

Participez à la conception d'une voiture autonome

<i>Introduction</i>	2
<i>Etat de l'art</i>	2
Segmentation d'image	2
Cityscapes.....	4
<i>Deep Learning pour segmentation d'Images</i>	5
Un modèle de réseau neurones convolution	5
FCN (Fully Convolutional Networks).....	8
U-Net.....	8
PSPNet (Pyramid Scene Parsing Network)	9
<i>Description des données</i>	9
<i>Méthodologie</i>	11
Importation des données.....	11
Preprocessing	11
Choix de l'architecture	12
Générateur des données.....	12
Entrainement.....	12
Évaluation.....	12
Optimisation des hyper-paramètres	14
Exécution finale	15
Prédictions.....	15
<i>Lecture des données de grand volume et augmentation des données.</i>	15
Les générateurs	15
Data Augmentation	16
<i>Comparaison des modèles</i>	18
FCN.....	18
U-Net.....	19
PSPNet.....	19
Augmentation des données	20
Optimisation de la fonction de perte	20
Modèle final	21
<i>Axes amélioration</i>	22
<i>Sources</i>	23

Introduction

Les systèmes embarqués de vision par ordinateur pour les véhicules autonomes contient plusieurs parties :

- acquisition des images en temps réel
- traitement des images
- segmentation des images
- système de décision

Le présent document a pour objet de décrire la partie « segmentation d'images ».

L'objectif de cette partie est d'entraîner et déployer un modèle de segmentation des images s'intégrant facilement dans la chaîne complète du système embarqué.

Les consignes à respecter :

- Les données à utiliser : [Cityscapes](#)
- Le résultat final doit être simple à utiliser.
- Prendre en compte les faits suivants :
- Le système d'acquisition n'est pas stable.
- Le volume de données est important.

Etat de l'art

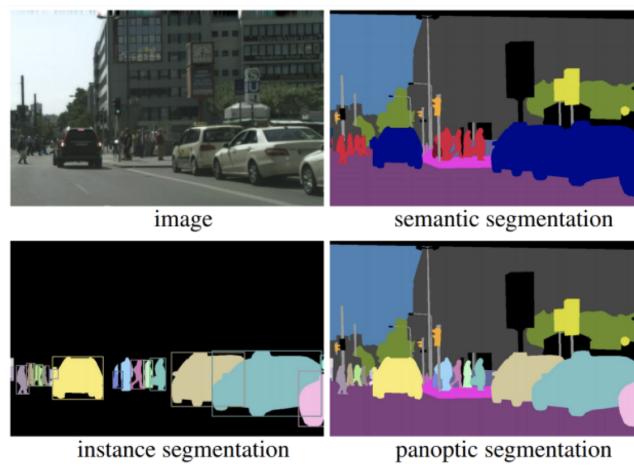
Segmentation d'image

La segmentation d'images est l'une des tâches très importantes en vision par ordinateur.

Les applications de la segmentation d'images sont très variées : elle est indispensable dans la conduite de voiture autonome, le diagnostic médical, l'essai virtuel de vêtements et la recherche visuelle.

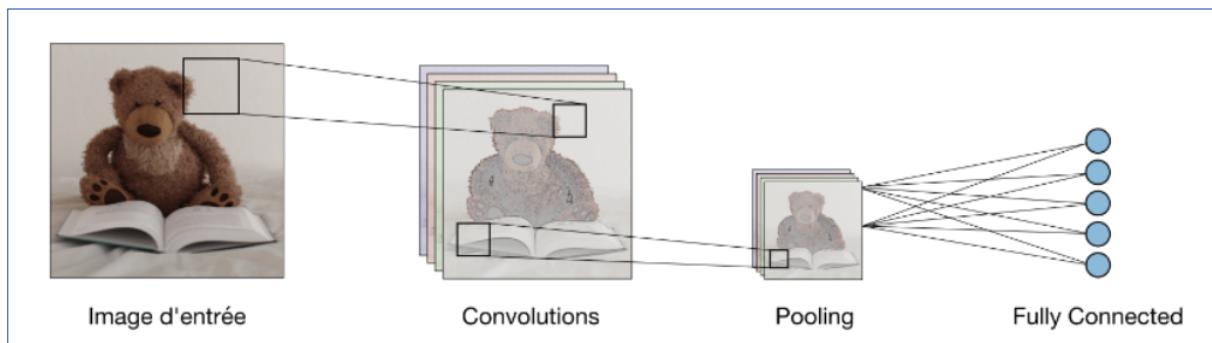
Grâce aux progrès des réseaux neuronaux profonds et à l'accès aux données à grande échelle, la segmentation d'images a connu des avancées significatives.

La segmentation d'images est un moyen de séparer une image en plusieurs régions en fonction des propriétés des pixels. Elle peut être de deux types : la segmentation sémantique et la segmentation d'instance. La version combinée de ces deux tâches est connue sous le nom de segmentation panoptique.



Example of different types of image segmentation [Image source](#)

Au cœur de la réussite du Deep Learning dans ce domaine se trouve un type de modèle appelé CNN (Convolutional Neural Networks). Un CNN fonctionne généralement en extrayant des features des images, puis en alimentant ces features dans un réseau neuronal entièrement connecté pour générer une prédiction. Les couches d'extraction de features du réseau ont pour effet de réduire la taille du feature map potentiellement énorme à un plus petit ensemble de features en prenant en charge la prédiction d'étiquette.



Fully Convolutional Network (FCN) a été la première approche à utiliser des couches de convolution pour remplacer les couches entièrement connectées pour la segmentation sémantique.

De nombreuses méthodes de segmentation sémantique récentes basées sur le FCN ont donné des résultats impressionnantes. Parmi ces travaux, l'une des approches les plus efficaces est l'utilisation d'informations contextuelles de la scène pour améliorer la compréhension de divers objets au niveau sémantique. Cela permet d'apprendre non seulement la sémantique des pixels individuels, mais également les structures spatiales qui les relient. Ce qui est particulièrement adapté à la segmentation sémantique d'images.

DeconvNet , SegNet et RefineNet etc., utilisent la structure codeur-décodeur pour combiner les informations de bas niveau et de haut niveau afin d'optimiser les résultats de la segmentation.

De plus, le champ aléatoire de Markov (MRF) et le champ aléatoire conditionnel (CRF) ont été largement utilisés pour modéliser les dépendances à longue distance pour prédire la structure fine dans la segmentation sémantique [1]

Lors du projet en cours, nous avons étudié quelques modèles populaires de segmentation sémantique, décrit leurs architectures et comparé leurs performances.

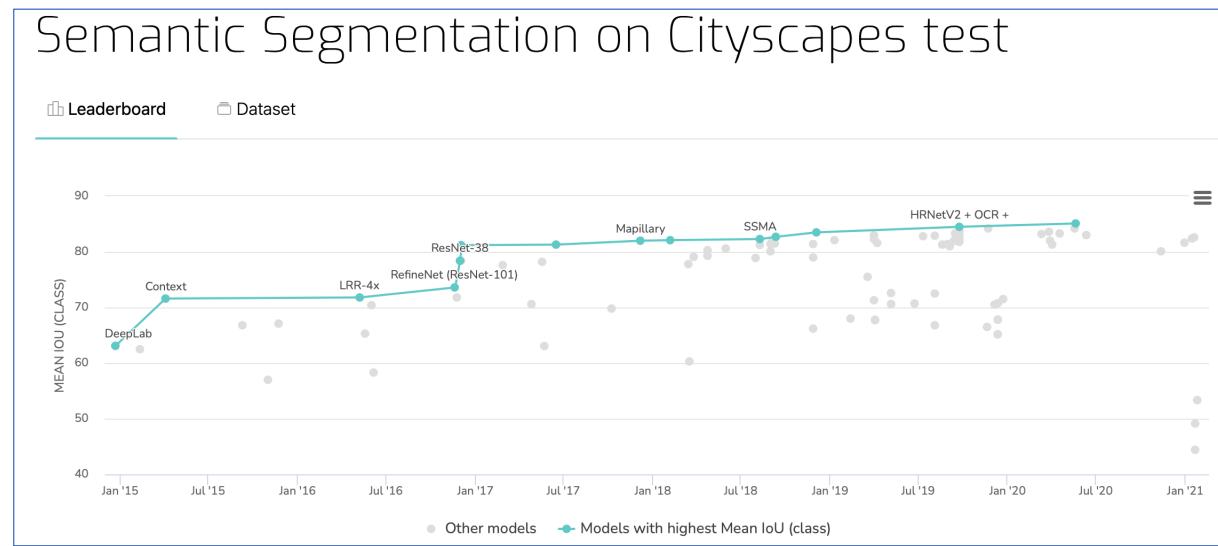
Cityscapes

Les données Cityscapes font partie des données open-sources disponibles pour la compréhension de scénarios urbains complexes.

Le jeu de données Cityscapes est destiné à

- évaluer les performances des algorithmes de vision pour les principales tâches de compréhension sémantique des scènes urbaines : étiquetage sémantique au niveau du pixel, de l'instance et panoptique ;
- soutenir la recherche visant à exploiter de grands volumes de données annotées, par exemple pour l'entraînement de réseaux neurones profonds.

Plusieurs tests ont été effectués en utilisant différents modèles sur ces données depuis leur apparition :



Deep Learning pour segmentation d'Images

Un modèle de réseau neurones convolution

Pour la tâche la segmentation d'images l'utilisation du CNN est la meilleure solution. Un CNN extrait des caractéristiques des images, puis les intègre dans un réseau neuronal entièrement connecté pour générer une prédiction. Il se compose de plusieurs couches, chacune effectuant une tâche spécifique dans l'extraction des features ou la prédiction des étiquettes.

Il existe trois types des couches :

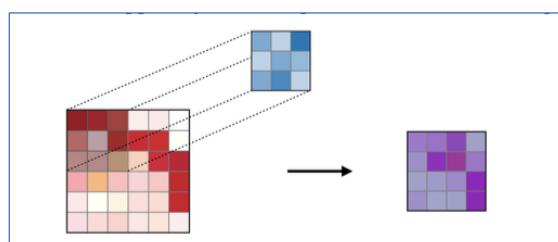
- **Couche d'entrée** : prend en charge le nombre de features que le (modèle traitera
- **Couche cachée** : effectue différentes tâches spécifiques dans l'extraction des features. Il est possible de choisir le nombre de couches masquées et le nombre de neurones dans chacun d'eux. Cependant, aucun contrôle sur les valeurs d'entrée et de sortie pour ces couches n'est possible. Celles-ci sont déterminées par le processus d'apprentissage du modèle.
- **Couche de sortie** : contient un neurone pour chaque valeur de probabilité de classe devant être prédite par le modèle.

Description certaines couches masquées utilisées dans nos modèles :

Couche de convolution (CONV)

Extrait des features importantes dans les images en appliquant un filtre. Le filtre est une matrice de valeurs de poids.

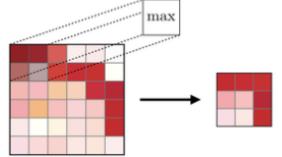
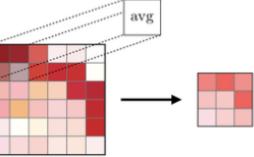
En entrée, la couche de convolution prend une image ou une feature map; en sortie, on trouve une feature map extraite grâce à la convolution.



La sortie de la convolution est généralement transmise à une fonction d'activation (ReLU). Elle a pour but d'introduire des complexités non linéaires au réseau.

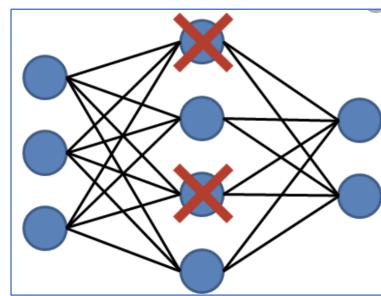
Couche de Pooling (POOL)

La couche de pooling est une opération de sous-échantillonnage typiquement appliquée après une couche de convolution afin de réduire la taille de la matrice de features . En particulier, les types de pooling les plus populaires sont le max et l'average pooling, où les valeurs maximales et moyennes sont prises, respectivement. Les couches de Pooling fonctionnent en appliquant le filtre.

Max pooling	Average pooling
Chaque opération de pooling sélectionne la valeur maximale de la surface	Chaque opération de pooling sélectionne la valeur moyenne de la surface
	

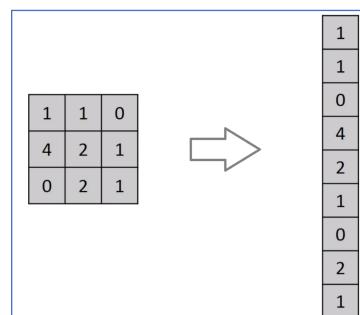
Couche de Suppression (Dropout)

Élimine des features map de façon aléatoire. Utilisée pour atténuer le surentraînement.



Couche d'Aplatissement (Flatten)

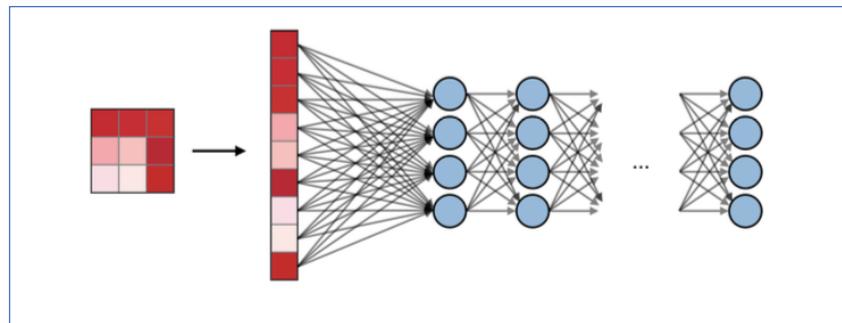
Aplatit les caractéristiques en un vecteur de valeurs qui peuvent être utilisées comme entrée dans une couche entièrement connectée.



Couche Fully Connected (FC)

Réseau neuronal standard utilisé pour la classification. La couche de fully connected s'applique sur une entrée préalablement aplatie où chaque entrée est connectée à tous les neurones.

Les couches de fully connected sont typiquement présentes à la fin des architectures de CNN et peuvent être utilisées pour optimiser des objectifs tels que les scores de classe.

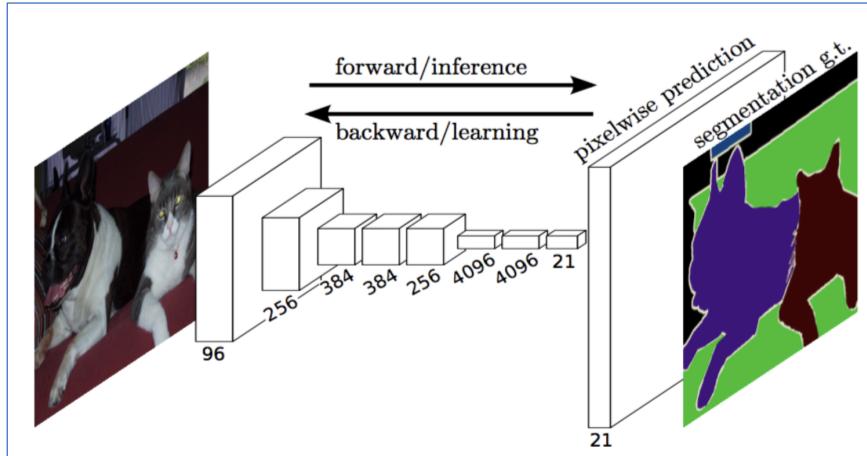


A retenir

- Chaque couche reçoit en entrée des données et les renvoie transformées. Pour cela, elle calcule une combinaison linéaire puis applique éventuellement une fonction non-linéaire, appelée fonction d'activation. Les coefficients de la combinaison linéaire définissent les paramètres (ou poids) de la couche
- Un réseau de neurones est construit en empilant les couches : la sortie d'une couche correspond à l'entrée de la suivante.
- Cet empilement de couches définit la sortie finale du réseau comme le résultat d'une fonction différentiable de l'entrée

Comme modèle de référence, le modèle FCN (Fully Convolutional Networks) a été utilisé. Nous avons testé également les modèles U-Net et PSPNet(Pyramid Scene Parsing Network).

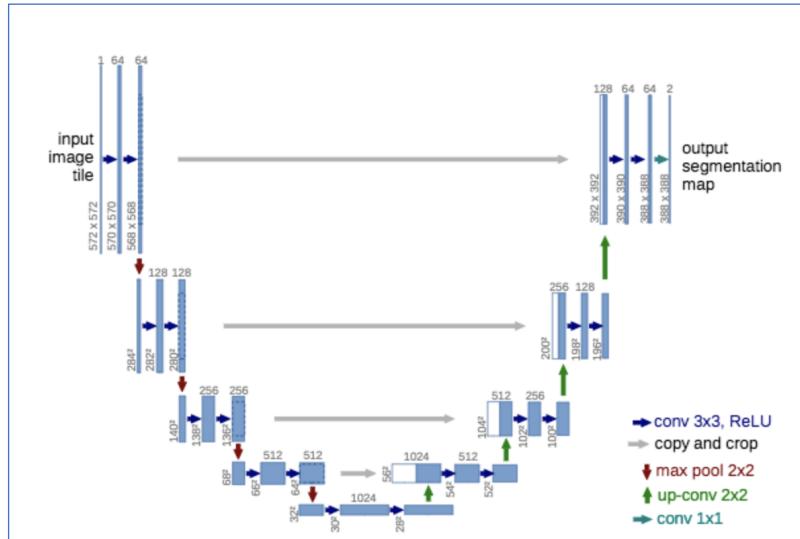
FCN (Fully Convolutional Networks)



Architecture of FCN [Image source](#)

FCN est l'un des premiers modèles proposés pour la segmentation sémantique complète. Dans un FCN, les convolutions transposées sont utilisées pour sur-échantillonner, contrairement à d'autres approches où des interpolations mathématiques sont utilisées.

U-Net

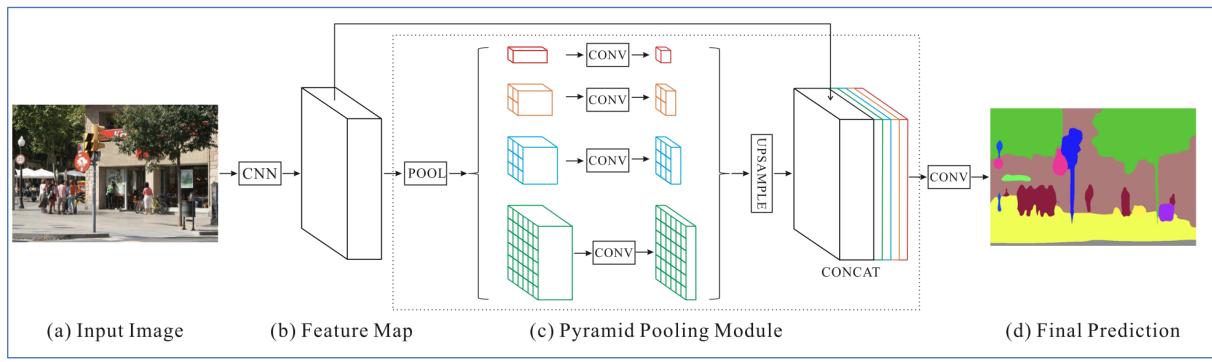


Architecture of U-Net [Image source](#)

Un U-Net se compose d'un encodeur (downsampler) et d'un décodeur (upsampler) avec des skip-connection.

Les couches encodeur et décodeur sont symétriques l'une à l'autre.

PSPNet (Pyramid Scene Parsing Network)



Architecture of PSPNet [Image source](#)

Le Pyramid Scene Parsing Network est optimisé pour apprendre une meilleure représentation du contexte global d'une scène. Tout d'abord, l'image est transmise au réseau de base pour obtenir une feature map. Puis la feature map est sous-échantillonnée à différentes échelles et la convolution est appliquée. Ensuite, toutes les feature map sont ré-échantillonnées à une échelle commune et concaténées ensemble. Enfin, une autre couche de convolution est utilisée pour produire les résultats de la segmentation finale. Ici, les petits objets sont bien capturés par les caractéristiques regroupées à une haute résolution, tandis que les grands objets sont capturés par les features regroupées à une plus petite taille.

Description des données

Les données Cityscapes sont composées d'un ensemble de séquences vidéo enregistrées dans les rues des villes différentes. La plupart de ces images ont des annotations de haute qualité au niveau du pixel.

Diversité des données

- 50 villes
- Différentes saisons (printemps, été, automne)
- Différents moments de la journée
- Conditions météorologiques bonnes/moyennes

Annotations :

- Grand nombre d'objets dynamiques
- Disposition variable de la scène
- Arrière-plan variable

Les images sont des données à 3 dimensions :

- hauteur
- longueur
- couleur (Rouge, Vert, Bleu)

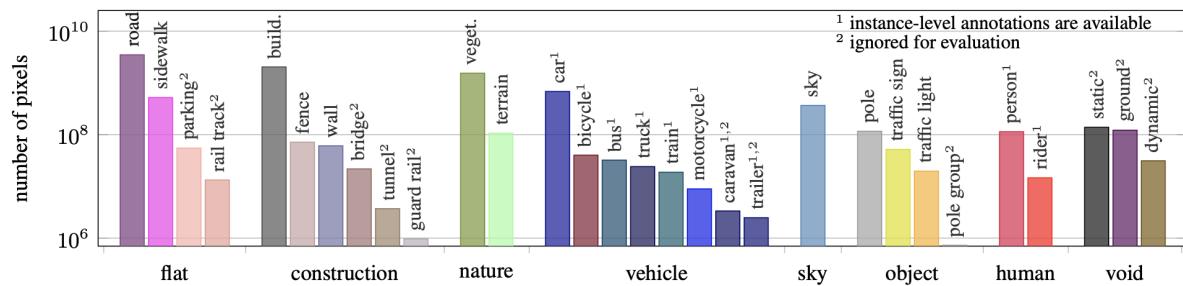
Où chaque couleur (RVB) est représenté sur une échelle de nuance de 0 à 255.

NOTE TECHNIQUE - MODELE DE SEGMENTATION DES IMAGES

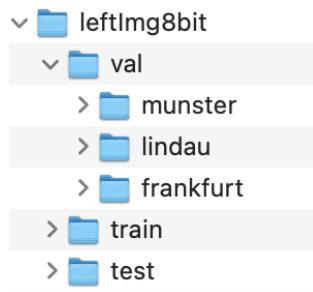
Les données sont séparées en trainset, validset et testset :



Il existe 8 catégories de labels



Structure des données images :



Structure des données labellisées :



Pour les données labellisées, uniquement les fichiers *_color.png ont été utilisés.

Méthodologie

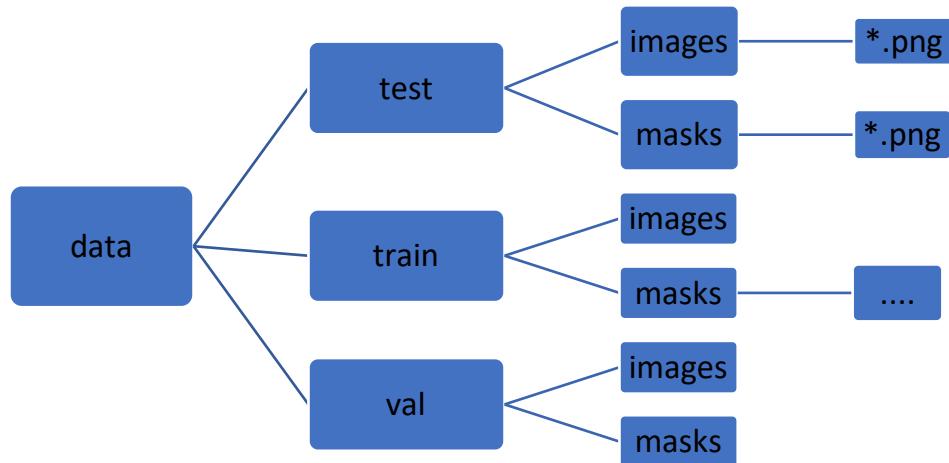
Importation des données.

L'ensemble de données contient déjà les séparations requises :

- entraînement
- validation
- test

Nous avons utilisé le même fractionnement.

Les données Cityscapes ont été importées et sauvegardées dans une structure comme suit :



Les images et masques correspondant ont été renommés afin d'avoir le même nom que les fichiers dans les répertoires « images » et « masks ».

Preprocessing

Les algorithmes Deep Learning ne nécessitent qu'un minimum de prétraitement des données car, les CNN font déjà le travail de traitement d'une image et d'extraction des features.

Une contrainte importante du CNN, est la nécessité de redimensionner les images à une dimension unifiée. Les images doivent être prétraitées pour avoir une largeur et une hauteur identiques avant d'être transmises à l'algorithme d'apprentissage.

La normalisation est l'étape la plus cruciale du prétraitement. Il s'agit de remettre à l'échelle les valeurs des pixels afin qu'elles se situent dans une plage restreinte afin de faciliter la propagation des gradients.

Dans notre cas, le redimensionnement et normalisation par division sur 255 (applicable pour les images RGB) ont été appliqués.

```
norm1_image = image/255
```

Nous avons également utilisé l'augmentation des données. Cette technique permet d'élargir l'ensemble de données et d'exposer le CNN à une grande variété des images. Plusieurs techniques ont été testées (rotation, scaling, contraste aléatoire, etc) afin de sélectionner les techniques donnant le meilleur résultat. La partie d'augmentation des données a été intégrée dans data générateur et décrite dans le paragraphe [Data Augmentation](#).

Pour les masques nous avons effectué des traitements complémentaires :

- Regroupement des sous-catégories en catégorie pour en avoir 8 au lieu de 32. Pour cela nous avons effectué un remplacement des couleurs afin d'avoir une couleur par catégorie.
- Utilisation de la technique one-encoding. Cela consiste à créer pour chaque pixel un vecteur de booléen. Celui-ci sera de taille n, correspondant au nombre de classes total présent au sein de le dataset.

Choix de l'architecture

Pour la tâche de segmentation d'images, un modèle à couches de convolution est la meilleure solution.

Nous avons testé trois modèles afin de choisir le plus performant :

- Le modèle FCN a été utilisé comme un modèle de référence.
- Le réseau U-Net avec les couches encoder-décoder
- Le modèle PSPNet, avec ResNet pré-entraîné utilisé comme encodeur.

Générateur des données

Afin d'accélérer l'entraînement du modèle un générateur des données personnalisé permettant de traiter les données par lot a été créé.

Entrainement

Nous avons compilé et entraîné les modèles.

Après le choix du modèle le plus performant, il a été entraîné sur un nombre d'epochs importants afin de vérifier quand le modèle fait de l'overfitting et sélectionner le nombre d'epochs suffisant.

Évaluation

Pour évaluer nos modèles, nous avons testé trois métriques ci-dessous :

Pixel Accuracy

Il s'agit du pourcentage de pixels de l'image qui ont été classés correctement. Cependant, cette métrique n'est pas efficace dans le cas des données déséquilibrées.

Intersection-Over-Union (IoU), Jaccard Index

L'Intersection-Over-Union (IoU), également connue sous le nom d'indice de Jaccard, est l'une des métriques les plus couramment utilisées dans le domaine de la segmentation séquentielle.

Comme son nom l'indique, nous divisons l'intersection (le nombre de pixels similaires) par l'union (nombre total de pixels uniques dans les deux images)

Cette métrique s'étend de 0 à 1 (0-100%), 0 signifiant l'absence de chevauchement et 1 signifiant un chevauchement parfait de la segmentation. Pour une segmentation multi-classes, l'IoU moyen de l'image est calculé en prenant l'IoU de chaque classe et en en faisant la moyenne.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

IoU

$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$

Dice Coefficient (F1 Score)

Le coefficient Dice est très similaire à l'IoU. Ils sont positivement corrélés. Dans ce cas nous comptons les pixels similaires (en prenant l'intersection, présents dans les deux images) dans les deux images que nous comparons et nous les multiplions par 2, puis nous les divisons par le nombre total de pixels dans les deux images

$$\frac{2|X \cap Y|}{|X| + |Y|}$$

Dice Coefficient

$\frac{2 \times \text{Intersection Area}}{\text{Area of Circle 1} + \text{Area of Circle 2}}$

Comme le coefficient IoU, il va de 0 à 1, 1 signifiant la plus grande similitude entre la prédiction et la vérité.

Ces trois métriques ont été étudiées lors de comparaison des modèles. Cependant, IoU a été sélectionné arbitrairement comme une métrique de référence.

Optimisation des hyper-paramètres

La fonction de perte (*loss function*) est utilisée pour optimiser le réseau de neurones. Elle est associée à la couche finale pour calculer l'erreur de classification.

Nous avons testé différentes fonctions pour améliorer notre modèle.

Categorical Cross Entropy

La fonction de Categorical Cross Entropy est largement utilisée pendant l'apprentissage de la segmentation des images. Cependant, cette fonction est impactée par le déséquilibre des classes.

Pour obtenir les informations granulaires d'une image, il faut recourir à des fonctions de perte plus avancées.

$$L(gt, pr) = -gt \cdot \log(pr)$$

gt – ground truth

pr- prediction

Categorical Focal Loss

Cette fonction de perte est une amélioration de l'entropie croisée. Sa forme est modifiée de telle sorte à ce que la perte attribuée aux exemples bien classés est pondérée à la baisse. En fin de compte, cela garantit qu'il n'y a pas de déséquilibre entre les classes. Dans cette fonction de perte, la perte d'entropie croisée est mise à l'échelle, les facteurs d'échelle diminuant à zéro à mesure que la confiance dans les classes correctes augmente. Le facteur d'échelle pondère automatiquement la contribution des exemples faciles au moment de la formation et se concentre sur les exemples difficiles.

$$L(gt, pr) = -gt \cdot \alpha \cdot (1 - pr)^\gamma \cdot \log(pr)$$

gt – ground truth

pr- prediction

JaccardLoss

La fonction Jaccard loss est Inspiré du coefficient IoU, métrique utilisée pour évaluer la performance d'un réseau de segmentation sémantique d'image.

Cette fonction de perte de classification équilibrée par IoU vise à augmenter le gradient des échantillons avec un IoU élevé et à diminuer le gradient des échantillons avec un IoU faible. De cette façon, la précision de localisation des modèles d'apprentissage automatique est augmentée.

$$L(A, B) = 1 - \frac{A \cap B}{A \cup B}$$

Dice loss

La fonction Dice loss est inspiré du coefficient Dice, métrique utilisée pour évaluer la performance en mesurant le chevauchement entre deux objets.

$$L(precision, recall) = 1 - (1 + \beta^2) \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

Elle est la plus couramment utilisée dans les problèmes de segmentation.

Exécution finale

Le modèle avec les paramètres et nombre d'epochs sélectionnés a été entraîné et sauvegardé.

Prédictions

Nous avons effectué quelques prédictions pour visualiser le résultat

Lecture des données de grand volume et augmentation des données.

Les générateurs

Les générateurs sont des fonctions permettant de stocker des images, des variables et des éléments après leur chargement dans l'environnement.

Les générateurs ne calculent pas la valeur de chaque élément, ils calculent les éléments uniquement lorsqu'on leur demande de le faire. Cette technique s'appelle « lazy evaluation ».

« Lazy evaluation » est utile lorsque l'on a un très grand nombre de données à calculer. Elle permet d'utiliser les données déjà calculées, pendant que le reste des données est en cours de calcul.

Les générateurs permettent un gain significatif de rapidité et d'espace mémoire.

Cette technique est particulièrement utile pour l'entraînement d'un modèle de Deep Learning qui fonctionne sur des lots de données(batch). Les lots sont chargés par itération lorsque le modèle en a besoin.

Keras implémente la classe *ImageDataGenerator* permettant de charger les données de grand volume et effectuer une augmentation des données.

Cependant, elle ne nous convient pas, car nous avons besoins de faire un prétraitement spécifique pour les images lors de chargement (comme, par exemple, remplacement de la couleur dans les fichiers « mask » pour avoir 8 catégories au lieu de 32).

Un générateur personnalisé a été créé pour ce projet. Une class DataGenerator (`keras.utils.Sequence`) a été redéfinie en prenant en compte le traitement spécifique à notre projet.

Data Augmentation

La technique de la Data Augmentation permet d'augmenter la quantité des données (les images dans notre cas) pour l'entraînement des modèles Deep Learning et éviter l'overfitting.

La Data Augmentation crée des images artificielles à partir d'une image réelle en appliquant différentes transformations aléatoires.

Il existe deux méthodes d'augmentation :

Online augmentation : les images sont augmentées en direct lors de l'entraînement du réseau.

Offline augmentation : les images sont augmentées en amont de l'entraînement.

La méthode online augmentation a été sélectionnée dans notre cas car elle permet de s'assurer que le réseau voit des images variées à chaque epoch.

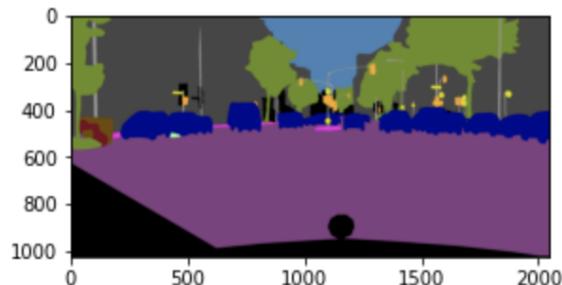
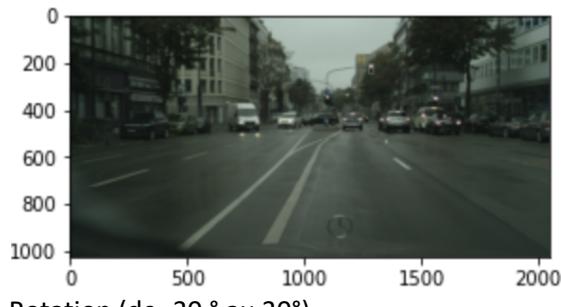
Les images ne sont pas sauvegardées en mémoire, et le réseau ne rencontre jamais deux fois la même image.

Différents types de transformation sont possibles : rotation, recadrage, translation, éclairage, mise à l'échelle, ajout de bruit, etc.

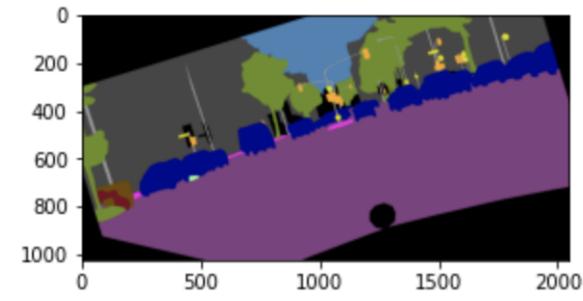
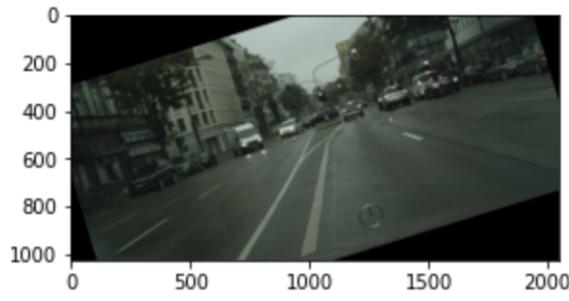
NOTE TECHNIQUE - MODELE DE SEGMENTATION DES IMAGES

Pour notre projet les transformations suivantes ont été testées :

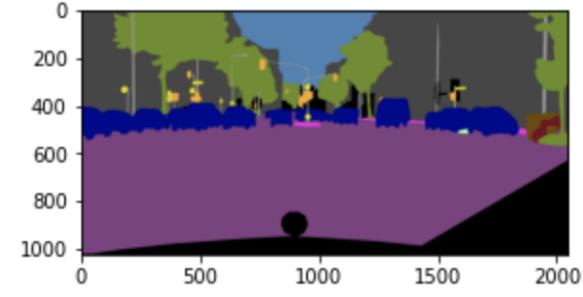
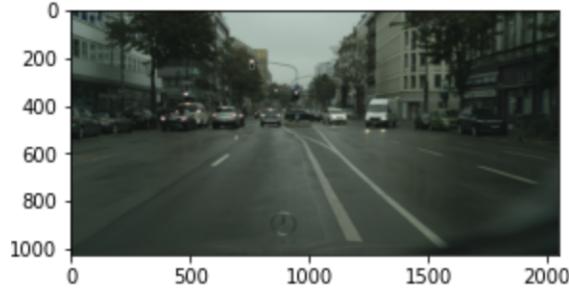
Image d'origine



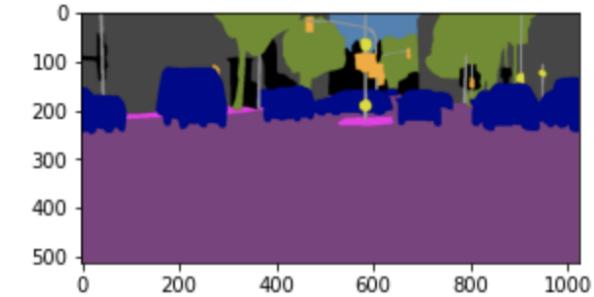
Rotation (de -30° au 30°)



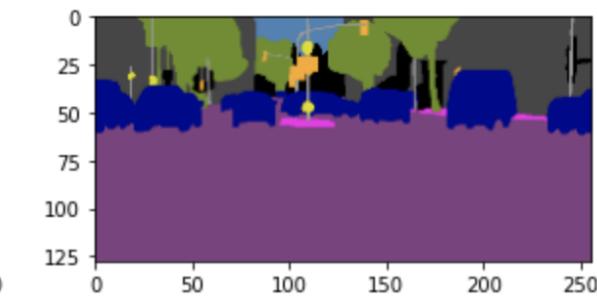
Effet de miroir



Recadrage



Modification de la luminosité et du contraste



Ces transformations ont été définies dans une fonction `data_augmentation` de la classe `DataGenerator` personnalisée.

Donc, notre modèle apprendra sur beaucoup plus de données tout en ne rencontrant jamais deux fois la même image.

La Data Generation permet au modèle d'être exposé à davantage de données et de mieux généraliser.

Comparaison des modèles

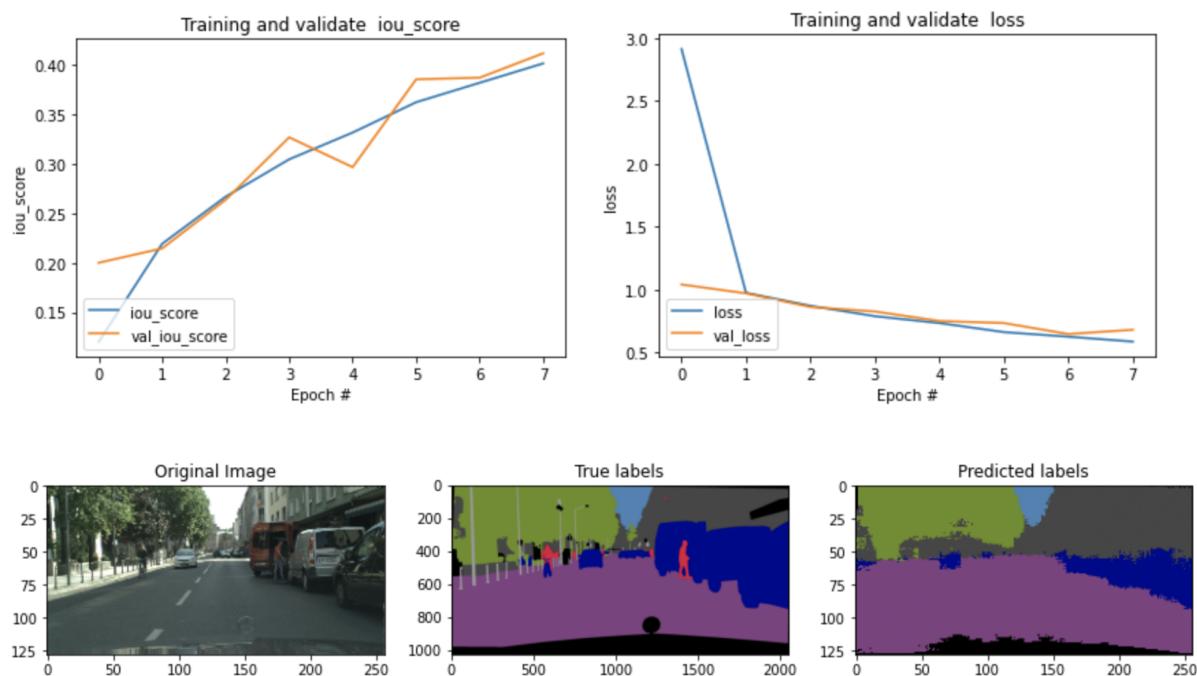
Nous avons entraîné trois modèles :

- FCN
- U-Net
- PSPNet avec un modèle ResNet pré-entraîné utilisé comme encodeur

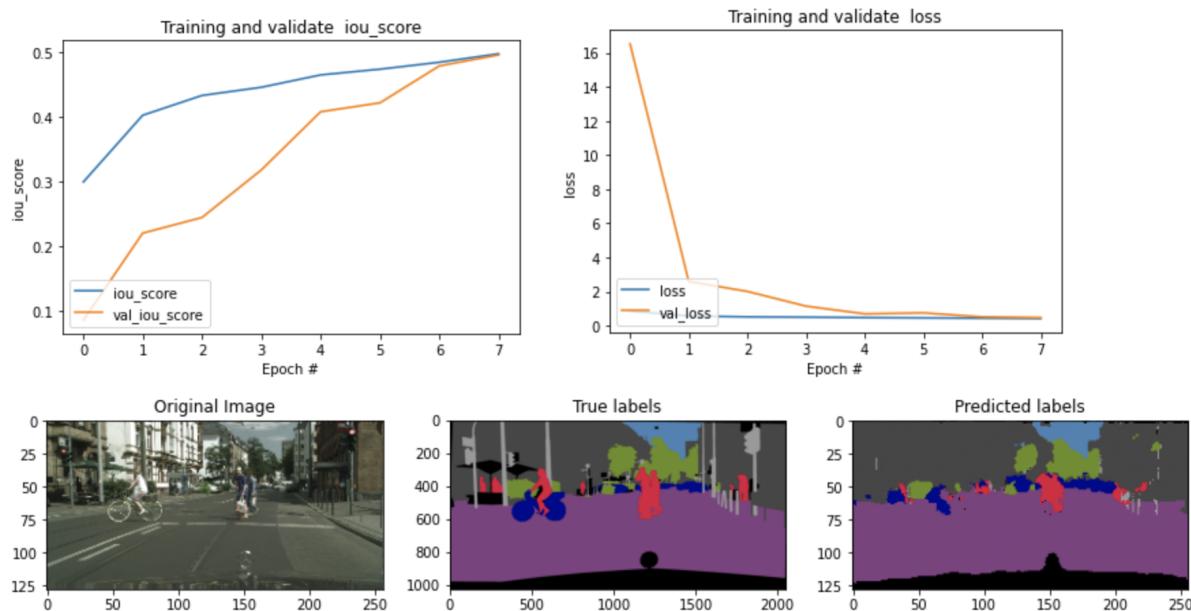
Avec les paramètres suivants :

- 500 images sélectionnées aléatoirement
- Nombre d'epochs 8
- Taille d'image 128x256 et 194x288 pour PSPNet
- Loss : `categorical_crossentropy`

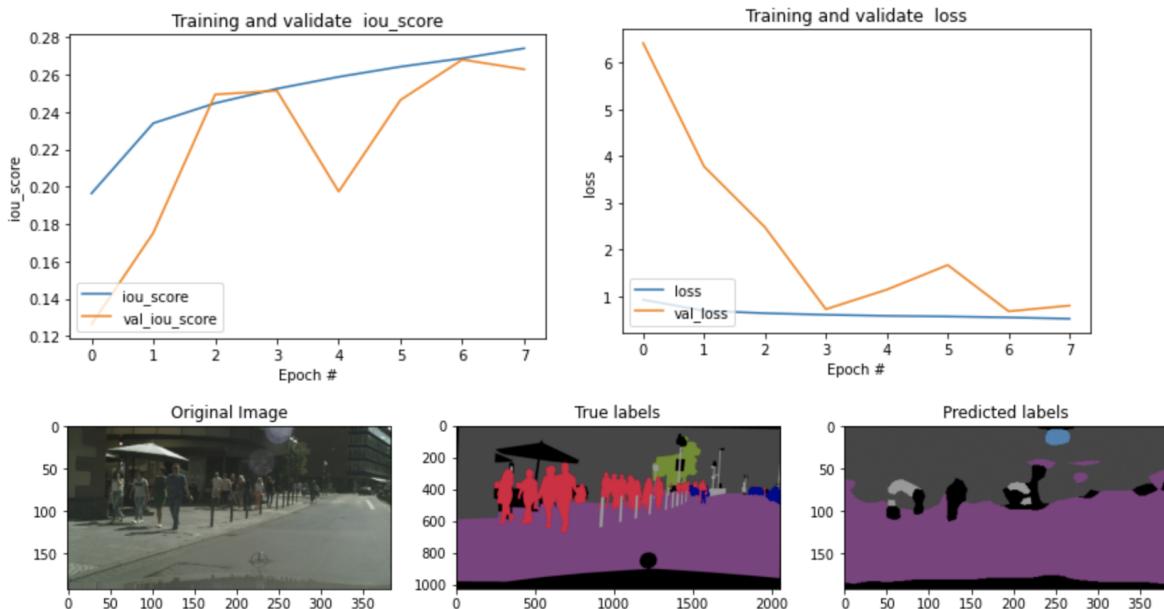
FCN



U-Net



PSPNet



Résultat final :

Model	Data augmentation	Categorical accuracy	Loss	Dice Coefficient	Intersection-Over-Union	Build training time	Predict data time
0 FCN	No	79.238462	0.681045	52.714139	41.285670	00:20:24.18	00:00:38.88
1 U-Net	No	84.781492	0.491049	61.347270	49.478778	00:15:36.52	00:00:38.65
2 PSPNet	No	75.543064	0.803589	38.535050	26.256573	00:18:34.18	00:00:48.12

Nous avons constaté que U-Net a montré la meilleure score IoU.

Augmentation des données

Model	Data augmentation	Categorical accuracy	Loss	Dice Coefficient	Intersection-Over-Union	Build training time	Predict data time
0	U-Net Random crop	80.467063	0.567984	54.512894	41.998458	00:18:38.90	00:00:41.14
1	U-Net Rotation	81.326109	0.605095	56.319416	44.245347	00:29:16.55	00:00:42.71
2	U-Net No	83.482659	0.515926	57.460308	45.169681	00:19:22.66	00:00:44.48
3	U-Net Brightness Contrast	81.936610	0.583728	58.283561	45.969439	00:18:24.36	00:00:40.97
4	U-Net Flip left right	83.307791	0.592340	60.249293	48.261976	00:31:47.44	00:00:44.02
5	U-Net Flip left right and Brightness Contrast	82.963610	0.615495	59.686375	48.668182	00:14:42.16	00:00:43.62

Nous avons testé plusieurs techniques d'augmentation de données. Finalement, nous avons constaté que les deux techniques : Flip left right (effet miroir) et Brightness Contrast (changement aléatoire de la luminosité et du contraste) augmentent la valeurs IoU. Nous les avons donc utilisées lors de l'entraînement de notre modèle.

Optimisation de la fonction de perte

Model	Data augmentation	Loss type	Loss	Dice Coefficient	Intersection-Over-Union	Categorical accuracy
0	U-Net Yes	categorical_crossentropy	0.084604	54.574651	43.376240	79.709393
1	U-Net Yes	focal_loss	0.011332	46.517482	33.667034	79.366159
2	U-Net Yes	dice_loss	0.405394	59.460545	46.992576	80.340624
3	U-Net Yes	jaccard_loss	0.496699	62.027341	50.330079	81.976974

Nous avons obtenu le meilleur résultat avec la fonction « jaccard loss »

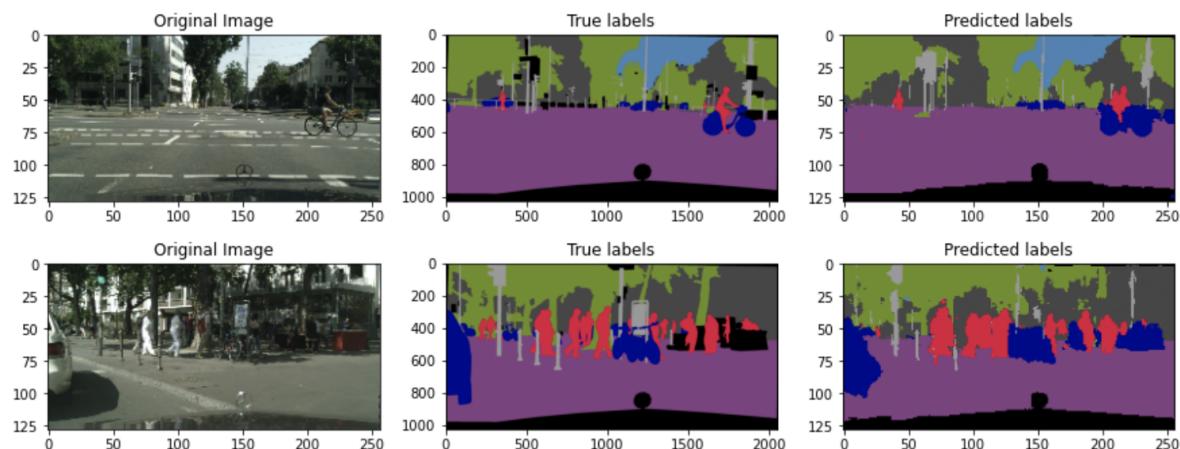
Modèle final

Le modèle final U-Net a été entraîné sur 2000 images avec les paramètres suivants :

- 30 époques
- Taille d'image 128x256
- Optimiser : évaluation de la dynamique adaptative (Adam)
- Metric : "IoU"
- Loss : "Jaccard Loss"



Exemples de prédiction finale :



Axes amélioration

Nous avons identifié plusieurs axes d'amélioration. Pour les mettre en place, une machine avec un GPU (Graphics Processing Unit) puissant est probablement nécessaire.

Travailler avec les images de taille plus élevé (les images disponibles pour l'entraînement ont la taille 512x1204)

Gérer le déséquilibre des classes - implémenter la pondération du poids pour chaque classe

Tester les fonctions loss plus complexes telle que Boundary loss (Perte au niveau des frontières) ou Weighted cross-entropy (Entropie croisée pondérée)

Augmenter le nombre d'epochs afin d'obtenir le résultat le plus précis

Utiliser un encodeur personnalisé pour PSPN au lieu d'un encodeur pré-entraîné.

Tester d'autres algorithmes de segmentation d'image

Pour pouvoir effectuer tous ces tests, une machine assez puissante est nécessaire.

Sources

1. DCANet: Dense Context-Aware Network for Semantic Segmentation
<https://arxiv.org/pdf/2104.02533.pdf>
2. Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey
<https://arxiv.org/pdf/2001.04074.pdf>
3. The Cityscapes Dataset for Semantic Urban Scene Understanding
<https://www.cityscapes-dataset.com/wordpress/wp-content/papercite-data/pdf/cordts2016cityscapes.pdf>
4. Data Augmentation, améliorer rapidement son modèle de Deep Learning
<https://inside-machinelearning.com/data-augmentation-ameliorer-rapidelement-son-modele-de-deep-learning/>
5. Image Segmentation in 2021: Architectures, Losses, Datasets, and Frameworks
<https://neptune.ai/blog/image-segmentation-in-2020>
6. Data Preprocessing and Network Building in CNN
[Data Preprocessing and Network Building in CNN | by Tanya Dayanand | Towards Data Science](https://towardsdatascience.com/data-preprocessing-and-network-building-in-cnn-by-tanya-dayanand-towards-data-science)
7. Metrics to Evaluate your Semantic Segmentation Model
<https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>
8. Introduction to Convolutional Neural Networks
<https://rubikscode.net/2018/02/26/introduction-to-convolutional-neural-networks/>
9. Loss-Function for Image Segmentation
<https://jeune-research.tistory.com/entry/Loss-Functions-for-Image-Segmentation-Distance-Based-Losses>
10. Segmentation Models Python API
<https://segmentation-models.readthedocs.io/en/latest/api.html>
11. Pense-bete-reseaux-neurones-convolutionnels
<https://github.com/afshinea/stanford-cs-230-deep-learning/blob/master/fr/pense-bete-reseaux-neurones-convolutionnels.pdf>