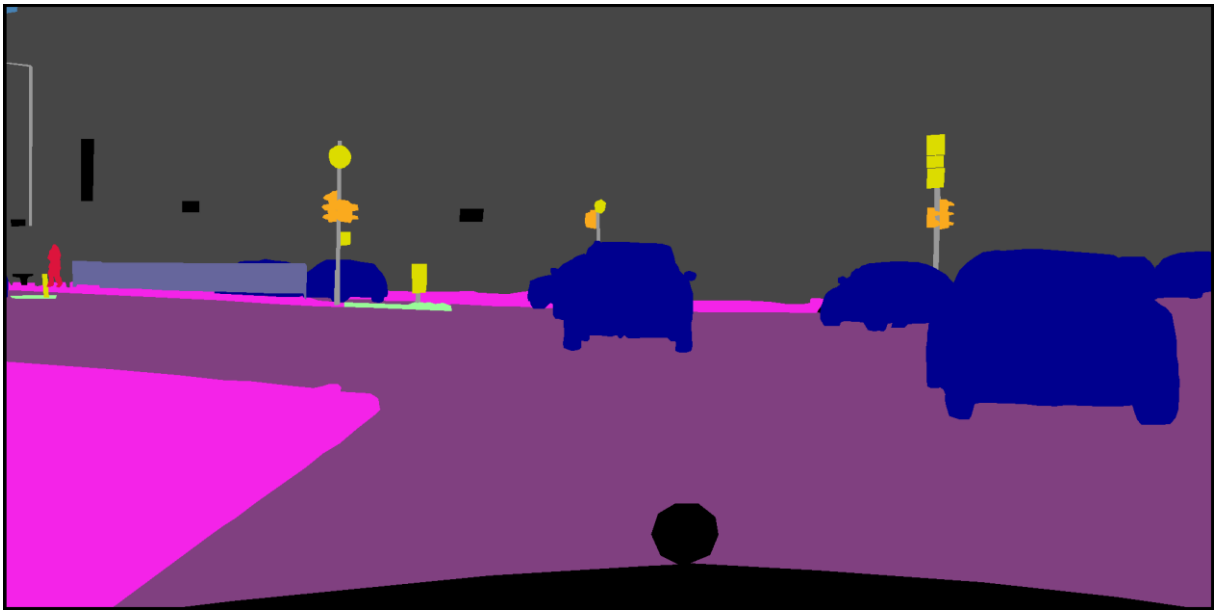
Note Technique

La segmentation d'images

Détection des éléments d'un paysage urbain par une voiture autonome



Auteur : Anthony RENARD – juin 2023

1. Introduction

Depuis des années, l'Homme rêve d'une voiture qui pourrait l'emmener où il veut sans conduire. Il pourrait pendant ce temps, travailler, jouer, regarder un film ou tout simplement dormir.

Aujourd'hui des prototypes de cette voiture existent grâce aux progrès faits dans le domaine de l'Intelligence Artificielle (IA) et plus précisément sur la détection automatique d'éléments dans notre environnement. En effet, plusieurs problématiques apparaissent notamment la détection des véhicules et des piétons autour de la voiture mais aussi la lecture des panneaux pour par exemple limiter la vitesse du véhicule ou stopper le véhicule de manière automatique sans l'intervention de l'Homme.

Nous verrons dans cette note technique comment la détection d'objets à partir d'images de paysages urbains est possible et comment réaliser cette détection grâce à l'intelligence artificielle.

2. Les paysages urbains

2.1 L'environnement

Lorsque l'Homme conduit une voiture, il va croiser beaucoup d'éléments que ça soit du matériel, des êtres vivants mais aussi des panneaux de signalisation pour lui dicter les paramètres à respecter pour sa conduite (panneaux STOP, feux tricolores, rond-point, …). Il prendra donc des décisions et transmettra ses décisions au véhicule via le volant, les pédales et autres commandes comme les clignotants.

Figure 1: véhicule à l'arrêt devant des piétons traversant la route

Dans ce cas, la « vue » est le sens nécessaire pour que l'Homme puisse analyser les différentes scènes avec son cerveau et ainsi manipuler la voiture et lui faire faire les actions adéquates en fonction des situations qu'elle va rencontrer lors de son trajet.

Pour une voiture autonome, c'est une **caméra** qui servira pour la « **vue** » et **l'intelligence artificielle** qui fera office de **cerveau** afin de détecter les éléments de la scène. Le scope de cette note technique sera consacré à **la détection des éléments environnant le véhicule à un instant t**. Le modèle d'intelligence artificielle prendra donc en entrée une photo prise par le véhicule et ressortira une image labellisée de cette photo (1 label par pixel).

Remarque : Les conséquences de ces détections sur le mouvement du véhicule ne seront pas traitées dans ce document.

2.2 Les photos labellisées

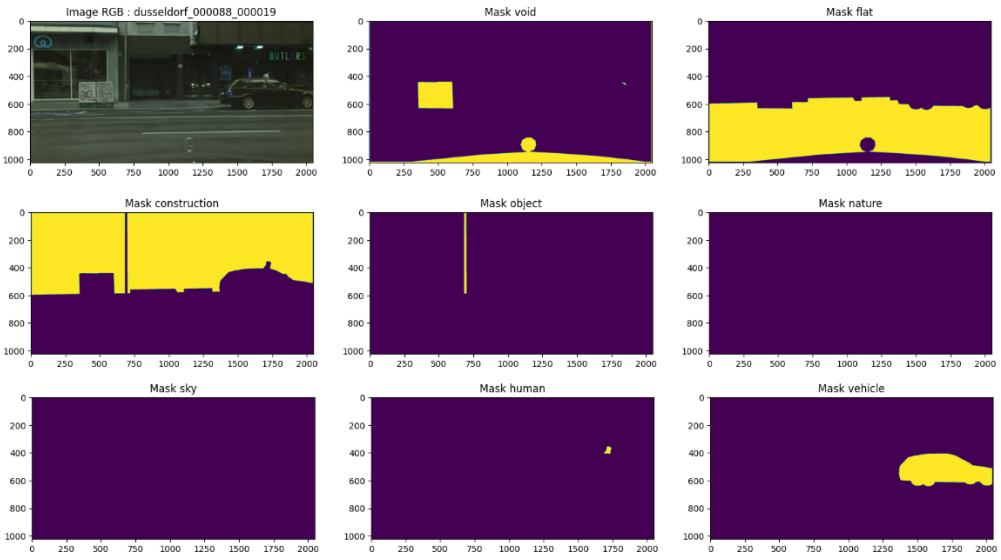
Afin de pouvoir entrainer notre modèle d'intelligence artificielle de manière supervisée, on va devoir se procurer des photos ainsi que les images annotées correspondantes. En effet, nous avons la chance d'avoir à notre disposition des images associées à leur masque (1 couleur par label de pixel) qui serviront à l'entrainement et à la validation de notre modèle. Ces entrées sont disponibles sous : Dataset Overview – Cityscapes Dataset (cityscapes-dataset.com).

Voici un exemple d'une photo et son masque de segmentation associée :

Le masque de droite correspond aux 30 classes proposées par Cityscapes Dataset. Cependant, pour notre étude, nous prendrons seulement les 8 catégories suivantes dans lesquelles les 30 classes y seront distribuées. Ces 8 catégories sont :

1. Void
2. Flat
3. Construction
4. Object
5. Nature
6. Sky
7. Human
8. Vehicule

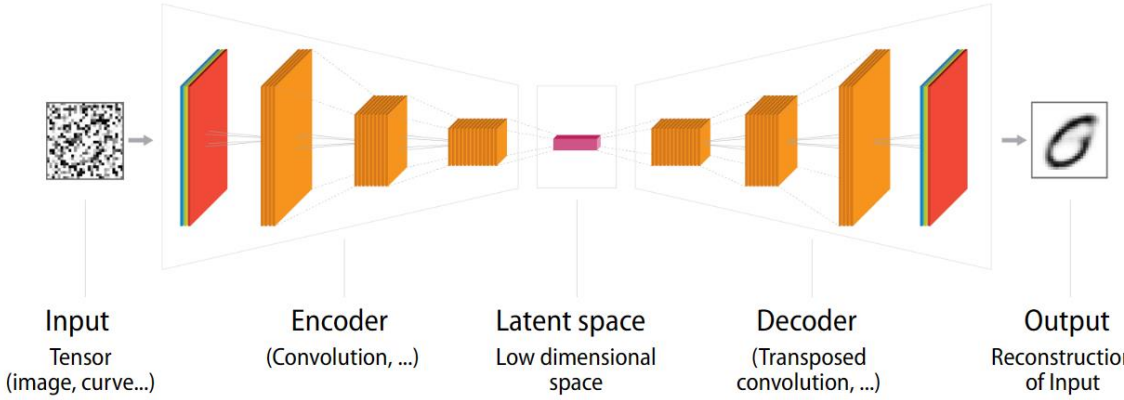
Chaque image sera donc décomposée en 8 masques qui serviront pour l'entrainement de notre modèle de segmentations :



3. Les modèles

3.1 Structure générale

Afin de mettre en valeur les « features » de l'image, nous utiliserons sur une structure « auto-encodeur » qui est composée d'un **encodeur** et d'un **décodeur** reliés par un **espace latent**.

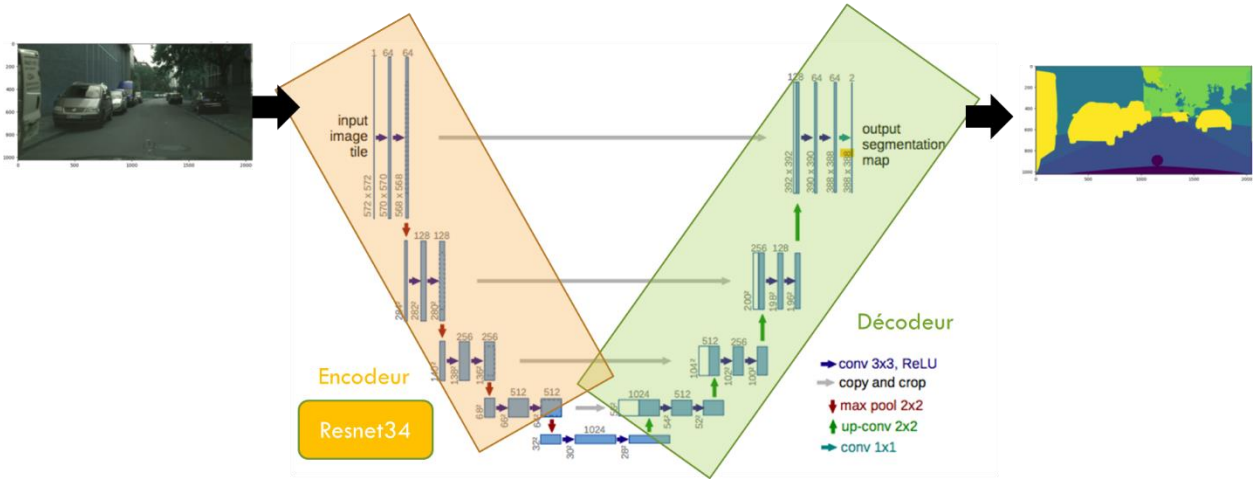


- **L'encodeur** va permettre, grâce à des couches convolutionnelles et de poolings de réduire les dimensions de l'image d'entrée en ne gardant que les informations essentielles de l'image (les features). On va donc récupérer un vecteur qui sera une projection des données dans cet **espace latent** qui est de petite taille.

- **Le décodeur** va en revanche réaugmenter les dimensions du vecteur de **l'espace latent** (qui contient les caractéristiques essentielles des données de départ) afin de reconstituer l'image d'entrée « non bruitée » en s'appuyant sur les caractéristiques du vecteur de l'espace latent. Il est composé de couches de convolutions transposées.

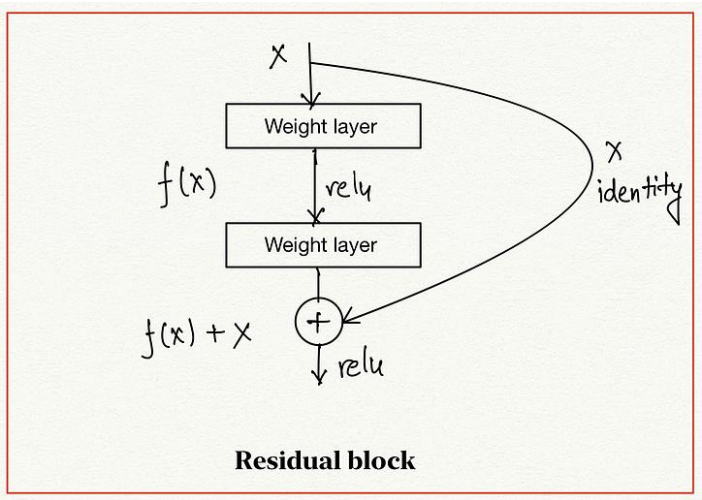
3.2 Modèle de référence : UNET-RESNET34

UNET est une structure en forme de « U »



Son avantage est de pouvoir détecter des objets de petites tailles.

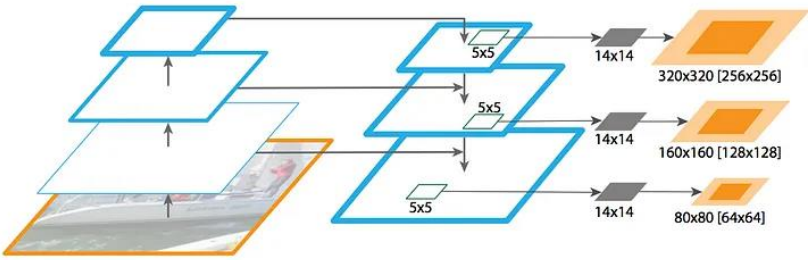
Le décodeur choisi en référence sera Resnet34 qui contient des sorties connectées à l'entrée pour éviter le phénomène de disparition du gradient :



Ce modèle servira de point de comparaison avec les 3 modèles complémentaires ci-après.

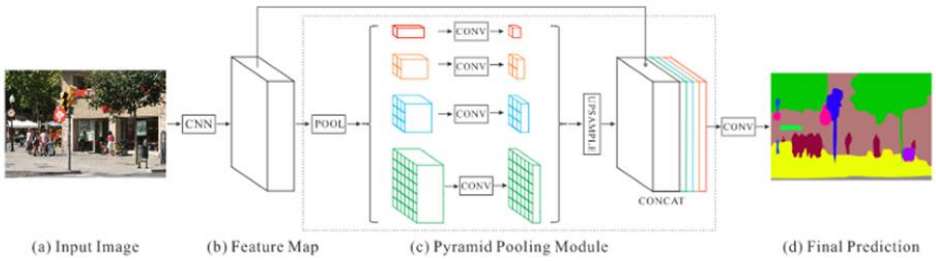
3.3 Modèles complémentaires

3.3.1 FPN : Features Pyramid Network



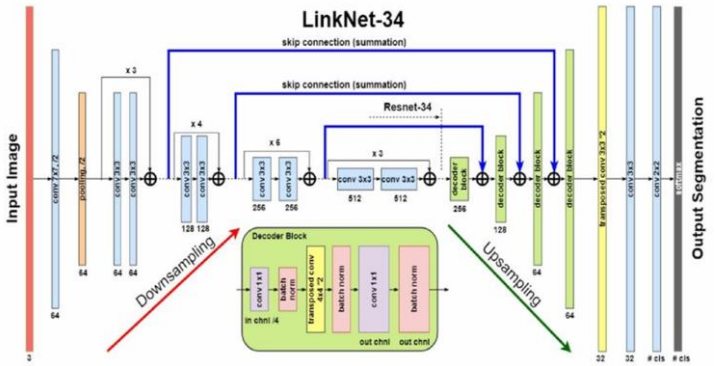
Cette structure pyramidale a pour avantage de détecter les mêmes éléments à des échelles différentes ; ce qui semble très utile pour notre problématique de détection des éléments dans l'espace.

3.3.2 PSPnet : Pyramid Scene Parsing network



Ce réseau, en plus de détecter les éléments à différentes échelles, va aussi prendre en compte le contexte de l'image.

3.3.3 LinkNet



Dérivé de l'architecture Resnet, ce réseau minimise le nombre de paramètres ; par cet aspect, il est idéal pour de la segmentation en temps réel.

4. Métriques et fonction de pertes

4.1 Les métriques

Les métriques vont nous permettre d'évaluer la performance de nos modèles. En plus du temps d'entrainement, nous allons sélectionner 3 métriques et trouver celle la plus adaptée à notre problématique c'est-à-dire celle qui permettra de départager les 4 modèles vus précédemment.

Ces 3 métriques sont :

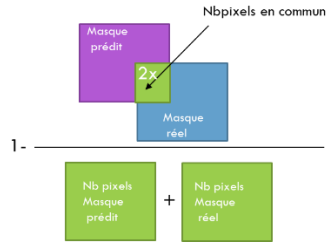
- **Accuracy** : calcul la fréquence de bonnes prédictions parmi tous les labels de l'image
- **keras.losses.loss** : contient la logique de calcul des pertes en utilisant y_true et y_pred
- **Binary_IOU** : calcul de « Intersection Over union et plus précisément : $\mathrm{BIOU} = \frac{\mathrm{TP}}{\mathrm{TP+FP+FN}}$

Remarque : une fonction de perte peut être utilisée en métrique ; c'est le cas de « keras.losses.loss ».

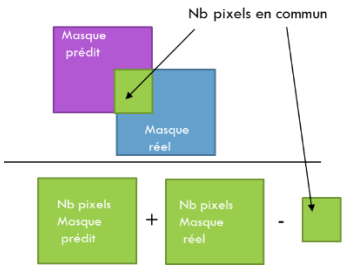
4.2 Les fonctions de perte

Les fonctions de perte sont les paramètres utilisés pour optimiser le modèle par l'optimiseur. Nous allons en selectionner 3 et trouver celle la plus adaptée à notre problématique c'est-à-dire celle qui permettra de départager les 4 modèles vus précédemment. Ces 3 fonctions de pertes sont :

- **Dice Loss** $= 1 - \frac{\text{2 x le nombre de pixels en commun}}{\text{Le nombre total de pixel sur les 2 images}}$



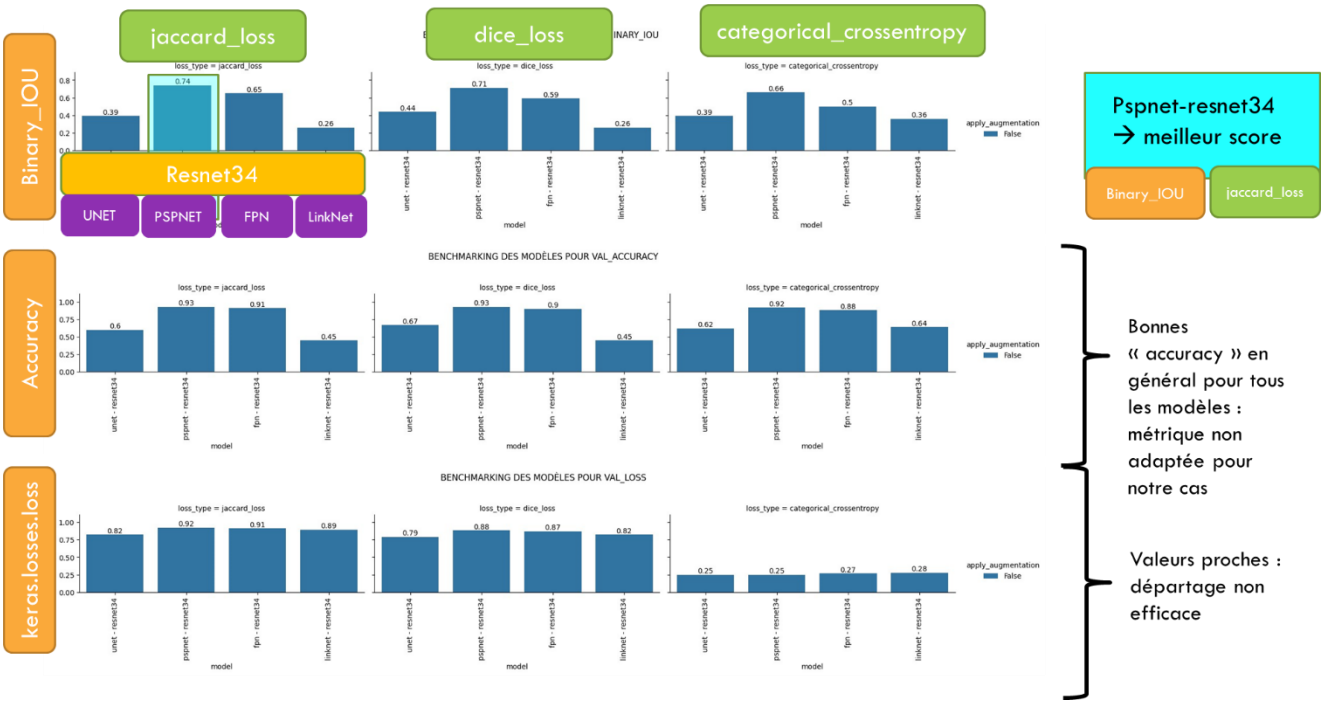
- **Jaccard Loss** $= \frac{\text{le nombre de pixels en commun}}{\text{Le nombre total de pixel sur les 2 images} - \text{le nombre de pixels en commun}}$



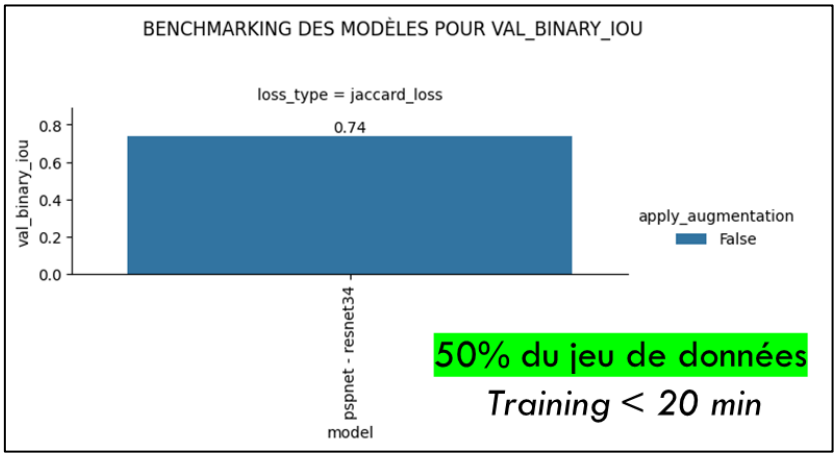
- **Categorical-crossentropie** : calcul de la perte « cross entropie » entre les labels et la prédiction

4.3 Sélection de la métrique et de la fonction de perte

En entrainant les 4 modèles sur 50% du jeu de données ont peut déduire que la métrique **Binary_IOU** et la fonction de perte **Jaccard_Loss** sont les 2 indicateurs les plus pertinents pour notre problématique car ils permettent de départager nos 4 modèles.



Aussi le modèle le plus performant sur ces 2 indicateurs est le modèle PSPnet-resnet34 :

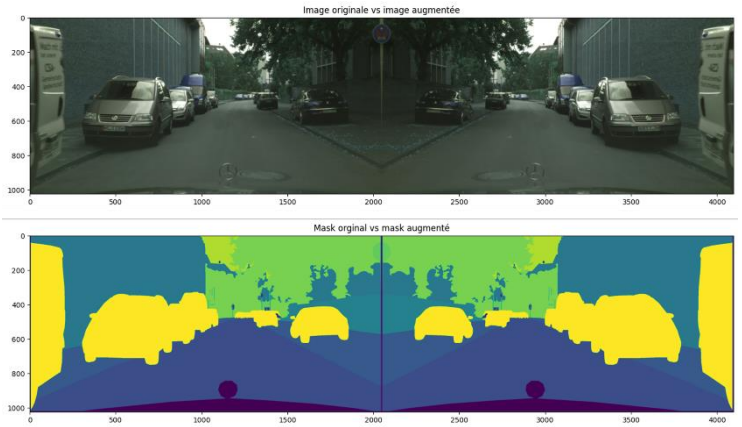


5. Amélioration des performances

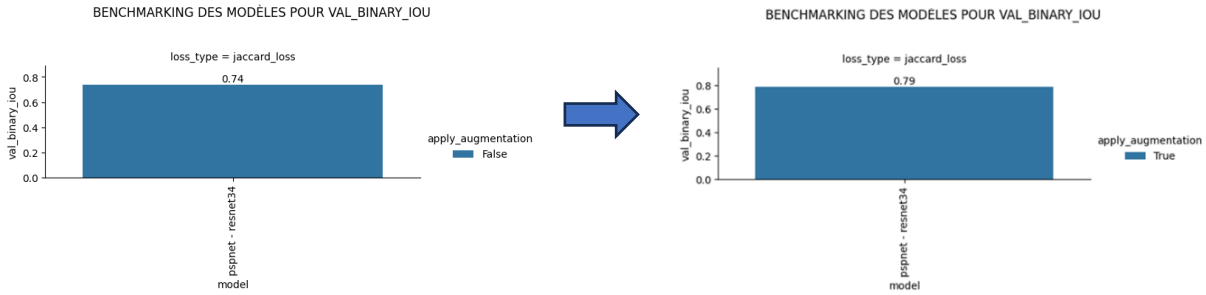
5.1 Augmentation

L'augmentation va permettre de créer de nouvelles images à partir des 2975 disponibles dans le jeu de données d'entrainement. Nous retrouverons donc les transformations suivantes :

- Rotation
- Retournement vertical
- Retournement horizontal ➔
- Recadrage des images
- Changement d'échelle
- Floutage Gaussien
- Floutage dynamique
- Changement de teinte
- Mise en noire et blanc
- Egalisation d'histogramme



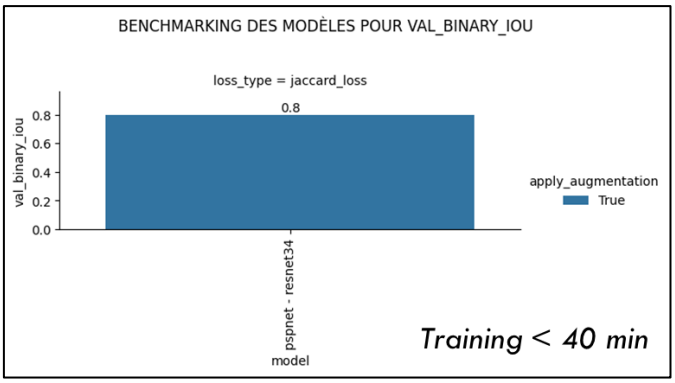
Le jeu de données sera donc plus conséquent et notre entrainement sera plus efficace. Avec 50% du jeu de données l'Augmentation apporte une performance complémentaire de 5% (74% →79%).



Conclusion : l'Augmentation permet d'améliorer les performances du modèle.

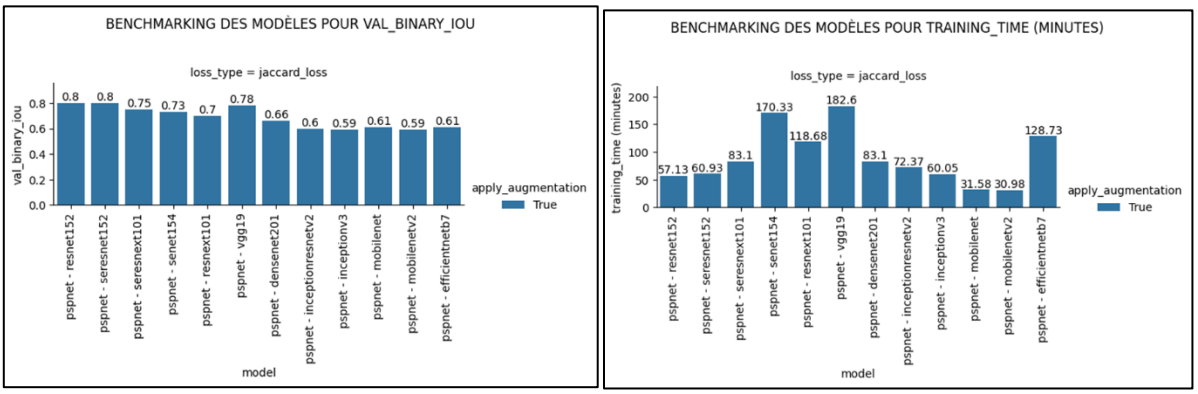
5.2 Utilisation de 100% du jeu de données

En utilisant 100% du jeu de donnée avec « augmentation », nous obtenons un résultat de 80% :



5.3 Autres encodeurs

Le modèle **PSPnet** encapsulé de l'encodeur **Resnet34** permet d'avoir Binary_IOU = 80% avec 100% du jeu de données. Voici le résultat obtenu avec d'autres encodeurs :



On remarque que d'un point de vue **performance/training_time** le modèle **PSPnet-Resnet34** est le plus performant, avec le jeu de données que l'on a disposition, avec un score de 80% pour 40min d'entrainement.

5.4 Autres Optimiseurs

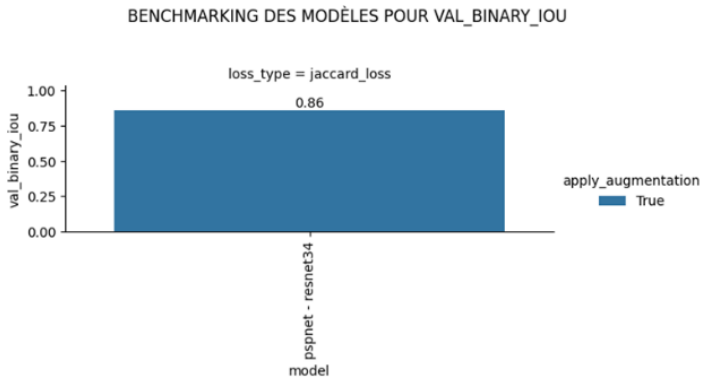
L'optimiseur utilisé pour les analyses précédente est ADAM. Sur la même configuration que précédemment, voici les résultats obtenus en changeant l'optimiseur :

Optimiseur	Binary_IOU	Training time
Adam	80%	<40 min
Sgd	82%	<40 min
RMS	83%	<45min
ftrl	44%	<45min
adadelta	70%	<45min

Conclusion : le modèle est amélioré de +3% grâce à l'optimiseur RMS (80% → 83%)

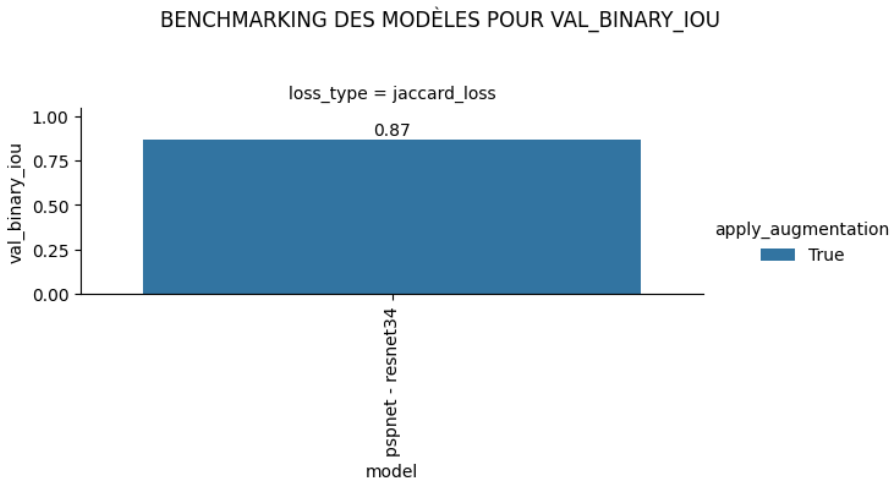
5.5 EPOCH

En augmentant le nombre d'EPOCH à 50, le modèle s'améliore encore de +3% **(83% → 86%).**



5.6 Réentrainement du meilleur modèle

On réutilise le modèle précédent pour le réentraîner. Cette méthode améliore le modèle de 1% **(86% → 87%).**

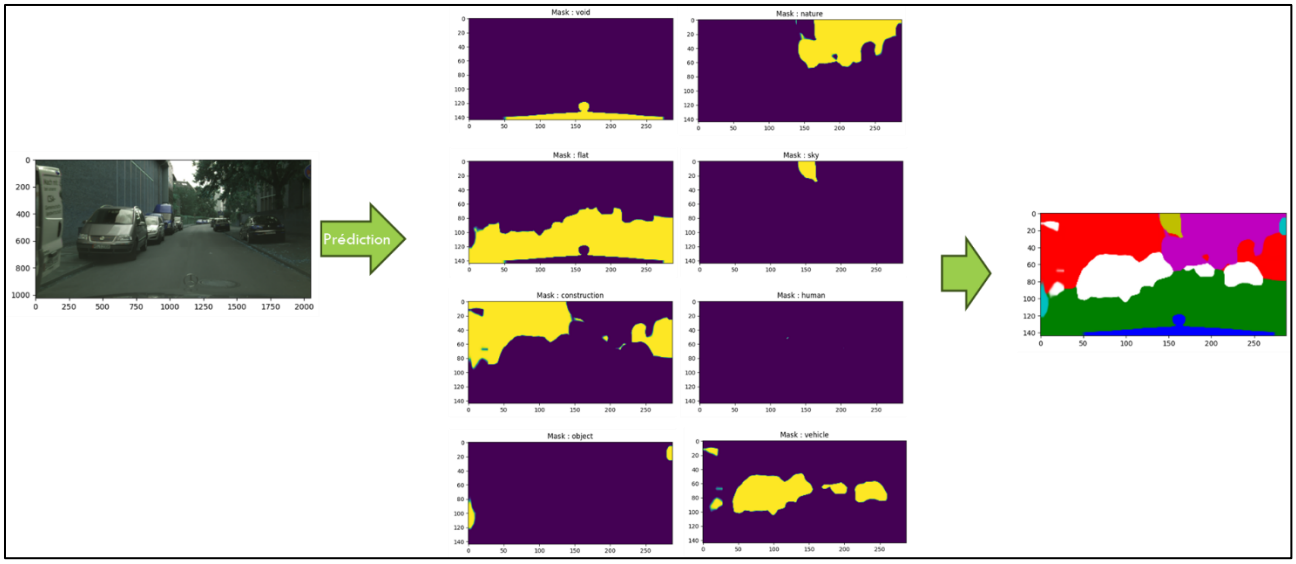


5.7 Conclusion

Grâce aux méthodes d'amélioration des performances, le modèle choisi a vu son score augmenter de **13% au total.** Le modèle final a une performance de **87%.**

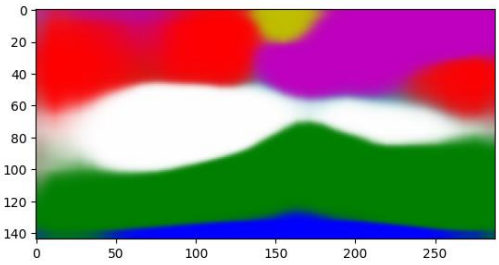
6. Prédictions

L'objectif de ce modèle de segmentation est de prédire un masque contenant les 8 catégories citées précédemment. Voici le résultat obtenu avec un exemple concret :

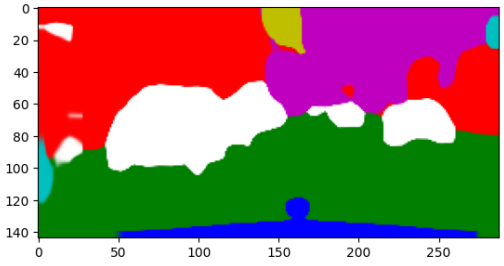


A gauche la photo prise par la voiture et à droite les 8 masques par catégories qui forment ensuite le masque final.

Plus le score du modèle sera élevé et plus la délimitation de chaque objet dans l'image sera nette :



Binary_iou = 80%



Binary_iou = 87%

7. Conclusion

Le modèle de segmentation créé permet de détecter les éléments entourant le véhicule. L'entrainement du modèle est basé sur 3000 images environ. Grace aux techniques d'augmentation, d'autres images ont pu être recrées et ainsi entrainer le modèle avec plus d'échantillons ce qui a augmenté sa performance. La recherche d'autres paramètres (EPOCH, encodeurs, ..) a également contribuée à l'augmentation de la performance. Au total le gain en performance est de +13% ce qui permet d'obtenir un modèle de segmentation ayant un score **binary_iou=87%.**

En termes d'opportunités, on pourrait :

→Utiliser des images de plus hautes qualités pour améliorer le modèle et ainsi augmenter la quantité de notre jeu de données.

→Multiplier les augmentations par photo

→Utiliser de nouvelles photos segmentées