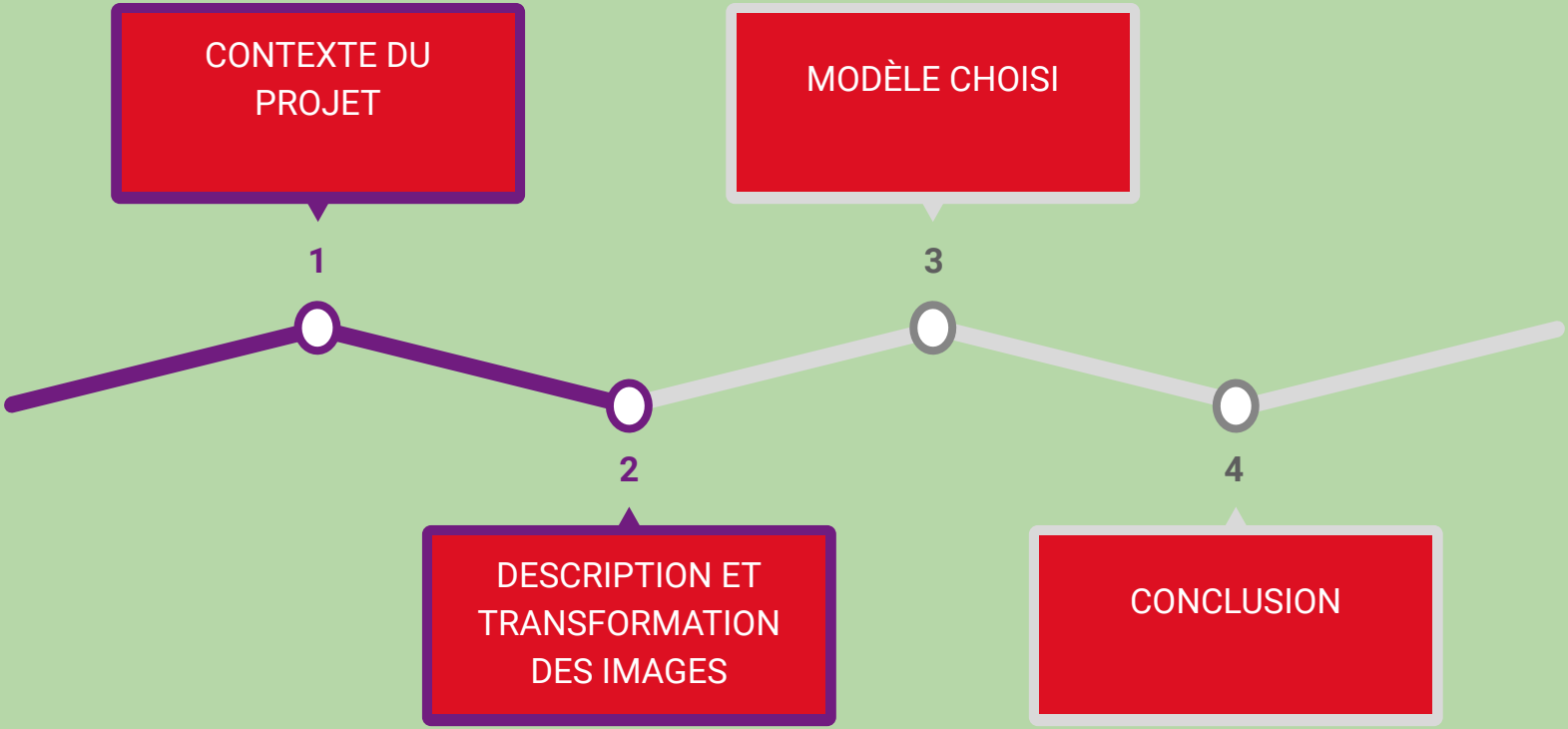
Participez à la conception à la voiture autonome

Moussa KIBALY

OPENCLASSROOMS

SOMMAIRE


CONTEXTE DU PROJET
1
DESCRIPTION ET TRANSFORMATION DES IMAGES
2
MODÈLE CHOISI
3
CONCLUSION
4

1. CONTEXTE DU PROJET

La société Future Vision Transport conçoit des systèmes embarqués de vision par ordinateur pour les véhicules autonomes. L ’ équipe R&D responsable de la segmentation des images doit concevoir et déployer un modèle de segmentation sémantique des images acquis en temps réel, basé sur des réseaux de neurones profonds, à l’aide du studio **Azure ML de Microsoft**.

2. DESCRIPTION ET TRANSFORMATION DES IMAGES

On part d'un jeu de données Cityscape composé de 2975 images d'entraînement et les labels associés, 500 images et labels de validation, 1525 images et labels de test.

La dimension des images est 1024X2048.

Avant d'alimenter notre modèle de segmentation, les images et les labels associés vont être traités par un changement de dimension et par des transformations (rotation, translation, ...) pour augmenter le jeu de données

Image du jeu d’entraînement avec le label associé

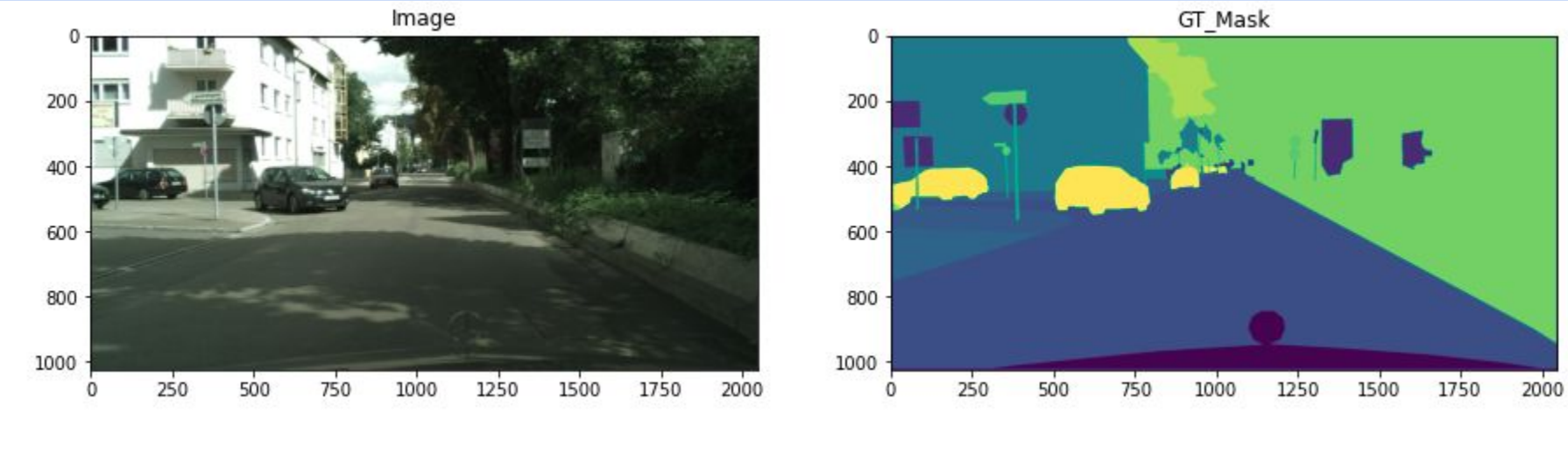


Image et son mask segmenté à droite. Ils vont servir à entraîner le modèle de segmentation sémantique.

PRÉ-TRAITEMENT DES IMAGES ET MASKS

Les dimensions des images sont redéfinies en 256X256 et normalisées.

L'augmentation d'image est effectuée afin d'avoir un jeu de données plus important et plus varié ce qui va améliorer les performances du modèle.

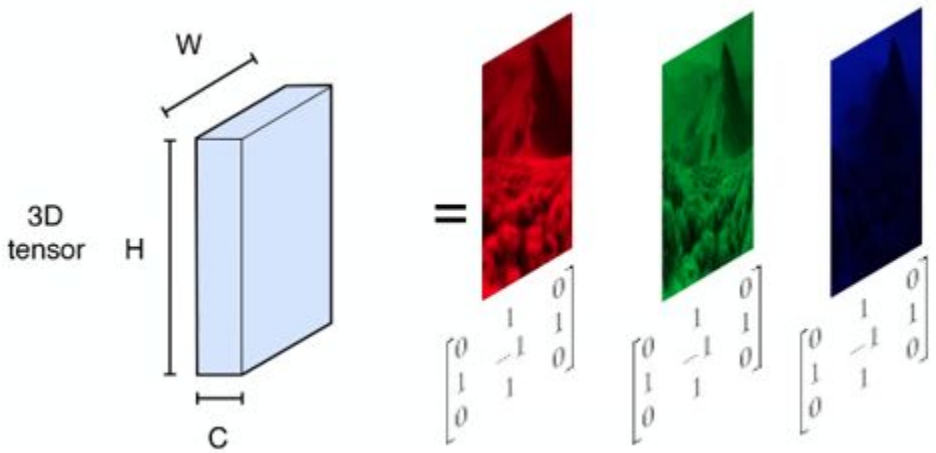
Cela passe par des transformations telles que la rotation, le zoom, la translation horizontale ou verticale, le filtrage des images, ...

L'entraînement du modèle est facilité par l'utilisation des générateurs de données qui permettent de charger les images à la volée étant donné que la taille de ces données est importante.

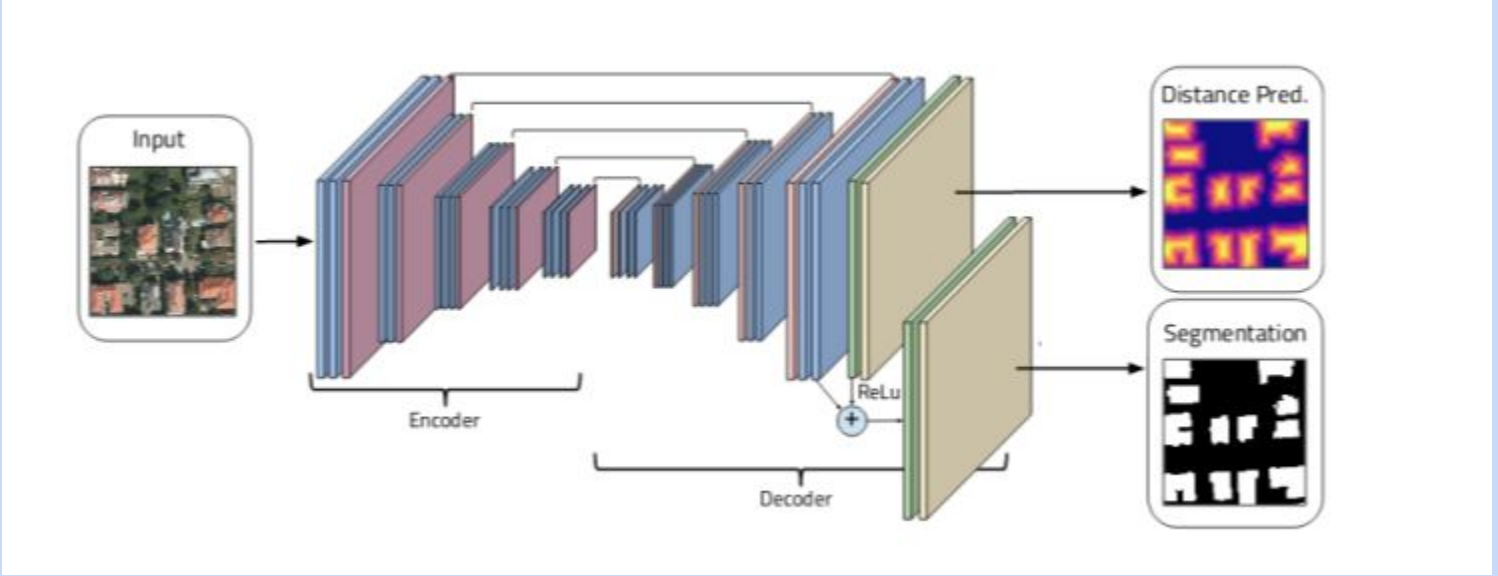
Transformation en tenseurs

Les images en couleur sont transformées en tenseurs de dimension 256X256X3. Les masks associés sont convertis en tenseurs 256X256X8. La dimension 8 vient des 8 classes à définir pour chaque pixel d'une image.

Les classes sont les suivantes : void, flat, construction, object, nature, sky, human, vehicle.



3. MODÈLE DE SEGMENTATION SÉMANTIQUE



Architecture encodeur-décodeur du modèle Unet

L' encodeur est composé d'empilement de couches complexes contenant chacune 2 couches de convolution 2D utilisant la fonction d'activation ReLu suivies par un max pooling. Le principe de la partie encodeur est de produire des tenseurs de basse dimension contenant toutes les informations spatiales de l'image.

Le décodeur permet de récupérer les tenseurs de basse dimension afin de produire en sortie des cartes de segmentation (segmentation sémantique). L'augmentation de dimension des tenseurs se fait par une couche Upsampling combinée avec une couche de convolution + ReLu et de max pooling. Des sauts de connexion sont faites entre les sorties des couches intermédiaires de convolution de l'encodeur et l'entrée de ces mêmes couches du décodeur.

Ces sauts de connexion permettent de remédier à la perte d'information le long des couches empilées.

Représentation du modèle Unet

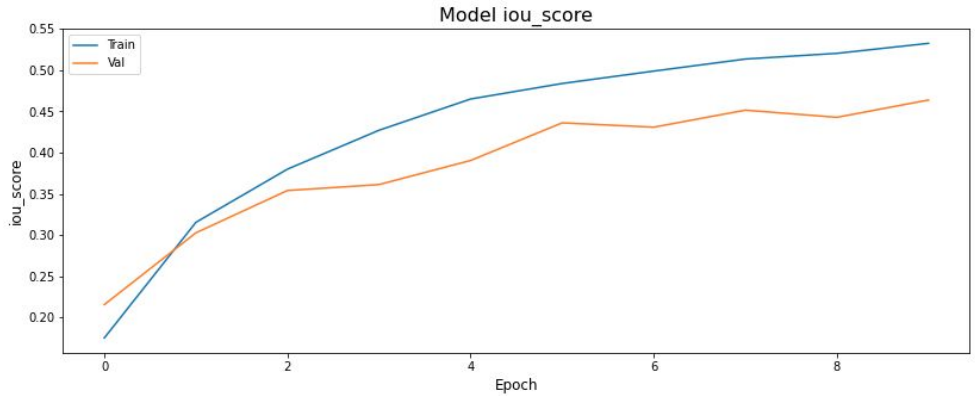
```
# DOWNSAMPLING LAYERS
conv_1 = Conv2D(32, (3, 3), activation='relu', padding='same', kernel_initializer = 'he_normal')(input_0)
conv_1 = Dropout(0.2)(conv_1)
conv_1 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv_1)
pool_1 = MaxPooling2D((2, 2))(conv_1)

conv_2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(pool_1)
conv_2 = Dropout(0.2)(conv_2)
conv_2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv_2)
pool_2 = MaxPooling2D((2, 2))(conv_2)

conv_3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(pool_2)
conv_3 = Dropout(0.2)(conv_3)
conv_3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv_3)

# UPSAMPLING LAYERS
up_1 = concatenate([UpSampling2D((2, 2))(conv_3), conv_2], axis=-1)
conv_4 = Conv2D(32, (3, 3), activation='relu', padding='same')(up_1)
conv_4 = Dropout(0.2)(conv_4)
conv_4 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv_4)

up_2 = concatenate([UpSampling2D((2, 2))(conv_4), conv_1], axis=-1)
conv_5 = Conv2D(32, (3, 3), activation='relu', padding='same')(up_2)
conv_5 = Dropout(0.2)(conv_5)
conv_5 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv_5)

segm_X = Conv2D(INPUT_CLASS_NB, (1, 1), activation='softmax', name='seg')(conv_5)

unet_model = Model(inputs=[input_0], outputs=[segm_X])
```

La fonction Relu est appliqué à chaque couche de convolution 2D qui contient 32 filtres chacune.

La fonction d'activation Softmax permet de générer les prédictions des pixels de chaque image en 8 classes.

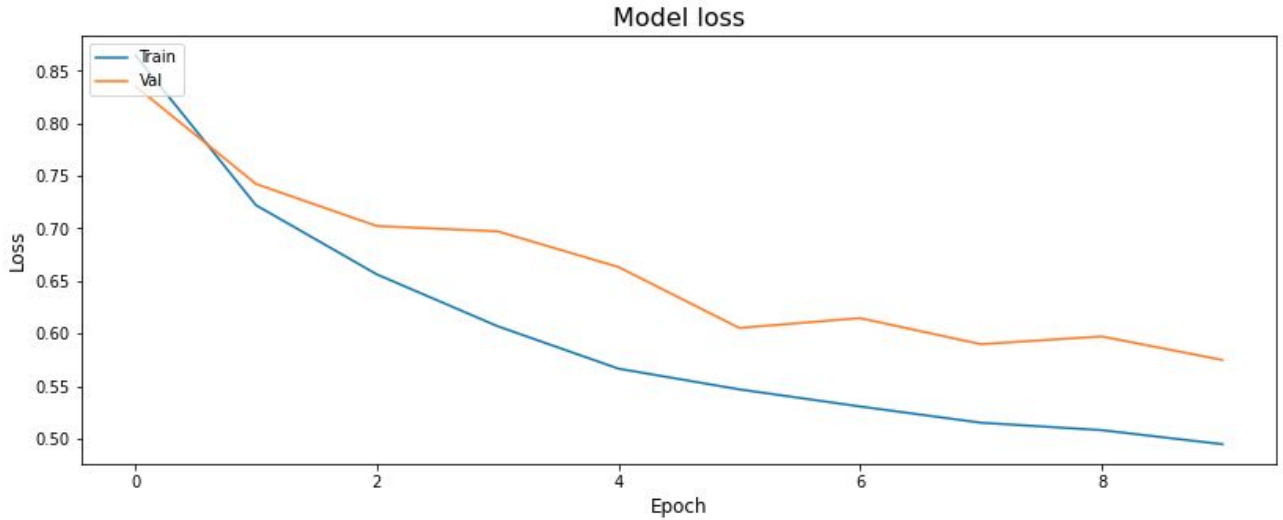
Performance du modèle

IOU_SCORE de 53.24%

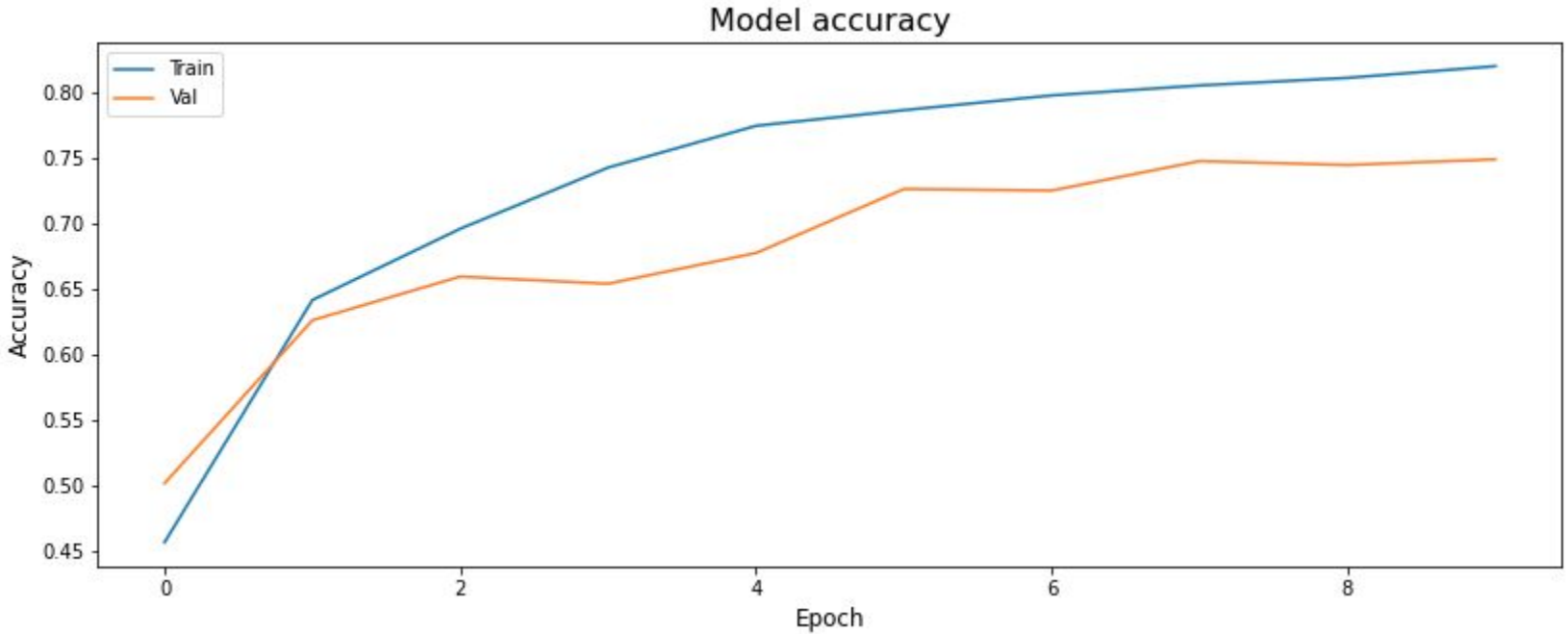


On sauvegarde le modèle qui a le meilleur score de IoU de validation. Le score Intersection over Union sert à mesurer l'adéquation entre l'image segmentée et l'image réelle, elle est plus adaptée pour la segmentation sémantique que ne peut l'être la précision (accuracy).

Fonction de perte de 49.46%



Accuracy de 81.99%



L'analyse de la performance par le biaise de l' **accuracy** et la fonction de perte (**loss**) nous montre une augmentation de la première et une diminution de la seconde en fonction des itérations (**epoch**). De même pour l'index de Jacquard qui augmente au fil des itérations.

Le modèle s'améliore

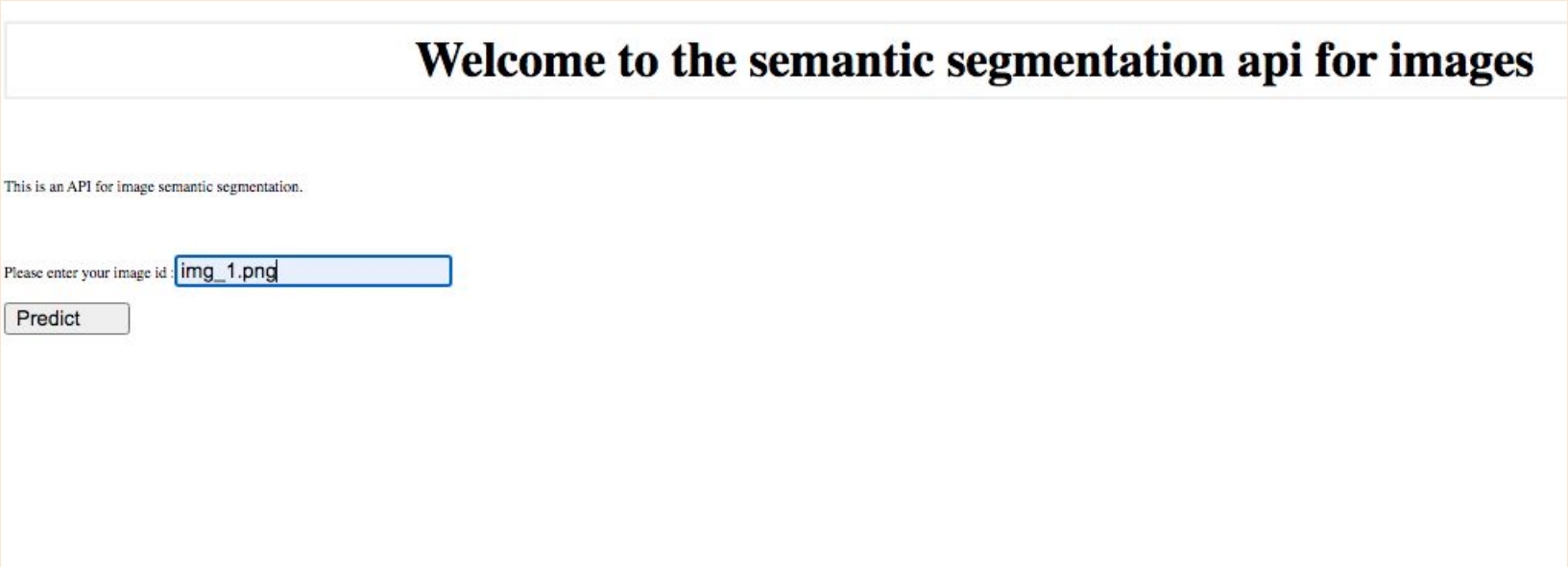
Augmentation d’images

Les métriques de performance du modèle ont pu être amélioré par le biais de la technique d'augmentation d'images en utilisant des packages tels que Albumentations ou Imgaug.

Cela consiste à générer de nouvelles images sur la base des images d'origine en effectuant des rotations, des zooms, des translations ou applications de filtres et autres techniques d'image.

4. CONCLUSION

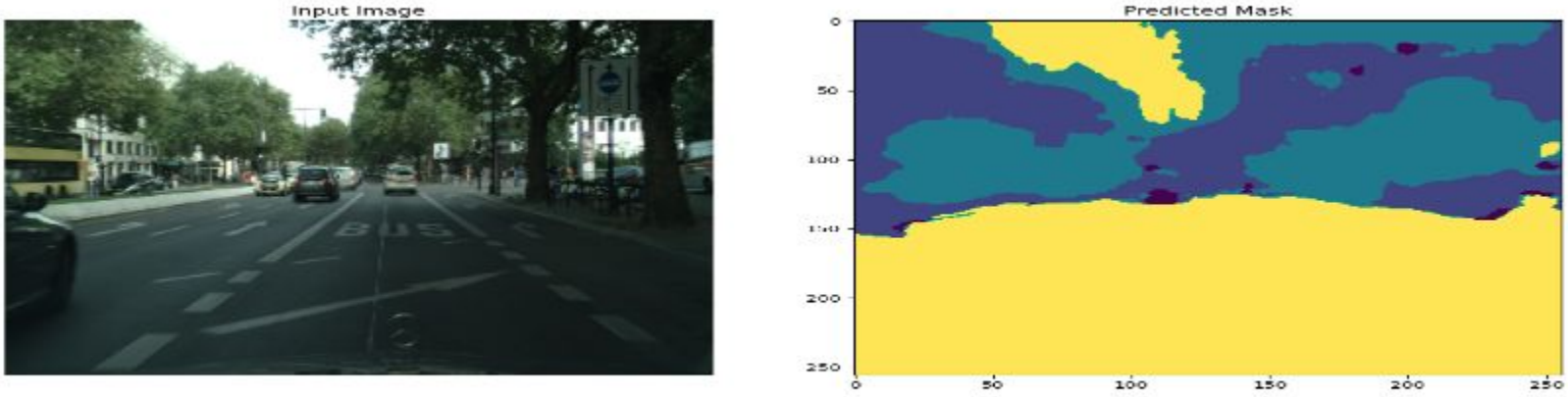
Modèle fonctionnel de segmentation sémantique



Le modèle Unet implémenté et déployé dans Azure Machine Learning est appelé à travers une application web Azure Web App utilisant le framework **Flask.**

Ainsi, nous pouvons déterminer la segmentation sémantique d'une image via une API Flask puis en cliquant sur **Prédire**.

Résultat de la segmentation sémantique



L'application est accessible via l'url **https://segapp.azurewebsites.net**