbalance". En effet, la MIoU calcule l'accuracy pour chaque classe individuellement, ce qui permet de mieux prendre en compte les classes avec un nombre de pixels plus faible. Cela évite qu'une classe, comme le *background*, qui contient beaucoup de pixels, ne domine les résultats au détriment des autres classes qui sont moins représentées.

La Cross-entropy et la Dice-Sørensen sont les deux fonctions de perte qui ont été utilisées pour l'entraînement des différents réseaux de neurones.

La Cross-entropy est calculée comme suit :

$$H(p, q) = - \sum_i p(x_i) \log(q(x_i))$$

où $p(x_i)$ est la distribution réelle des classes et $q(x_i)$ est la distribution prédite des classes. La Cross-entropy est une fonction de perte largement répandue et utilisée pour l'entraînement des réseaux de neurones.

La Dice-Sørensen Loss est calculée comme suit :

$$\text{Dice-Sørensen Loss} = 1 - \frac{2 \sum_i p(x_i) \cdot q(x_i)}{\sum_i p(x_i) + \sum_i q(x_i)}$$

La Dice-Sørensen Loss, étant très similaire à la formule de la MIoU, permet lors de la rétropropagation de traiter toutes les classes de manière équitable, ce qui est particulièrement utile pour gérer le problème de déséquilibre des classes. En effet, contrairement à d'autres fonctions de perte qui peuvent favoriser les classes les plus représentées, la Dice Loss met l'accent sur la similarité entre les régions d'intérêt, indépendamment de leur taille relative dans l'image. C'est pourquoi cette fonction de perte est couramment utilisée dans des tâches de segmentation d'images notamment médicales, où l'on doit souvent détecter des structures de petite taille ou rares, telles que des tumeurs, des lésions, ou des organes spécifiques. Dans ces contextes, la Dice-Sørensen Loss permet au modèle d'optimiser la précision de la segmentation, même lorsqu'une classe est fortement sous-représentée.

8 Résultats

Nous avons entraîné deux modèles sur une partie du dataset fourni pour le projet. Les images sont choisies aléatoirement dans celui-ci. Nous avons utilisé comme hyperparamètres un **batch size**

de 16 et un **learning rate** de 0.001. Trois datasets ont été utilisés pour l'entraînement et l'évaluation des modèles : *sunny*, *rainy* et *denoised_rainy* renommé *denoised* dans la présentation des résultats. Le dataset *denoised* a été obtenu en appliquant le modèle IBCLN de débruitage présenté dans la partie III sur le dataset *rainy*.

Pour les deux modèles, U-Net et PSPNet, nous avons entraîné le modèle sur 50 **epochs** et testé celui-ci avec plusieurs fonctions de perte, dont la *Cross entropy* et la *Dice Loss* décrites dans la partie 7.

L'accuracy finale, le score MIoU et la loss (*Cross entropy* ou *Dice Loss*) du modèle U-Net et PSPNet obtenus sont présentés dans la Table 2. De plus, des exemples de prédiction sont présentés dans les Tables 3 et 4.

| Model | Loss Function | Dataset | Accuracy | Loss value | MIoU |
|--------------|----------------------|----------------|-----------------|-------------------|--------------|
| U-Net | Cross Entropy | Sunny | 90.54% | 0.304 | 0.416 |
| U-Net | Cross Entropy | Rainy | 88.40% | 0.463 | 0.383 |
| U-Net | Cross Entropy | Denoised | 93.14% | 0.208 | 0.458 |
| U-Net | Dice Loss | Sunny | 86.11% | 0.239 | 0.244 |
| U-Net | Dice Loss | Rainy | 84.32% | 0.274 | 0.206 |
| U-Net | Dice Loss | Denoised | 87.89% | 0.246 | 0.267 |
| PSPNet | Cross Entropy | Sunny | 89.35% | 0.340 | 0.390 |
| PSPNet | Cross Entropy | Rainy | 88.06% | 0.431 | 0.380 |
| PSPNet | Cross Entropy | Denoised | 90.97% | 0.254 | 0.411 |
| PSPNet | Dice Loss | Sunny | 87.11% | 0.225 | 0.295 |
| PSPNet | Dice Loss | Rainy | 85.75% | 0.245 | 0.269 |
| PSPNet | Dice Loss | Denoised | 85.51% | 0.227 | 0.288 |

TABLE 2 – Accuracy, loss et score MIoU des modèles U-Net et PSPNet, selon le dataset et la fonction de perte utilisée.

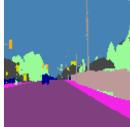
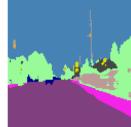
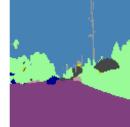
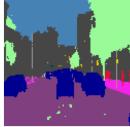
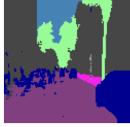
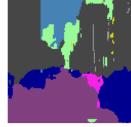
| Dataset Type | Original Image | Target Image | Cross Entropy Prediction | Dice Prediction |
|---------------------|---|---|--|---|
| Sunny |  |  |  |  |
| Rainy |  |  |  |  |
| Denoised |  |  |  |  |

TABLE 3 – Exemple de prédictions du modèle U-Net, selon le dataset et la fonction de perte utilisée.

| Dataset Type | Original Image | Target Image | Cross Entropy Prediction | Dice Prediction |
|--------------|----------------|--------------|--------------------------|-----------------|
| Sunny | | | | |
| Rainy | | | | |
| Denoised | | | | |

TABLE 4 – Exemple de prédictions du modèle PSPNet, selon le dataset et la fonction de perte utilisée.

9 Analyse et Conclusion

Les résultats obtenus montrent que le modèle U-Net est plus performant que le modèle PSPNet. En effet, l'accuracy du modèle U-Net est généralement plus élevée que celle du modèle PSPNet, bien que ce n'est pas toujours le cas. De plus, la MIoU du modèle U-Net est généralement plus élevée que celle du modèle PSPNet. On remarque également que la *Cross entropy* donne une meilleure accuracy que la *Dice Loss* mais aussi une meilleure MIoU¹.

Les résultats obtenus montrent également que les images du dataset *denoised*, soit les images *rainy* débruitées, donnent de meilleurs résultats que les images *rainy* ce qui justifie l'utilisation d'un modèle de débruitage avant la segmentation. On observe notamment une nette amélioration de l'ordre de +5% en ce qui concerne le modèle U-Net entraîné avec la *Cross entropy*.

Si l'on observe les images de prédiction données dans les Tables 3 et 4, on remarque que les modèles ne predisent pas parfaitement les images. Par exemple, les panneaux de signalisation, les feux de circulation ou les poteaux ne sont pas toujours bien segmentés. Cela peut partiellement s'expliquer par le fait que les images sont rétrécies pour l'entraînement et l'évaluation des modèles, mais aussi car les fonctions de perte utilisées ne sont pas parfaites pour ce type de tâche. En effet, la *Cross entropy* par exemple aurait pu être améliorée en utilisant des poids, pour que les classes moins représentées soient mieux prises en compte. Sur les exemples donnés on remarque également que la *Dice Loss* donne des résultats moins satisfaisants que la *Cross entropy*, en particulier sur les images *denoised*.

En conclusion, les résultats obtenus montrent que le modèle U-Net est plus performant que le modèle PSPNet pour la segmentation d'images. De même les images *denoised* donnent de meilleurs résultats que les images *rainy* simples. Cependant, les résultats obtenus ne sont pas parfaits et

1. Il est important de noter que les résultats ont été calculé à partir d'une moyenne des données sur plusieurs modèles. Par défaut, le nombre de modèles testés est au nombre de cinq. Le paramètre à modifier est `it` du script `./script/train.py`.

pourraient être améliorés en utilisant des fonctions de pertes plus adaptées au problème donné. Aussi et comme mentionné, il aurait été intéressant de construire un dataset propre au problème en débruitant manuellement les images du dataset *rainy*. Nous nous sommes aussi restreints à une partie du dataset, celui-ci étant trop grand par rapport au matériel à disposition pour pouvoir effectuer des entraînements raisonnables au niveau du temps et de la consommation énergétique. Les résultats des différents modèles seraient alors naturellement meilleurs en prenant en compte l'ensemble des datasets.

Références

- [1] Chao Li, Yixiao Yang, Kun He, Stephen Lin, and John E Hopcroft. Single image reflection removal through cascaded refinement. *arXiv preprint arXiv :1911.06634*, 2019.
- [2] Chao Li, Yixiao Yang, Kun He, Stephen Lin, and John E Hopcroft. Single image reflection removal through cascaded refinement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3565–3574, 2020.
- [3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention-MICCAI 2015 : 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [4] Nahian Siddique, Sidiqe Paheding, Colin P Elkin, and Vijay Devabhaktuni. U-net and its variants for medical image segmentation : A review of theory and applications. *IEEE access*, 9 :82031–82057, 2021.
- [5] Wikipedia. U-net. <https://en.wikipedia.org/wiki/U-Net>. Accessed : 2025-01-05.
- [6] Aming Wu and Cheng Deng. Single-domain generalized object detection in urban scene via cyclic-disentangled self-distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 847–856, 2022.
- [7] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.
- [8] Jie Zhao, Lei Dai, Mo Zhang, Fei Yu, Meng Li, Hongfeng Li, Wenjia Wang, and Li Zhang. Pgynet+ : progressive growing of u-net+ for automated cervical nuclei segmentation. In *Multiscale Multimodal Medical Imaging : First International Workshop, MMMI 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 13, 2019, Proceedings 1*, pages 51–58. Springer, 2020.