Entraîner un modèle de segmentation

1 Résumé

Nous avons dans ce projet réalisé un modèle capable de segmenter une image en 8 catégories, le meilleur score que nous avons obtenu (avec la métrique IoU) est de 40%.

Ce score a été obtenu avec le modèle Unet et avec la data augmentation.

Vous verrez dans cette note technique les différentes étapes par lesquelles nous sommes passés.

Ci-dessous, le plan de la note technique.

Sommaire

2 Contexte

Future Vision Transport est une entreprise qui conçoit des systèmes embarqués de vision par ordinateur pour les véhicules autonomes.

Notre rôle est de concevoir un premier modèle de segmentation d'images qui devra s'intégrer facilement dans la chaine complète du système embarqué.

La segmentation d'images est une technique de computer vision qui consiste à découper de façon automatique une image en zones de pixels appartenant à une même classe d'objets. La segmentation d'images a de nombreuses applications, notamment en imagerie médicale.

Nous avons utilisé les données de cityscape pour mener à bien ce projet. Nous avons ensuite classé nos images ainsi que nos mask dans des dossiers différents, redimensionné nos images en 256x256.

Puis, nous avons augmenté artificiellement la quantité de nos images avec la data augmentation.

Et enfin, nous avons entrainé deux modèles : FCN et Unet, sur les images de base et sur les images augmentées, pour ensuite avoir un comparatif.

Nous vous détaillerons dans cette note technique, les différentes étapes par lesquelles nous sommes passé pour mener à bien ce projet.

3 Objectifs

Obtenir un modèle de segmentation capable de segmenter sémantiquement une image en 8 catégories:

1. Le ciel
2. Les véhicules
3. Les humains
4. La végétation
5. Divers (vide)
6. Zone plane
7. Les construction
8. Les objets

4 Jeu de données

Les données proviennent de CityScape.

L'ensemble de données Cityscapes se concentre sur la compréhension sémantique des scènes de rue urbaines.

Cet ensemble de données est mis gratuitement à la disposition d'entités universitaires et non universitaires à des fins non commerciales telles que la recherche universitaire, l'enseignement, les publications scientifiques ou l'expérimentation personnelle.

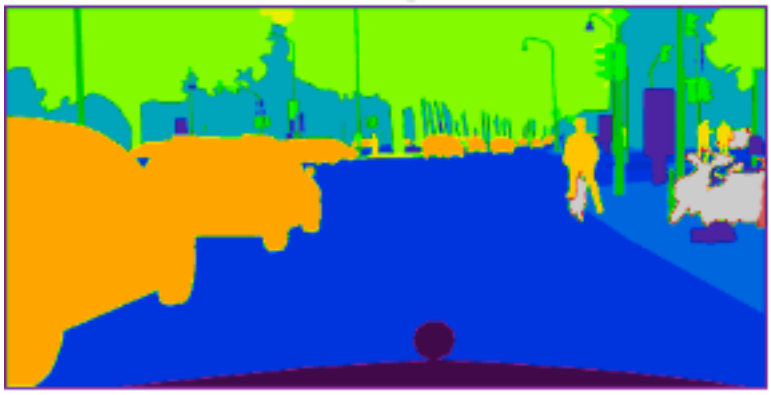
Nous avons donc utilisé ces données pour entrainer notre modèle.

Celle-ci est composée d'un dossier d'entrainement, de test (que nous n'avons pas utilisé pour l'entrainement des modèles), et de validation.

Chacun de ces dossiers contient la même structure :

1. L'image original :

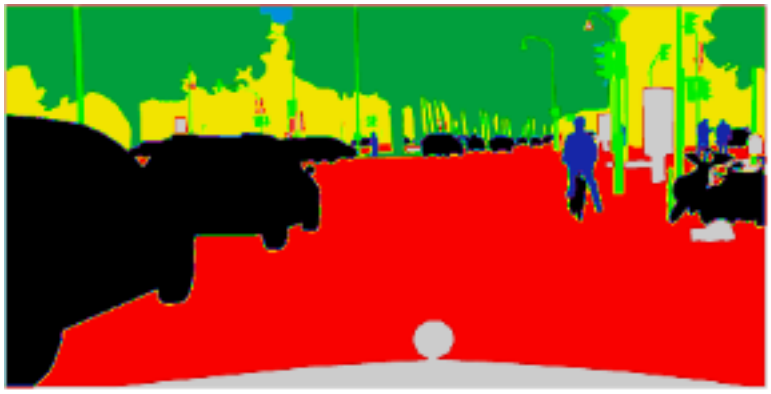
2. Le mask de l'image (target):



Après avoir organisé nos données, nous totalisons 2380 images/mask pour l'entrainement et 595 images/mask pour la validation. Nous avons également

créé un dossier supplémentaire contenant les mask avec 8 catégories plutôt que 30 sur les masks originaux comme demandés dans le projet:

1. Mask 8 catégories :



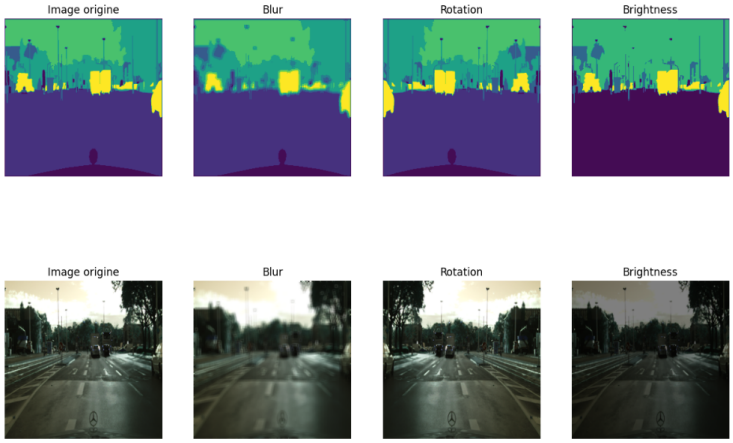
Les images et les mask ont tous étaient redimensionnés à 256x256.

5 Data Augmentation

Les algorithmes, même les plus complexes et puissants, ont besoin de données pour entrainer le modèle. L'augmentation des données est l'une des solutions les plus adaptées pour ce genre de problématique. Au lieu d'essayer de trouver et d'étiqueter plus de données, nous en construisons de nouveaux en fonction de ce que nous avons.

L'augmentation des données d'image est peut-être la technique la plus connue d'augmentation des données. Elle regroupe les techniques utilisées pour augmenter artificiellement la taille d'un groupe de données d'apprentissage en créant des versions modifiées d'images à partir des images d'apprentissage disponibles.

Nous avons donc augmenté les images en ajoutant du flou (blur), de rotation et d'assombrissement :



6 Générateur de données

Nous avons utilisé un générateur de données pour l'entrainement de nos modèles. Keras propose la fonction 'sequence' pour cela.

Nous avons donc créé une class 'sequence', composé des fonctions "__len__", "__getitem__" (qui sont obligatoires comme indiqué dans la documentation (https://www.tensorflow.org/api_doc et "on_epoch_end".

Voici le rôle de ces 3 fonctions :

6.1 "__len__":

Permets de créer la quantité de lot (batch) en fonction de la quantité d'images totale. Par exemple, si nous avons 1000 images, et que nous vous avons des lots de 10, cette fonction divisera les deux, nous aurons alors 100 lots de 10, où chaque lot aura un index.

6.2 "__getitem__":

Cette fonction permet de générer le batch correspondant à l'index appelé.

6.3 "on_epoch_end":

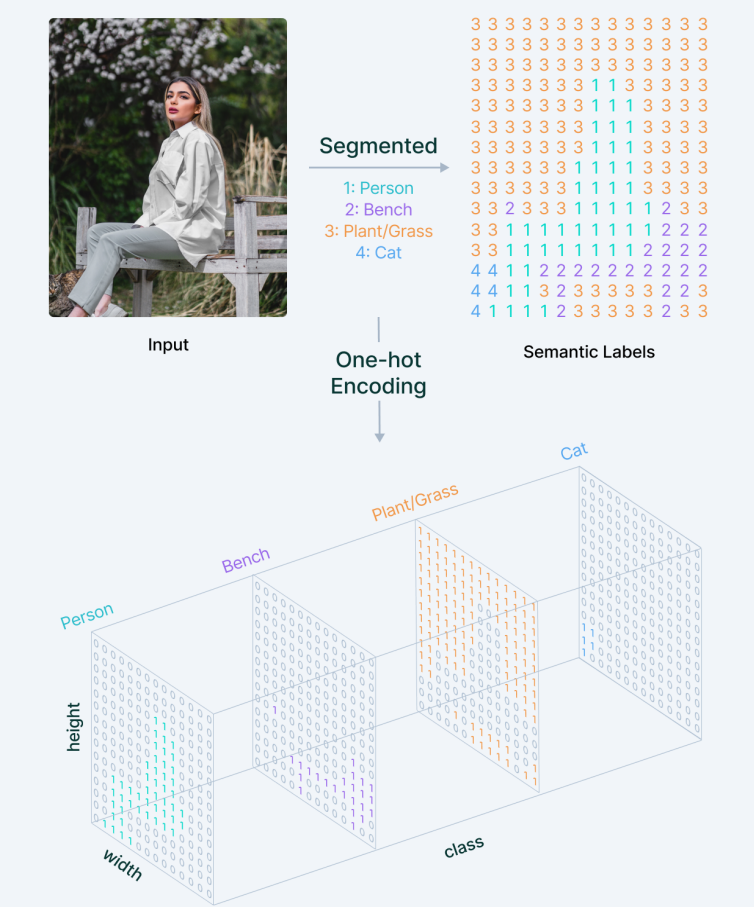
Cette fonction est exécutée à la première époque et à la fin de chaque époque, elle permet d'indiquer dans quel ordre les lots sont utilisés.

En ajoutant un shuffle, les lots de chaque époque ne se ressembleront pas.

7 Modèle

Le but des modèles est simplement de prendre une image et de générer une sortie telle qu'elle contienne une carte de segmentation où la valeur des pixels

(de 0 à 255) de l'image de sortie est transformée en une valeur d'étiquette de classe (0, 1, 2, ... n).



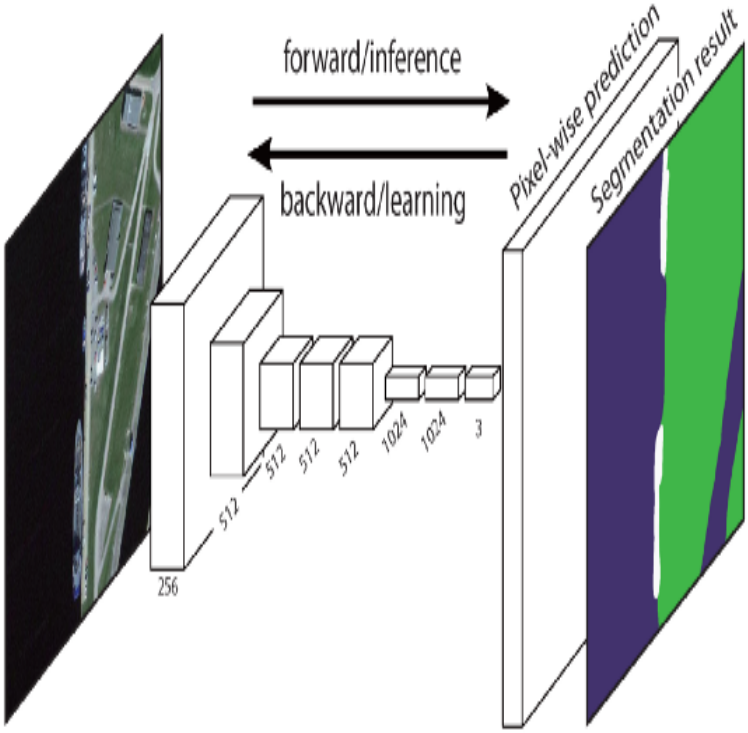
Il existe plusieurs modèles pour la segmentation d'images. Nous en avons utilisé deux, un simple qui est le modèle FCN, et un plus complexe qui est Unet.

Ces deux modèles sont basés sur VGG-16, ce modèle a remporté le concours ImageNet 2013 avec une précision de 92,7%. Il utilise une pile de couches de convolution avec de petits champs récepteurs dans les premières couches au lieu de quelques couches avec de grands champs récepteurs.

7.1 FCN

Les réseaux entièrement convolutifs, ou FCN, sont une architecture utilisée principalement pour la segmentation sémantique. Ils utilisent uniquement des couches connectées localement, telles que la convolution, la mise en commun et le suréchantillonnage. Éviter l'utilisation de couches denses signifie moins de paramètres (ce qui rend les réseaux plus rapides à former). Cela signifie

également qu'un FCN peut fonctionner pour des tailles d'image variables étant donné que toutes les connexions sont locales. Le réseau se compose d'un chemin de sous-échantillonnage, utilisé pour extraire et interpréter le contexte, et d'un chemin de suréchantillonnage, qui permet la localisation.



7.2 Unet

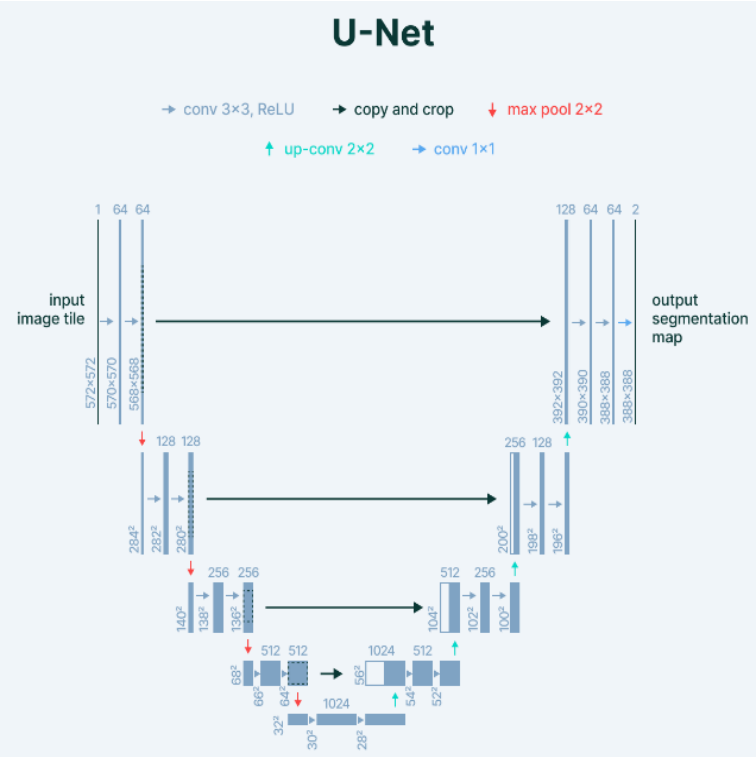
Le U-net est une modification d'un réseau entièrement convolutif.

Il a été introduit par Olaf Ronneberger et al. en 2015 à des fins médicales, principalement pour trouver des tumeurs dans les poumons et le cerveau.

Le U-net a une conception similaire d'un encodeur et d'un décodeur.

Le premier est utilisé pour extraire des caractéristiques par sous-échantillonnage, tandis que le second est utilisé pour suréchantillonner les caractéristiques extraites à l'aide des couches déconvolutionnelles.

La seule différence avec le modèle FCN, est que le FCN utilise les caractéristiques finales extraites pour suréchantillonner, alors que le U-net utilise ce que l'on appelle une connexion raccourcie.



La connexion par raccourci dans le U-Net est conçue pour résoudre le problème de la perte d'informations.

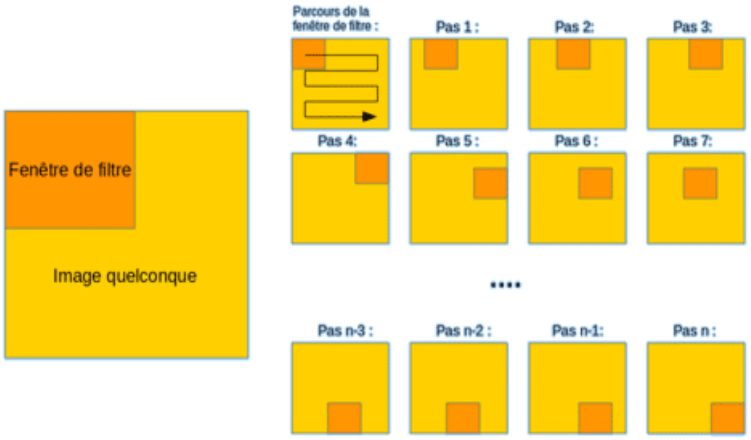
En effet, il regroupe simultanément les couches max-pools, ce qui signifie que des informations sont perdues au cours du processus. Cette architecture permet au réseau de capturer des informations plus fines et de conserver davantage d'informations en concaténant les fonctionnalités de haut niveau avec celles de bas niveau.

Ce processus de concaténation des informations de différents blocs permet à U-net de produire des résultats plus fins et plus précis.

7.3 Paramètre et détail

7.3.1 Convolution

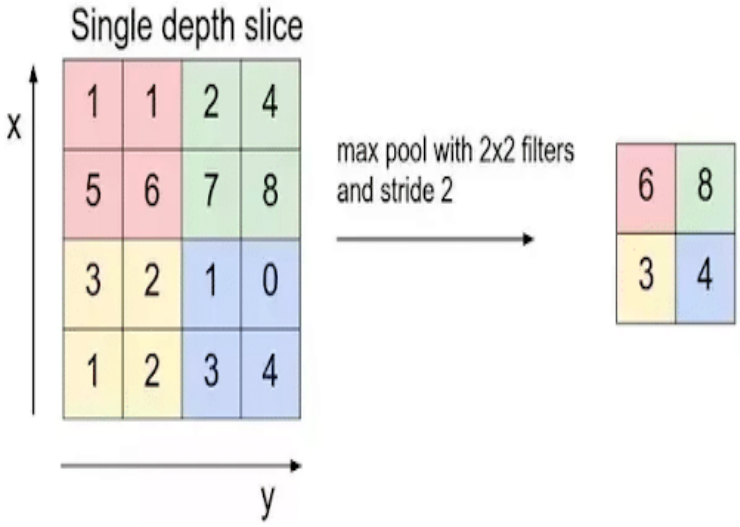
La convolution est une opération mathématique simple généralement utilisée pour le traitement et la reconnaissance d'images. Sur une image, son effet s'assimile à un filtrage dont voici le fonctionnement :



Dans un premier temps, on définit la taille de la fenêtre de filtre située en haut à gauche. La fenêtre de filtre, représentant la feature, se déplace progressivement de la gauche vers la droite d'un certain nombre de cases défini au préalable (le pas) jusqu'à arriver au bout de l'image. 'À chaque portion d'image rencontrée, un calcul de convolution s'effectue permettant d'obtenir en sortie une carte d'activation ou feature map qui indique où est localisées les features dans l'image : plus la feature map est élevée, plus la portion de l'image balayée ressemble à la feature.

7.3.2 Maxpooling

Le Max-Pooling est un processus de discrétisation basée sur des échantillons. Son objectif est de sous-échantillonner une représentation d'entrée (image, matrice de sortie de couche cachée, etc.) en réduisant sa dimension. Pour rendre plus concrète l'action du Max-Pooling, voici un exemple : imaginons que nous avons une matrice 4×4 représentant notre entrée initiale et un filtre d'une fenêtre de taille 2×2 que nous appliquerons sur notre entrée. Pour chacune des régions balayées par le filtre, le max-pooling prendra le maximum, créant ainsi par la même occasion une nouvelle matrice de sortie où chaque élément correspondra aux maximums de chaque région rencontrée.



7.3.3 Upsampling

Suréchantillonne l'image pour qu'elle soit de la même taille que l'image d'entrée.

7.3.4 Optimiseur

Les optimiseurs sont des algorithmes ou des m´méthodes utilisées pour minimiser une fonction d'erreur (fonction de perte) ou pour maximiser l'efficacité de la production. Ils aident 'a savoir comment modifier les poids et le taux d'apprentissage du réseau de neurones pour réduire les pertes. Nous avons utilisé l'optimiseur Adam sur nos deux modèles.

D'autre concept sont intéressant dans l'évaluation de nos modèles :

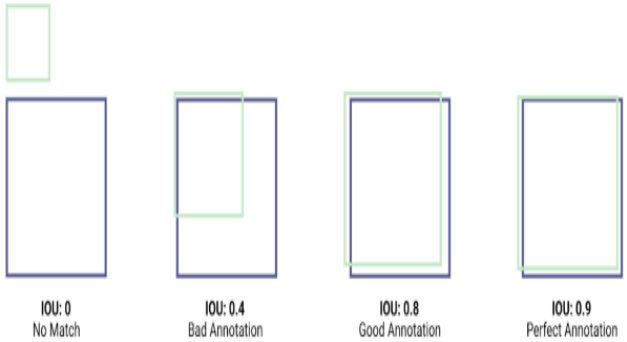
7.3.5 Fonction de perte (loss)

Une fonction de perte, ou Loss function, est une fonction qui évalue l'écart entre les prédictions réalisées par le réseau de neurones et les valeurs réelles des observations utilisées pendant l'apprentissage. Plus le résultat de cette fonction est minimisé, plus le réseau de neurones est performant. Sa minimisation, c'est-à-dire réduire au minimum l'écart entre la valeur prédite et la valeur réelle pour une observation donnée, se fait en ajustant les différents poids du réseau de neurones. Nous avons utilisé les fonctions de perte MSE pour Unet et Dice pour FCN.

7.3.6 Métrique

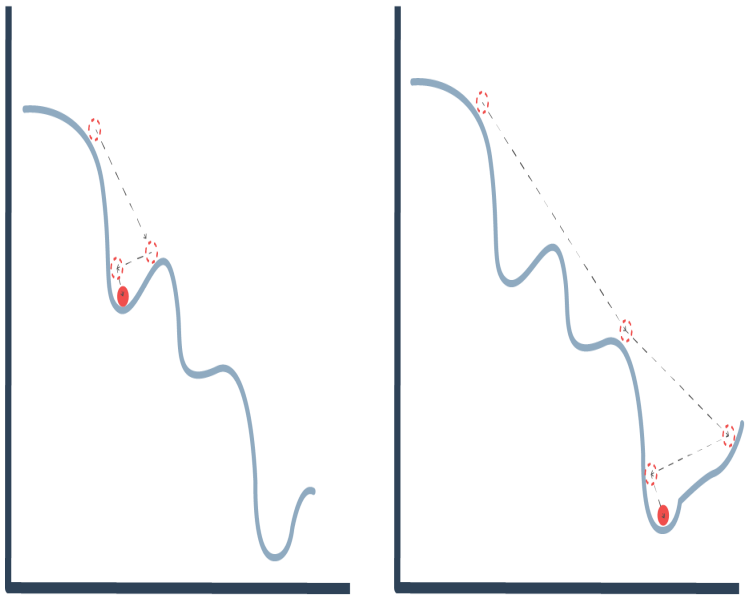
La métrique Intersection over Union (IoU), également appelée indice Jaccard, est une méthode pour quantifier le pourcentage de chevauchement entre le masque cible et notre sortie de prédiction. Cette métrique mesure le nombre de pixels

communs entre la cible et les masques de prédiction divisé par le nombre total de pixels présents sur les deux masques. Nous avons utilisé cette métrique pour la fiabilité de son score.



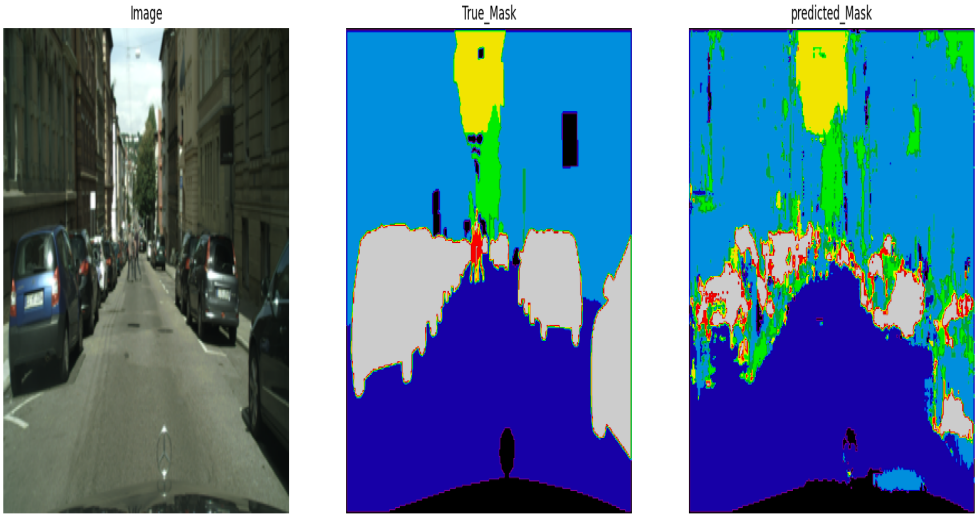
7.3.7 Learning Rate

Un taux d'apprentissage plus élevé signifie que les poids sont modifiés davantage à chaque itération, de sorte qu'ils peuvent atteindre leur valeur optimale plus rapidement, mais peuvent également manquer l'optimum exact. Un taux d'apprentissage plus faible signifie que les poids sont moins modifiés à chaque itération, il peut donc prendre plus de temps pour atteindre leur valeur optimale, mais ils sont moins susceptibles de manquer les optima de la fonction de perte.



8 Resultat

Voici un exemple de prédiction que nous avons eu avec notre modèle Unet.



9 Conclusion

En réglant tous les paramètres de nos modèles, nous avons obtenu des scores bien différents. En ce qui concerne FCN, nous n'avons pas réussi à aller au-dessus de 28%, et 41% pour Unet. L'entrainement des modèles est un processus très long au vu de la quantité de données nécessaire, malgré le générateur de données. Essayer les différents paramètres peut donc s'avérer difficile.

10 Axe d'amélioration

Nous aurions pu améliorer nos modèles en procédant différemment sur la data augmentation, en effet, la différence entre un modèle entrainer avec la data augmentation et sans n'est pas très grande, il se peut que nous ayons mal agi, par exemple en ajoutant un effet de sombre aux images. Nous aurions pu également tester plus de fonction de perte, d'optimiseur, jouer un peu plus avec les convolutions et leurs paramètres. Le problème est le temps d'entrainement des modèles qui est très long (pouvant aller jusqu'à 6h).