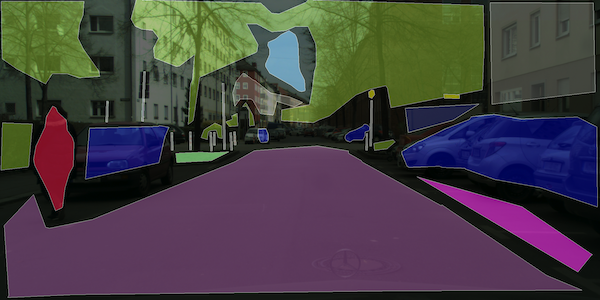




*Note technique*

[](https://user.oc-static.com/upload/2019/10/24/15719062195074_nuremberg01.png)

Dhaker KACEM, mai 2024.

***Table des matières***

I) **Présentation du projet**

II) **Présentation des données**

1. *Récupération des masques*
2. *Distribution des catégories dans les masques*

III) **Datagenrator et métrique personnalisée**

*1)Datagenrator*

*2) Personnalisation de la métrique*

IV) **Entrainement des modèles**

1. *Présentation du modèle U-Net Mini*
2. *Présentation du modèle VGG16*
3. *Synthèse des résultats*

V) **Déploiement du modèle retenu**

*1) Création de l’API Flask*

*2) Création de l’application Streamlit pour la visualisation des prédictions*

*3) Déploiement sur Azure Web App*

*4) Consommation de l’API*

VI) **Pistes d’amélioration**

1)

2)

**I) Présentation du projet**

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

Je suis ingénieurs IA au sein de l’équipe R&D de cette entreprise. Mon 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 :

1) Acquisition des images en temps réel

2) Traitement des images

3) Segmentation des images (c’est notre mission)

4) Système de décision

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

Dans cette première partie, notre mission sera, donc, d'entrainer plusieurs modèles de machine learning et de retenir celui qui produit les meilleures performances (temps de calcul et score IOU (Intersection Over Union).

Cette première partie consistera, donc, à faire de la classification sémantique.

La deuxième partie du projet consistera en le déploiement de ce modèle sur le cloud (Azure) à travers la conception d'une API.

Et enfin, la troisième partie sera consacrée à la conception d'une application web de représentation des résultats. Pour se faire, nous utiliserons Flask et Streamlit.

Grace à ce modèle de classification sémantique, notre véhicule autonome, devrait, à terme, pouvoir, à partir de photos prises par une caméra embarquée, identifier toutes les catégories présentes à un instant t et prendre ses décisions en conséquence.

II) **Présentation des données**

Les données utilisées dans ce projet sont fournies par CityScapes qui fournit un jeu de données de 3000 images d’une caméra embarquée. Ces images sont fournies avec leurs masques annotés des différentes catégories.

En tout, nos images sont annotées de 8 catégories principales et de 32 sous catégories.

Pour les besoins du projet nous n’aurons besoin que des 8 catégories principales qui sont : vide, route, construction, objets, nature, ciel, humain et véhicule. Par conséquent, nous opérant une petite manipulation qui vise à convertir nos masques à 34 canaux en des masques à 8 canaux uniquement où chaque canal représente une catégorie. Cette manipulation retourne, ensuite, l’indice du canal avec la valeur maximale pour chaque pixel, ceci donne la catégorie principale pour chaque pixel.

Le jeu de données comprend deux dossiers, un dossier contenant les images brutes provenant d’une caméra embarquée et un autre contenant les masques de la segmentation sémantique.

Chaque dossier contient 3 sous dossiers qui sont : test, train et val qui ne sont autres que les données de test, d’entrainement et de validation.

Le dossier « leftImg8bit » contient des images en couleur RGB en format « png » de dimension 2048 x 1024. Ce fichier comprend 5000 images

Le dossier « gtFine » contient, quant à lui, les masques. En tout à chaque image correspond 4 fichiers de masques :

* Un fichier « \_gtFine\_color.pn » au format « png ». Cette image représente une vue segmentée d’une scène de rue. Les différentes couleurs indiquent diverses classes dans l’image. Ces couleurs sont utilisées pour des tâches de segmentation sémantique.
* Un fichier Json nommé « \_getFine\_polygons.json », ce fichier JSO-N