Ingénierie IA

Kindi BALDE

Participez à la conception d'une voiture autonome

Présentation du projet

Problématique métier

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

Elle traite toutes les parties de la chaîne de production du système embarqué de vision par ordinateur.

1. acquisition des images en temps réel
2. traitement des images
3. **segmentation des images**
4. système de décision

Challenge

Adaptation aux contraintes de la chaîne de production

- Acquisition des données de traitement des images
- Livraison d'une API pour nourrir le système de décision

Mission

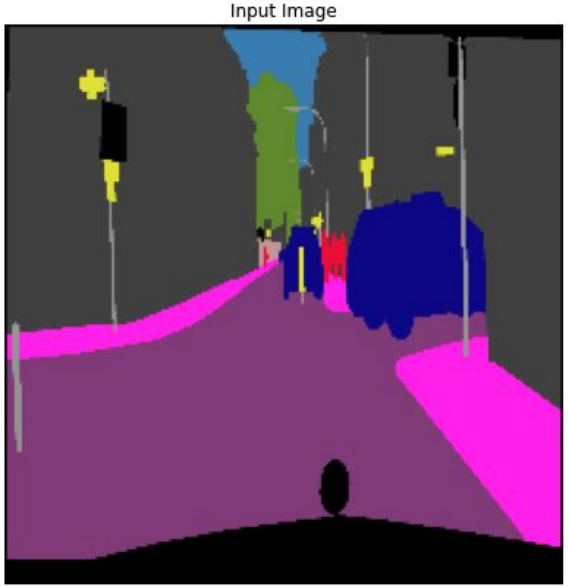
En tant qu' ingénieurs IA au sein de l'équipe R&D, notre rôle est de **concevoir un premier modèle de segmentation d'images** qui devra s'intégrer facilement dans la chaîne complète du système embarqué.

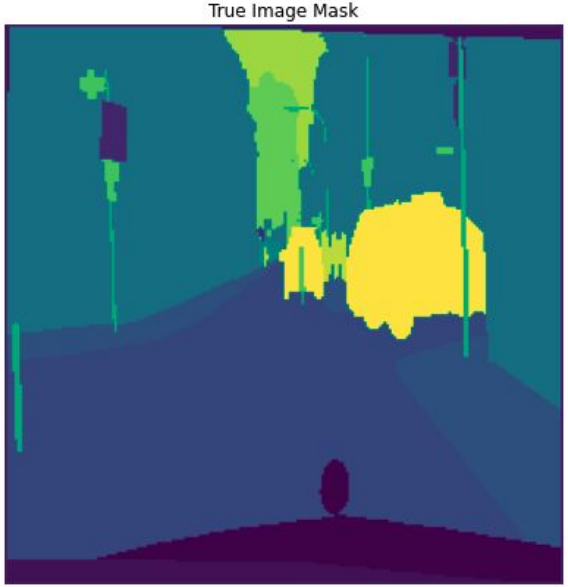
Sommaire

1. Préparation des données
2. Conception des modèles
3. Entraînement des modèles sur Azure
4. Déploiement de l’API Flask en production
5. Démo fonctionnelle de la Web Application

1. Préparation des données

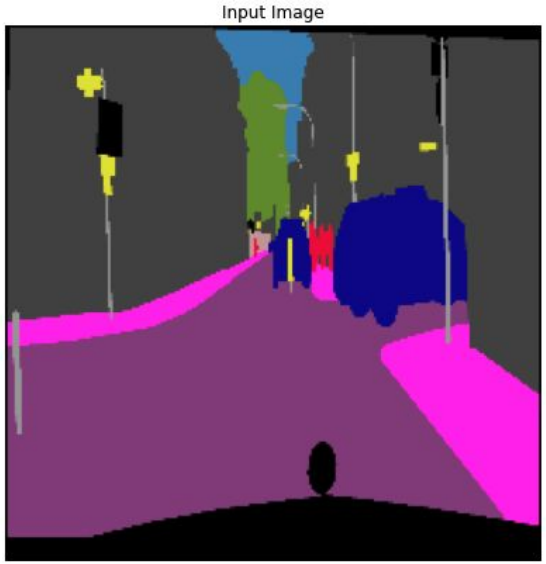
Nettoyage des données : création d'une arborescence image et mask pour chaque type de dataset (train, test, validation); suppression du quatrième channel des images inputs (pour s'adapter aux poids du VGG16)

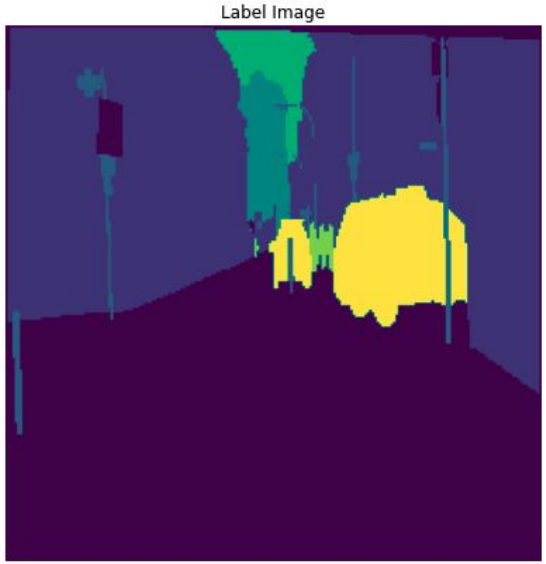




1. Préparation des données (2 / 3)

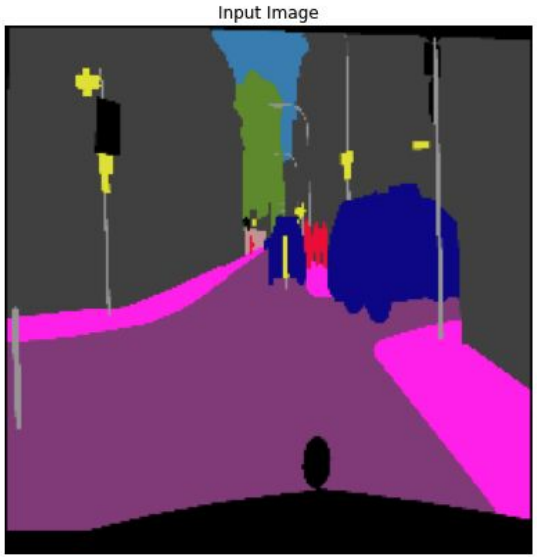
Génération des nouveaux segments : la création des nouveaux segments de 0 à 7 au lieu de 0 à 33. Cette transformation est faite uniquement sur les données mask label.

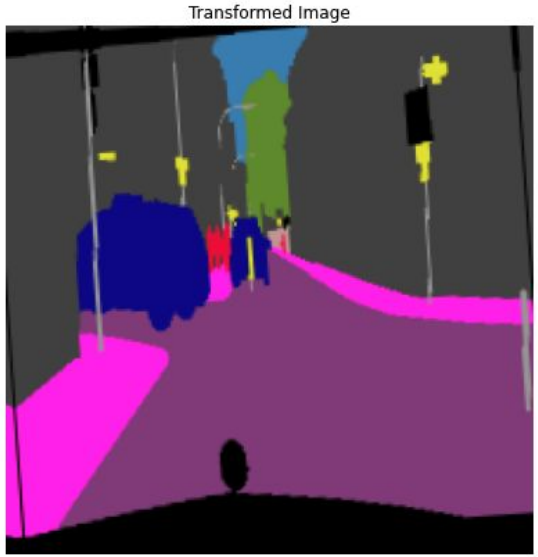




1. Préparation des données

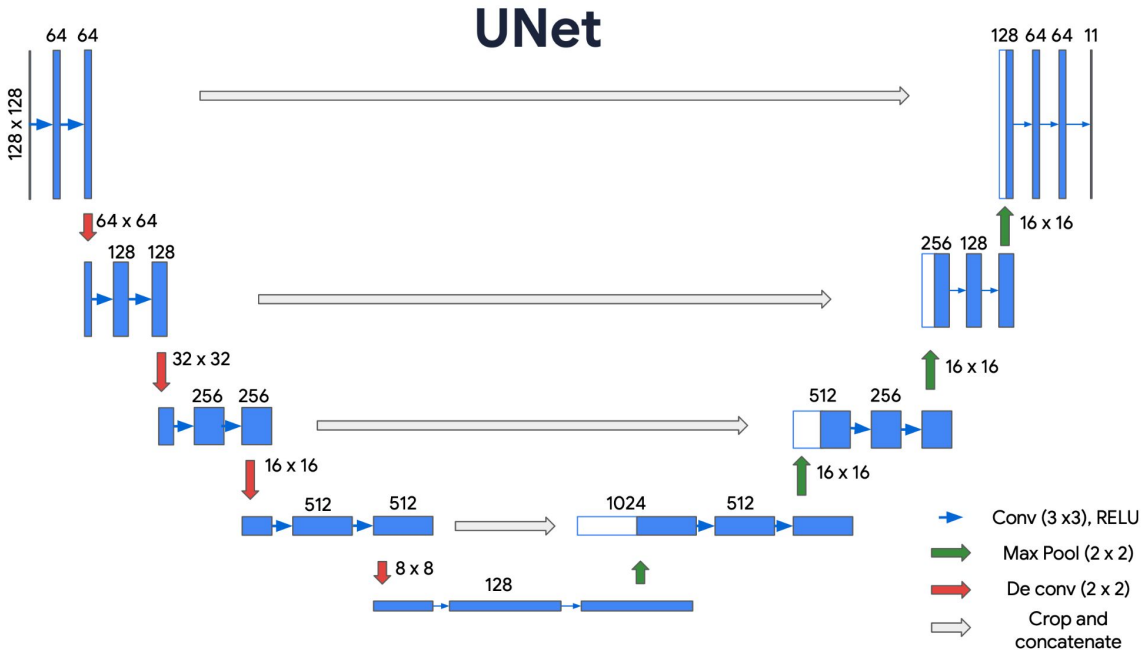
Augmentation des données : utilisation de la librairie **ALBUMENTION**. Les transformations sont : un flip horizontal, une petite rotation de 10 degrés, une baisse du contrast des images.





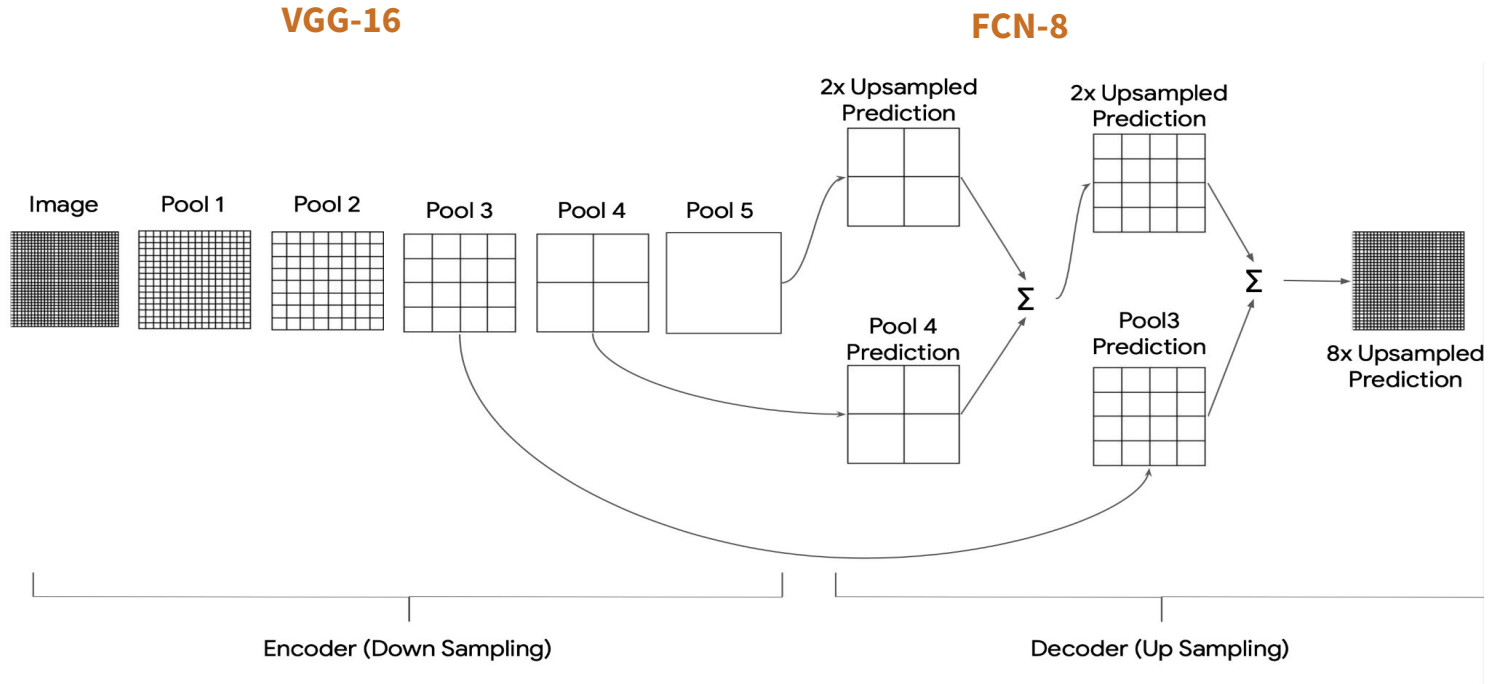
2. Conception des modèles (1 / 2)

Deux types de modèle d'image ***Semantic*** Segmentation : **U-Net** et **FCN8-VGG16**



2. Conception des modèles

Deux types de modèle d'image ***Semantic*** Segmentation : **U-Net** et **FCN8-VGG16**

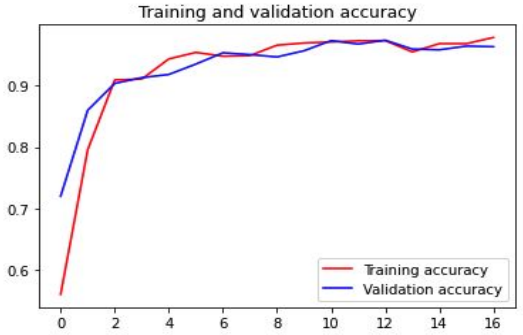


4. Entraînement des modèles sur Azure (1 / 2)

```
model_history = unet.fit(train_data_unet1,
          epochs=EPOCHS,
          validation_data=val_data_unet1,
          steps_per_epoch=steps_per_epoch,
          validation_steps=validation_steps,
          verbose=2,
          callbacks=[EarlyStopping(
              patience=7,
              min_delta=0.05,
              baseline=0.8,
              mode='min',
              monitor='val_loss',
              restore_best_weights=True,
              verbose=1)
          ])
```

```
Restoring model weights from the end of the best epoch: 10.
20/20 - 2s - loss: 0.0769 - accuracy: 0.9785 - val_loss: 0.1019 - val_accuracy:
0.9634 - 2s/epoch - 121ms/step
Epoch 00017: early stopping
```

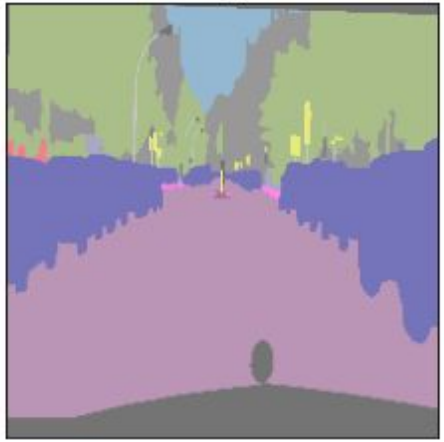
U-Net V2



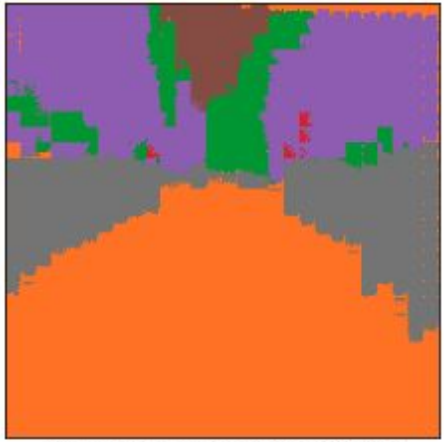
4. Entraînement des modèles sur Azure (2 / 2)

FCN_VGG16

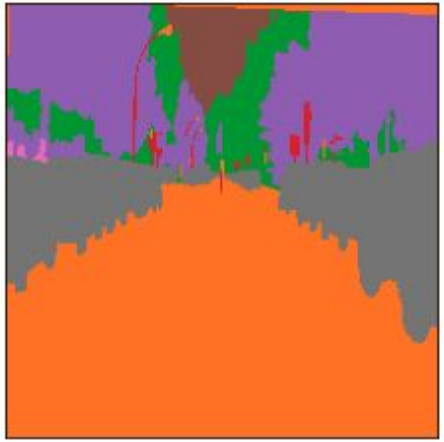
Image



Predicted Mask



True Mask



flat: IOU: 0.9538 Dice Score: 0.9764

vehicle: IOU: 0.8873 Dice Score: 0.9403

nature: IOU: 0.8642 Dice Score: 0.9271

sky: IOU: 0.8131 Dice Score: 0.8969

construction: IOU: 0.6645 Dice Score: 0.7984

object: IOU: 0.1145 Dice Score: 0.2055

5. Déploiement de l'API Flask en production

Déploiement du modèle U-Net sur Azure

1. Création d'un *Experiment*
2. Enregistrement du modèle **U-Net** dans *Experiment*
3. Création des configurations d'un *container*
4. Création d'un fichier python *score.py*
5. Création d'un fichier *YML* de liste des librairies nécessaire au modèle
6. Lancement du déploiement avec la method ***deploy*** de la librairie *Model*

6. Une démo fonctionnelle de la Web Application Flask sur Azure

https://openclassroomsflaskapi.azurewebsites.net/submit

Input sample image IDs to predict

frankfurt_000000_001016_gtFine_color
frankfurt_000000_000576_gtFine_color
frankfurt_000000_000294_gtFine_color