Note technique

INTRODUCTION

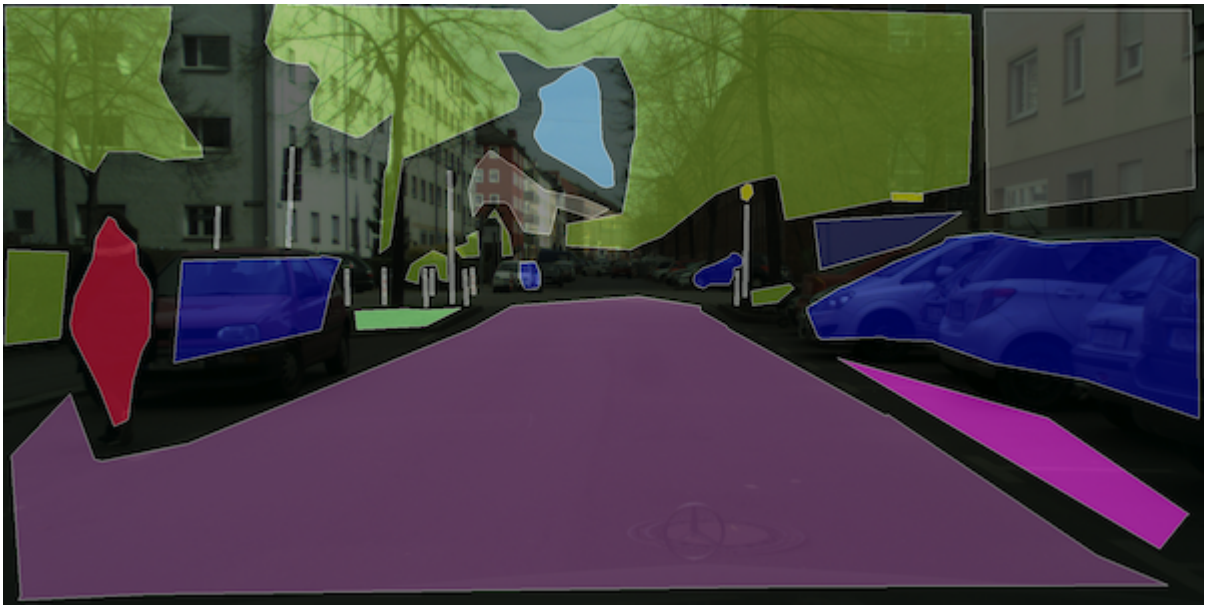
Je suis ingénieur IA au sein de l'équipe R&D de Future Vision Transport, une entreprise qui conçoit des systèmes embarqués de vision par ordinateur pour des véhicules autonomes.

Notre équipe est composée d'ingénieurs aux profils variés. Chacun des membres de l'équipe est spécialisé sur une des parties du système embarqué de vision par ordinateur.

Voici les différentes parties du système :

- acquisition des images en temps réel
- traitement des images
- segmentation des images (c'est vous !)
- système de décision

Je travaille sur la partie "segmentation des images". Je dois concevoir un modèle de segmentation d'images qui s'intégrera facilement dans la chaîne complète du système embarqué.



Voici le plan d'action :

- Entraîner un modèle de segmentation des images sur 8 catégories à partir du jeu de données "Cityscape Dataset"
- Concevoir une API de prédiction et une application web de présentation des résultats, qui seront déployés sur le cloud.

JEU DE DONNÉES

Le jeu de données est composé de 2 dossiers:

- **leftImg8bit** : ce dossier contient les images RGB d'origines. Ce sont les données d'éntrée
- **gtFine** : ce dossier contient les masques de segmentation. Ce sont les données de sortie






Les données d'entrée sont des images de 2048 x 1024 pixels sur 3 canaux (RGB). En ce qui concerne les données de sortie, il y a plusieurs types de fichiers, mais nous ne nous intéresserons qu'aux fichiers dont le nom se termine par "_labelIds.png". Ce sont les masques correspondant aux images RGB. Les valeurs des pixels des masques sont les identifiants des différents labels de segmentation. Il y a 32 catégories différentes, que nous devrons rapporter à 8 catégories principales.

Le dataset est séparé en 3 parties : un jeu de test (1525 images), un jeu d'entraînement (2975 images) et un jeu de validation (500 images).

ETAT DE L'ART

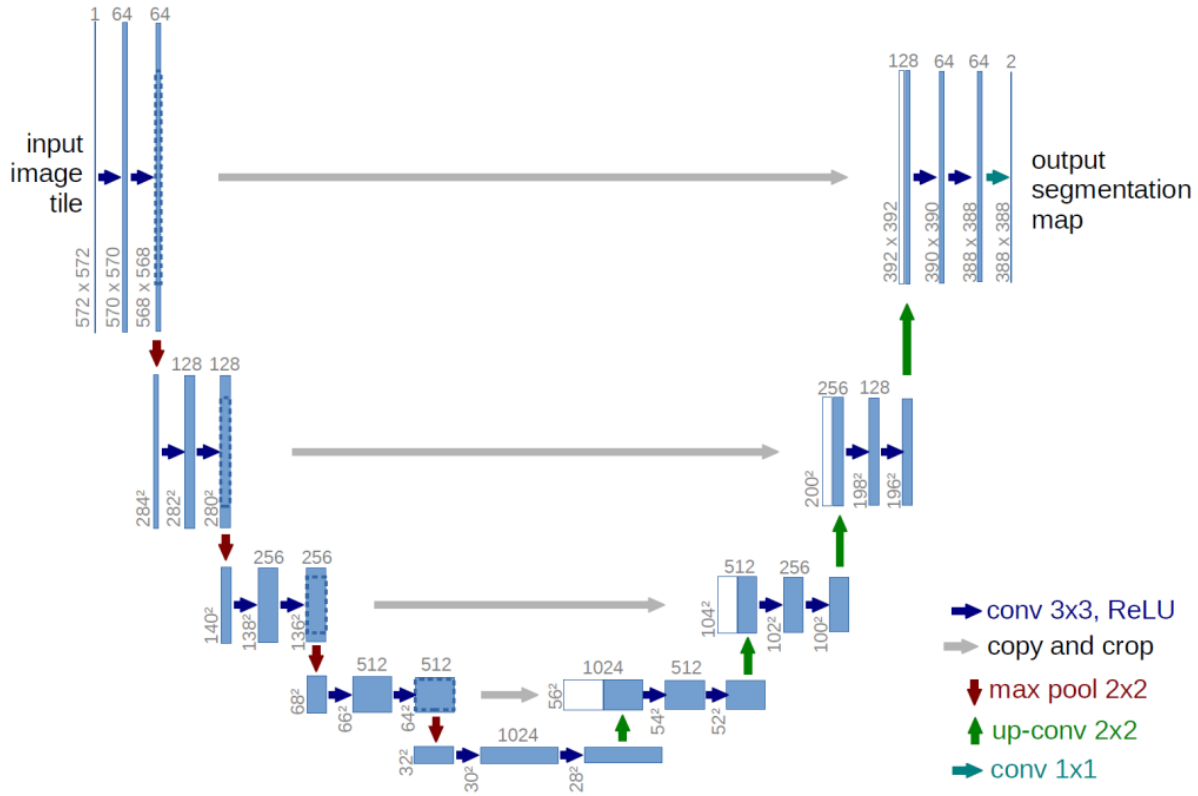
U-NET

Il s'agit d'un Modèle de Réseau de Neurones Entièrement Convolutif. Ce modèle fut initialement développé par Olaf Ronneberger, Phillip Fischer, et Thomas Brox en 2015 pour la segmentation d'images médicales.

L'architecture de U-NET est composée de deux » chemins « . Le premier est le chemin de contraction, aussi appelé encodeur. Il est utilisé pour capturer le contexte d'une image.

Il s'agit en fait d'un assemblage de couches de convolution et de couches de » max pooling » permettant de créer une carte de caractéristiques d'une image et de réduire sa taille pour diminuer le nombre de paramètres du réseau.

Le second chemin est celui de l'expansion symétrique, aussi appelé décodeur. Il permet aussi une localisation précise grâce à la convolution transposée.



Avantages de U-NET :

Dans le domaine du Deep Learning, il est nécessaire d'utiliser de larges ensembles de données pour entraîner les modèles. Il peut être difficile d'assembler de tels volumes de données pour résoudre un problème de classification d'images, en termes de temps, de budget et de ressources hardware.

L'étiquetage des données requiert aussi l'expertise de plusieurs développeurs et ingénieurs. C'est particulièrement le cas pour des domaines hautement spécialisés comme les diagnostics médicaux.

U-NET permet de remédier à ces problèmes, puisqu'il s'avère efficace même avec un ensemble de données limité. Il offre aussi une précision supérieure aux modèles conventionnels.

Une architecture autoencoder classique réduit la taille des informations entrées, puis les couches suivantes. Le décodage commence ensuite, la représentation de caractéristiques linéaire est apprise et la traille augmente progressivement. À la fin de cette architecture, la taille de sortie est égale à la taille d'entrée.

Une telle architecture est idéale pour préserver la taille initiale. Le problème est qu'elle compresse l'input de façon linéaire, ce qui empêche la transmission de la totalité des caractéristiques.

C'est là que U-NET tire son épingle du jeu grâce à son architecture en U. La déconvolution est effectuée du côté du décodeur, ce qui permet d'éviter le problème de goulot rencontré avec une architecture auto-encodeur et donc d'éviter la perte de caractéristiques.

VGG16-UNET

VGG est un réseau de neurones convolutionnels proposés par K. Simonyan et A. Zisserman de l'université d'Oxford et qui a acquis une notoriété en gagnant la compétition ILSVRC (ImageNet Large Scale Visual Recognition Challenge) en 2014. Le modèle a atteint une précision de 92.7% sur Imagenet (une base de données de plus de 14 millions d'images labellisées réparties dans plus de 1000 classes).

Notre modèle VGG16-UNET a l'architecture d'U-NET mais VGG16 est utilisé pour la partie encodeur

UNET-MINI

Il s’agit d’une version simplifiée d’U-NET qui nous servira de baseline.

Métriques

Sparse Categorical Accuracy

Elle calcule le pourcentage de valeurs prédites (yPred) qui correspondent aux valeurs réelles (yTrue) pour les étiquettes uniques. En d'autres termes "à quelle fréquence les prédictions ont un maximum au même endroit que les vraies valeurs". Elle se calcule en divisant le nombre de pixels prédits avec exactitude par le nombre total de pixels. Comme la précision catégorielle recherche l'indice de la valeur maximale, yPred peut être le logit ou la probabilité des prédictions. Un avantage de cette métrique est qu’il n’y a pas besoin de convertir les masques avec un One Hot Encoding.

Temps d’entraînement

Pour chaque modèle, les temps d’entraînement sont mesurés.

Fonction de perte

Sparse Categorical Crossentropie

Elle quantifie, pour chaque classe, la différence entre deux distributions de probabilités

Dice loss

Elle est définie par :

1 - (2 * intersection / union)

Augmentation de données

L'idée derrière la Data Augmentation est de reproduire les données préexistantes en leur appliquant une transformation aléatoire. Par exemple, appliquer un effet miroir sur une image. Lors de l'entraînement, notre modèle apprendra donc sur beaucoup plus de données tout en ne rencontrant jamais deux fois la même image. Cela a, en théorie, pour effet de réduire l'overfitting.

Les différentes transformations que j'ai appliquées aléatoirement sont :

- rotation
- zoom
- inversion gauche-droite
- changement de luminosité
- effet d'accentuation

Certaines de ces transformations ont été réalisées à l'aide de la librairie imgaug, d'autres ont été implémentées par mes soins.

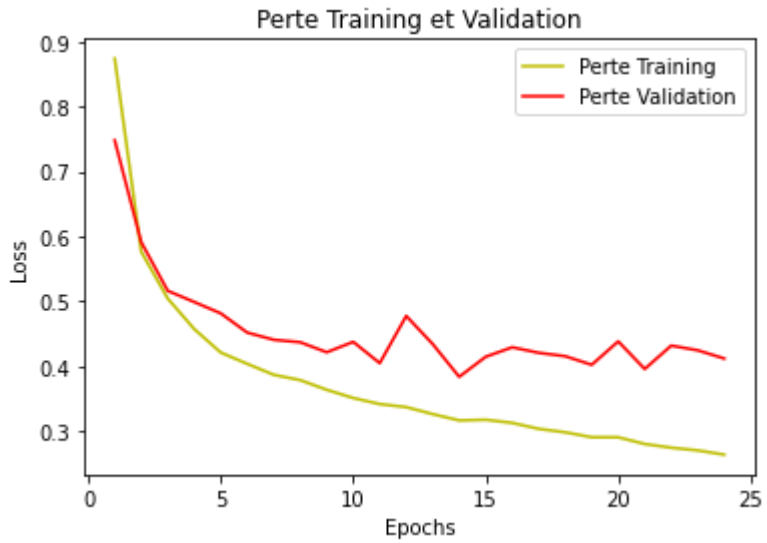
Générateur de données

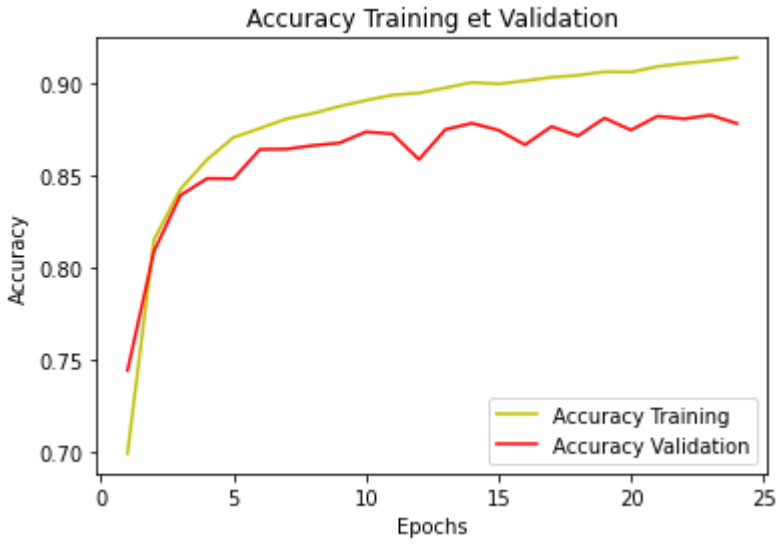
Comme le dataset est volumineux, on ne peut pas le charger en mémoire dans son intégralité. Pour contourner ce problème, j'ai implémenté un générateur de donnée qui se comporte comme un itérateur qui charge les fichiers par paquet (batch).

Résultats

Baseline - U-NET Mini

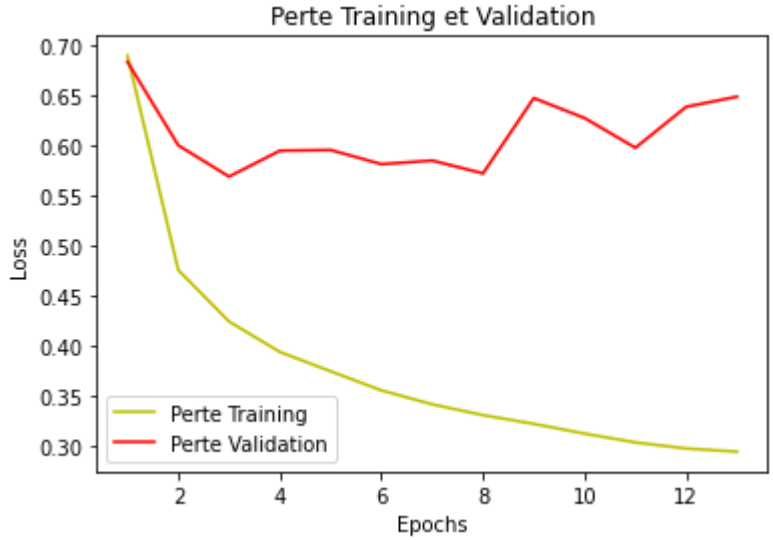
Sans augmentation, loss : Sparse Categorical Accuracy

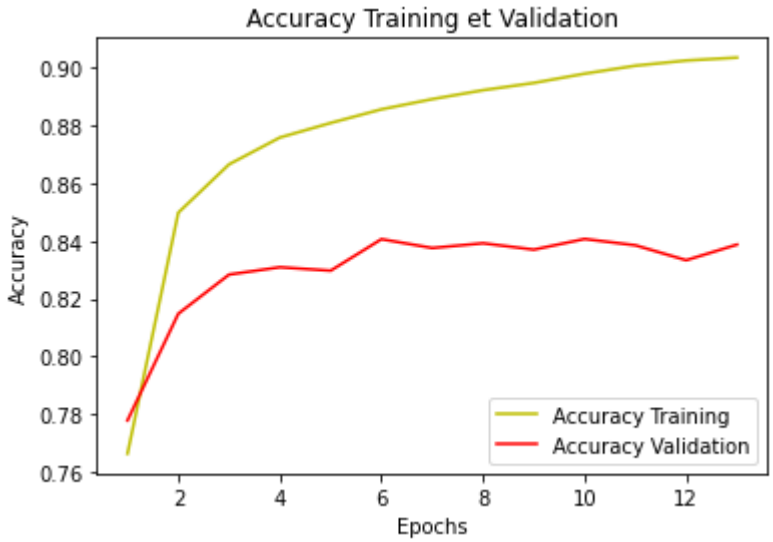




Sans augmentation, loss : Dice Loss

Avec augmentation, loss : Sparse Categorical Accuracy

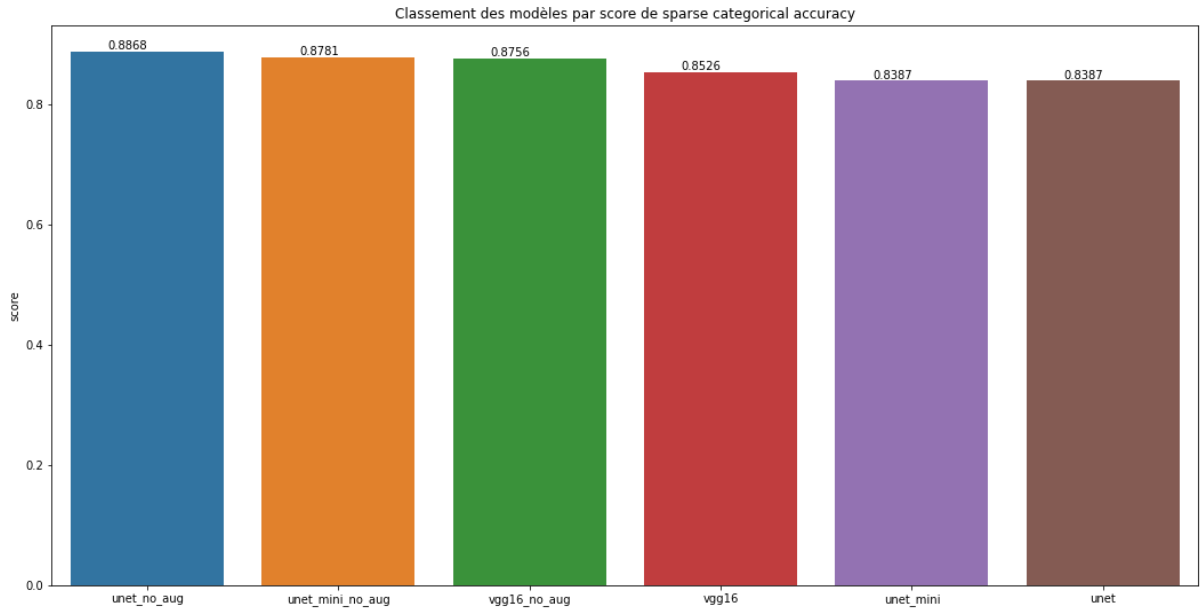




L'entraînement de ces trois modèles nous permet de déterminer que la fonction de perte à privilégier est sparse_categorical_accuracy, et que l'augmentation de données produit de moins bons résultats dans mon cas, ce point pourrait être à améliorer lors des prochaines itérations.

Pour chacun des modèles, j'ai également visualisé quelques exemples de segmentation :

Comparaison des modèles

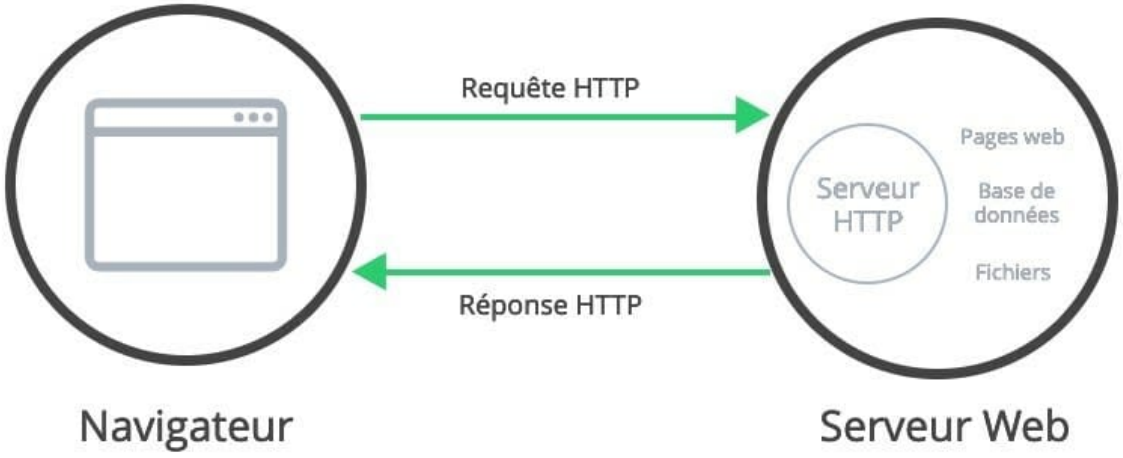


Le modèle qui obtient le meilleur score est le modèle U-NET sans augmentation.

Déploiement du modèle

API

Une API (interface de programmation d'applications) est un programme qui peut être utilisé par d'autres programmes, afin de permettre aux applications de dialoguer entre elles. Une API permet au serveur et au client de communiquer entre eux et d'échanger des informations et des données au format JSON. Les échanges se font sous la forme de requêtes et de réponses HTTP.



J'ai donc créé une API qui reçoit une image et envoie en retour le masque. L'API a été créée à l'aide de FastAPI, uvicorn et Nginx. L'API peut être consommée par n'importe quel service. L'adresse de son endpoint est : http://http://52.56.80.199/api/

Application

Pour faire la démonstration de l'API, j'ai créé une page web qui permet de charger une image avec un glisser/déposer, l'envoie à l'API et affiche la réponse de l'API : le masque et la liste des catégories détectées.

Future Vision Transport

DÉPOSEZ ICI

Future Vision Transport

Résultat de l'analyse :

Objets détectés :

- construction
- flat
- human
- nature
- object
- sky
- vehicle
- void

Hébergement

l'api et l'application web de démonstration ont été déployées sur une instance EC2 (Elastic Compute Cloud) d'AWS. EC2 est un service proposé par Amazon permettant à des tiers de louer des serveurs sur lesquels exécuter leurs propres applications web. EC2 permet un déploiement extensible des applications en fournissant une interface web par laquelle un client peut créer des machines virtuelles, c'est-à-dire des instances du serveur, sur lesquelles le client peut charger n'importe quel logiciel de son choix. Un client peut créer, lancer, et arrêter des instances de serveurs en fonction de ses besoins, et paye en fonction du temps d'usage des serveurs, d'où le terme d'« Élastique » (Elastic en anglais). Un client peut mettre en place des instances de serveurs isolées physiquement (qui ne s'exécutent pas sur le même serveur physique) les unes des autres, de telle façon qu'en cas de panne, il soit possible de restaurer les instances défaillantes et d'assurer la continuité du service.