

# PROJET 8

**PARTICIPEZ À LA CONCEPTION D'UNE VOITURE AUTONOME  
EN CONCEVANT UN MODÈLE DE SEGMENTATION D'IMAGES**

**#DEEP LEARNING #SEMANTIC SEGMENTATION #DATA GENERATOR  
#AUGMENTATION #DEPLOIEMENT  
#LOSS FUNCTION #IOU #DICE COEFFICIENT #VGG UNET # PSP UNET  
#AZURE ML #KERAS #FLASK #AZURE WEB SERVICES**

## Ingénieur IA

Développez et intégrez des algorithmes de Deep Learning au sein d'un produit IA

**OPENCLASSROOMS**

**OUDDANE NABIL**



# SOMMAIRE

## Projet 8

### Participez à la conception d'une voiture autonome

#### A. INTRODUCTION

1. Contexte
2. Objectifs
3. Ressources complémentaires

#### B. Image Segmentation - Pixel classification :

Modèles simples : Random Forest classifieur et SVC

#### C. Image Segmentation - Pixel classification :

Modèles de DEEP LEARNING

#### D. Déploiement d'une api flask de prediction de segmentation d'images

basée sur VGG-UNET via azure web app

#### E. Conclusion et auto évaluation



# INTRODUCTION

# 1. Contexte

- ENJEU global:
  - **Future Vision Transport** conçoit des systèmes embarqués de vision par ordinateur pour les véhicules autonomes
  - **4 Piliers:**
    - ACQUISITION IMAGE TEMPS REEL
    - TRAITEMENT IMAGES
    - SEGMENTATION DES IMAGES
    - SYSTÈME DE DECISION
- ENJEU Compétences DU P8:
  - **concevoir un premier modèle de segmentation d'images**
  - vision / deep learning / Azure
- **DONNEES SOURCES**
  - images segmentées et annotées de caméras embarquées
    - <https://www.cityscapes-dataset.com/dataset-overview/>
    - [https://s3-eu-west-1.amazonaws.com/static-oc-static.com/prod/courses/files/AI+Engineer/Project+8+-+Participez+%C3%A0+la+conception+d'une+voiture+autonome/P8\\_Cityscapes\\_gtFine\\_trainvaltest.zip](https://s3-eu-west-1.amazonaws.com/static-oc-static.com/prod/courses/files/AI+Engineer/Project+8+-+Participez+%C3%A0+la+conception+d'une+voiture+autonome/P8_Cityscapes_gtFine_trainvaltest.zip)
    - [https://s3-eu-west-1.amazonaws.com/static-oc-static.com/prod/courses/files/AI+Engineer/Project+8+-+Participez+%C3%A0+la+conception+d'une+voiture+autonome/P8\\_Cityscapes\\_leftImg8bit\\_trainvaltest.zip](https://s3-eu-west-1.amazonaws.com/static-oc-static.com/prod/courses/files/AI+Engineer/Project+8+-+Participez+%C3%A0+la+conception+d'une+voiture+autonome/P8_Cityscapes_leftImg8bit_trainvaltest.zip)

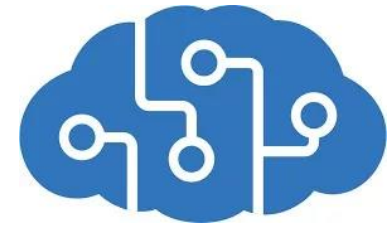


## 2. Objectifs

- **SCRIPT:**
  - **entraîner et déployer un modèle de segmentation** des images:
    - sur les 8 catégories principales
    - en utilisant **Azure Machine Learning**
    - **En utilisant KERAS**
  - déployer une **API Flask** grâce au **service azure qui** recevra en entrée l'identifiant d'une image et retournera l'image avec les segments identifiés par votre modèle et l'image avec les segments identifiés annotés dans le jeu de données
  - Une **note technique** de 10 pages environ contenant:
    - une présentation des différentes approches
    - une synthèse de l'état de l'art
    - la présentation plus détaillée du modèle et de l'architecture retenue
    - une synthèse des résultats obtenus (incluant les gains obtenus avec les approches d'augmentation des données)
    - une conclusion avec des pistes d'amélioration envisageables



Azure Machine Learning



Keras

# 3. Ressources complémentaires

- **Data generation et data augmentation**
  - Exemple de données cityscape et data generator: [https://github.com/srihari-humbarwadi/cityscapes-segmentation-with-Unet/blob/master/batch\\_training.py](https://github.com/srihari-humbarwadi/cityscapes-segmentation-with-Unet/blob/master/batch_training.py)
  - Principes data generator: <https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly>
  - Tuto data generator: <https://medium.com/datadriveninvestor/keras-training-on-large-datasets-3e9d9dbc09d4>
  - Data generator avec data augmentation: <https://github.com/Golbstein/Keras-segmentation-deeplab-v3.1/blob/e3f0daaa79a729c022da658fc86eef82a6c7ceeb/utils.py#L411>
  - Lib d'augmentation d'images: alumentations : <https://alumentations.ai/docs/examples/tensorflow-example/>
  - Lib d'augmentation d'images: imgaug: <https://github.com/aleju/imgaug>
- **Segmentation d'images**
  - Principes segmentation d'images: <https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html>
  - Exemples: <https://github.com/divamgupta/image-segmentation-keras>
  - Fonction loss pour la segmentation d'image: <https://lars76.github.io/2018/09/27/loss-functions-for-segmentation.html>

# 3. Ressources complémentaires

- **VRAC et AZURE**
  - Principes DNN et CNN: <https://docs.microsoft.com/fr-fr/learn/modules/train-evaluate-deep-learn-models/>
  - Gestion données azure: <https://docs.microsoft.com/fr-fr/learn/modules/work-with-data-in-aml/>
  - Entraînement à distance azure: <https://docs.microsoft.com/fr-fr/azure/machine-learning/tutorial-train-models-with-aml>
  - Authentification azure à partir de flask: <https://github.com/Azure/MachineLearningNotebooks/blob/master/how-to-use-azureml/manage-azureml-service/authentication-in-azureml/authentication-in-azureml.ipynb>
  - Serialization d'images avec json: <https://stackoverflow.com/questions/30698004/how-can-i-serialize-a-numpy-array-while-preserving-matrix-dimensions>
  - Azure webapp: <https://docs.microsoft.com/fr-fr/azure/app-service/>
  - Creation et deployment azure webapp: <https://docs.microsoft.com/fr-fr/learn/modules/host-a-web-app-with-azure-app-service/>
  - Mode de deployment de webapp: <https://docs.microsoft.com/fr-fr/learn/modules/host-a-web-app-with-azure-app-service/6-deploying-code-to-app-service>
  - Creation de webapp sur le portail: <https://docs.microsoft.com/fr-fr/learn/modules/host-a-web-app-with-azure-app-service/2-create-a-web-app-in-the-azure-portal>
  - Deploiement automatisé de webapp via github: <https://docs.microsoft.com/fr-fr/azure/app-service/deploy-continuous-deployment?tabs=github>
  - Creation et deployment de webapp en ligne de commande: <https://docs.microsoft.com/fr-fr/azure/app-service/quickstart-python?tabs=bash&pivots=python-framework-flask>
  - <https://docs.microsoft.com/fr-fr/azure/app-service/quickstart-python?tabs=bash&pivots=python-framework-flask>
- **Cours**
  - Cours flask: <https://openclassrooms.com/fr/courses/4425066-concevez-un-site-avec-flask>

**B**

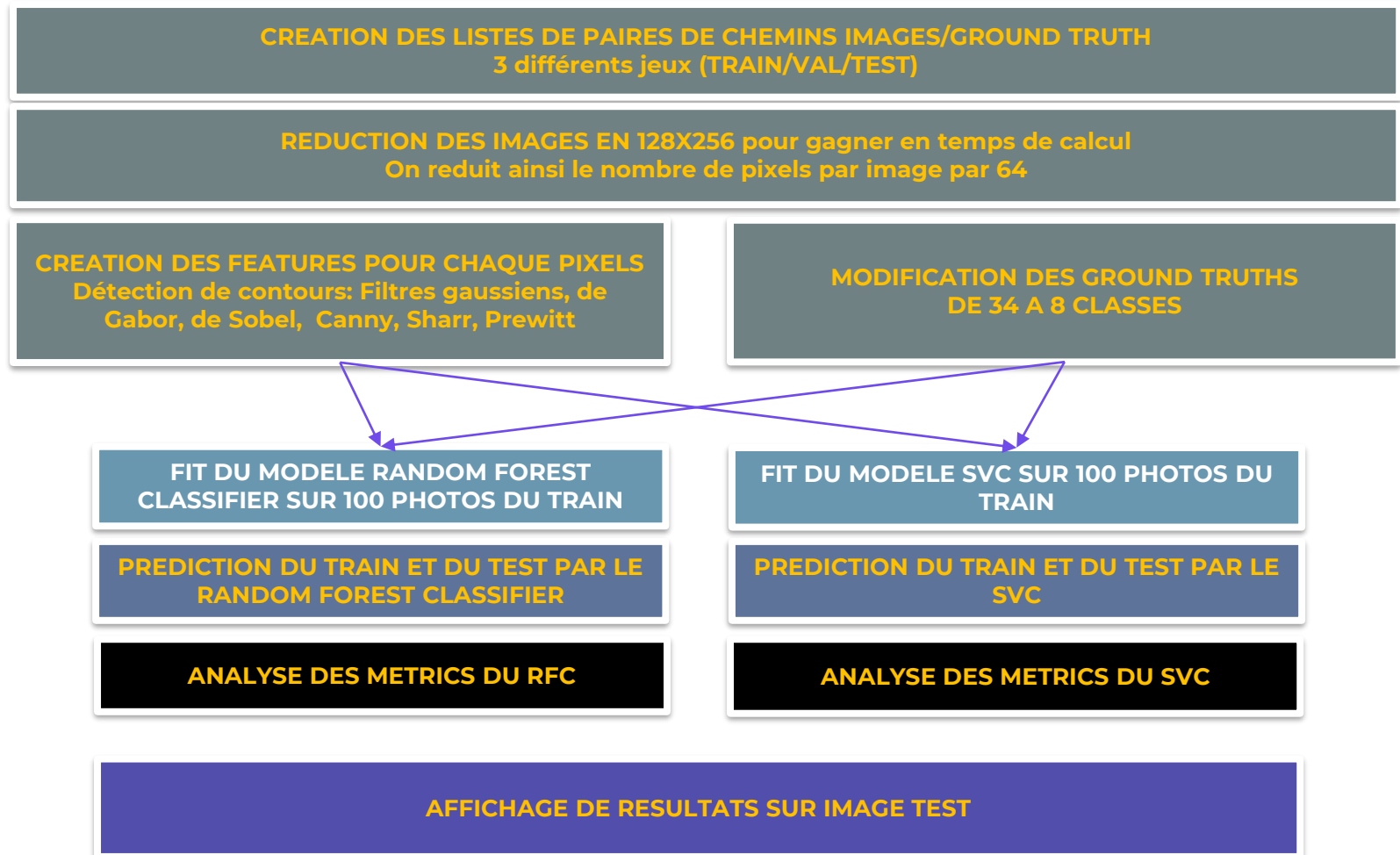
# **IMAGE SEGMENTATION - PIXEL CLASSIFICATION :**

## **MODÈLES SIMPLES : RANDOM FOREST CLASSIFIER ET SVC**

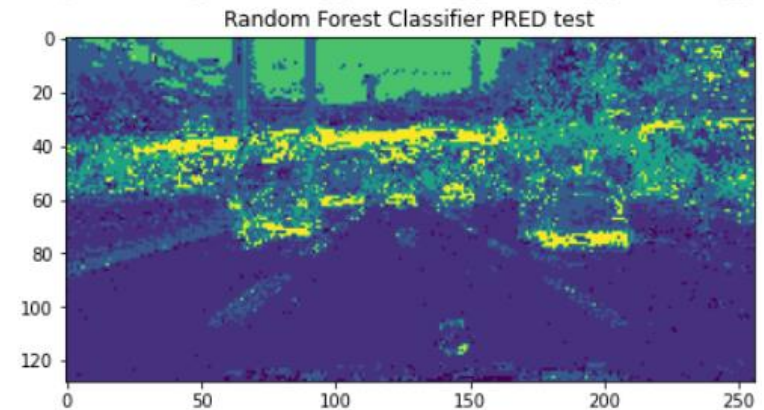
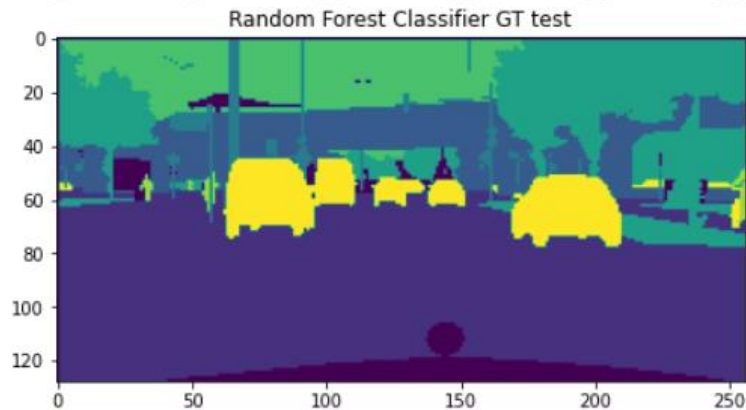


# A-Modèles Simples de Segmentation d'images

## WorkFlow

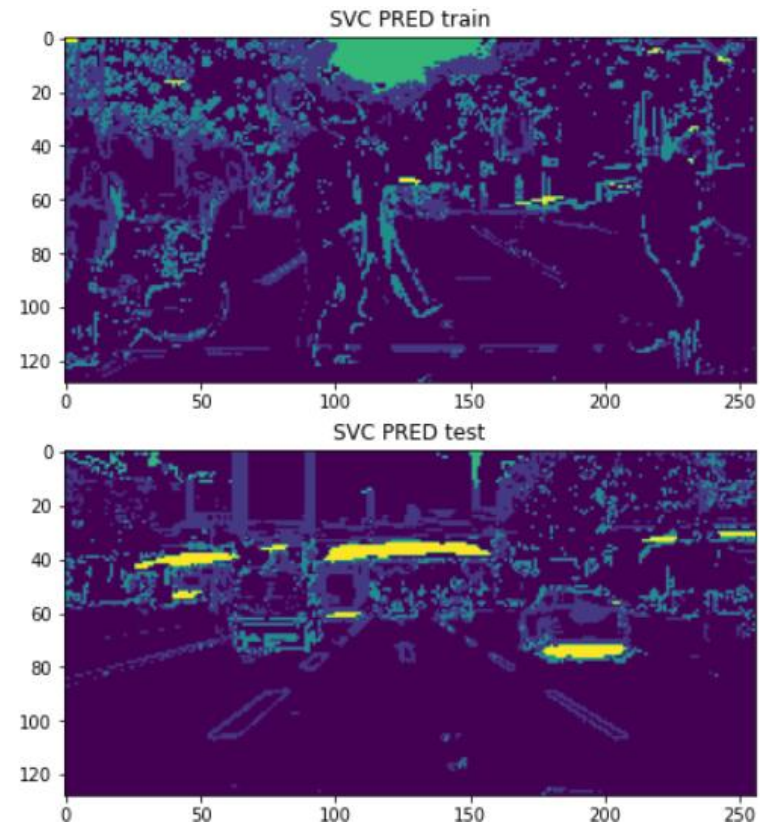
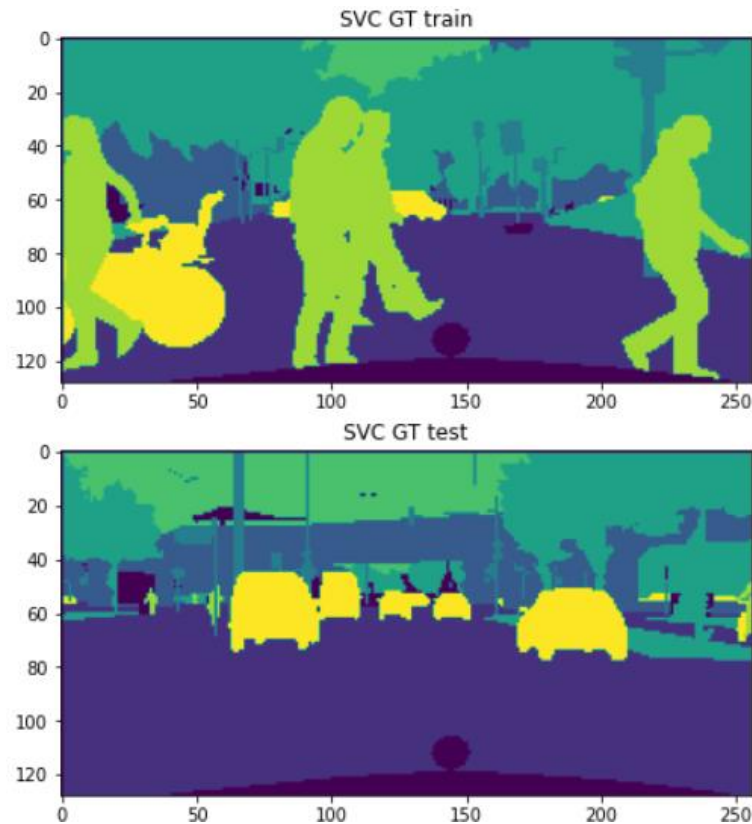


# B- Random Forest Classifier



- Temps de Fit de 4 minutes 30 pour **100 photos de 256x128**
- Jaccard /IoU de 0,99 pour le train et de 0,40 pour le test
- Jaccard macro / **Mean IoU** de 0,99 pour le train et de **0,26 pour le test**
- Accuracy de 99,9% pour le train et de 57% pour le test

# C- Linear SVC



- Temps de Fit de 1 à 10 minutes (en fonction du critère de tolérance de stop) pour **100 photos de 256x128**
- Jaccard /IoU de 0,30 pour le train et de 0,31 pour le test
- Jaccard macro / **Mean IoU** de 0,16 pour le train et de **0,17 pour le test**
- Accuracy de 46% pour le train et de 48% pour le test

# D- Amélioration possibles pour les méthodes de machine Learning simples

---

- Améliorations possibles
  - Mieux préprocesseur les inputs
  - Améliorer la Feature engineering
  - Entraîner avec un échantillon plus conséquent tout en gérant les problèmes de mémoire
  - Optimiser les hyperparamètres avec des GridSearchCV
- *Ces méthodes ne faisant plus parti de l'état de l'art depuis plus d'une dizaine d'années, il n'est pas judicieux de passer d'avantage de temps à essayer d'améliorer les résultats qui seront dans le meilleur des cas bien en dessous des résultats des méthodes de deep Learning*
- On pourra juste noter que Random forest donnera certainement de meilleurs résultats que SVC pour des problèmes multiclassés

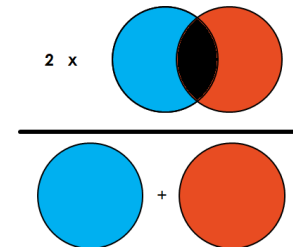
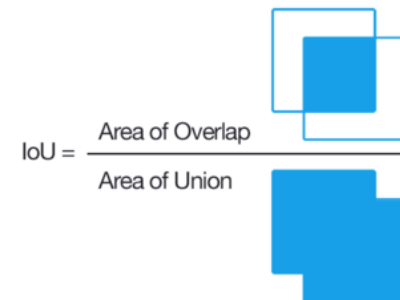
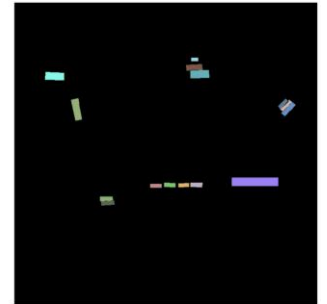
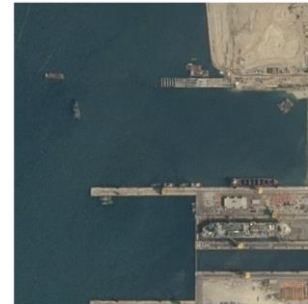


# IMAGE SEGMENTATION - PIXEL CLASSIFICATION : MODÈLES DE DEEP LEARNING

# A- Choix de la métrique

<https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>

- Segmentation d'image: quelles Métriques?
  - La pixel Accuracy : pourcentage de pixels correctement classés
    - Si une classe est fortement majoritaire , comme la mer entourant des bateaux sur une image prise du ciel, **la pixel accuracy peut facilement donner de très bons résultats , bien que la classe bateau , par exemple , soit mal segmentée,**
  - L'IoU intersection over union ou Jaccard Index
    - Le mean IoU vient pallier au problème cité ci-dessus. On calcule l'IoU par classe puis on moyenne. Si la segmentation est très bonne sur la classe majoritaire de la mer , par exemple , l'IoU de la classe mer sera proche de 1 mais si la segmentation de la classe bateau est très mauvaise , l'IoU de la classe bateau sera proche de 0 ce qui donnera un mean IoU sur les 2 classes de 0,5 environ au lieu d'un pixel accuracy proche de 90%
  - Le dice coefficient ou F1 score est proche de l'IoU



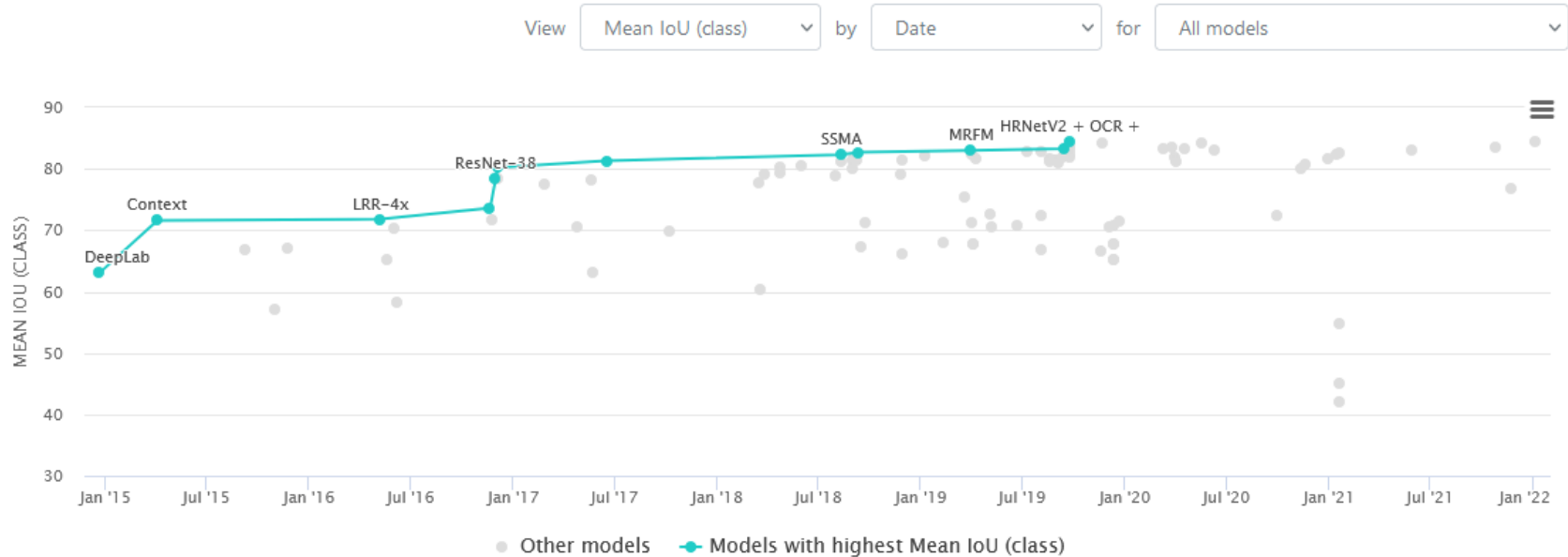
# B- Segmentation d'images: Etat de l'art

<https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes>

## Semantic Segmentation on Cityscapes test

Leaderboard

Dataset



- Avant de se lancer dans la construction d'une architecture de segmentation, nous avons regardé l'état de l'art. HRNET-OCR et les transformers semblent mener les hostilités même si en terme de performance pure, les gains semblent limiter depuis 5 ans

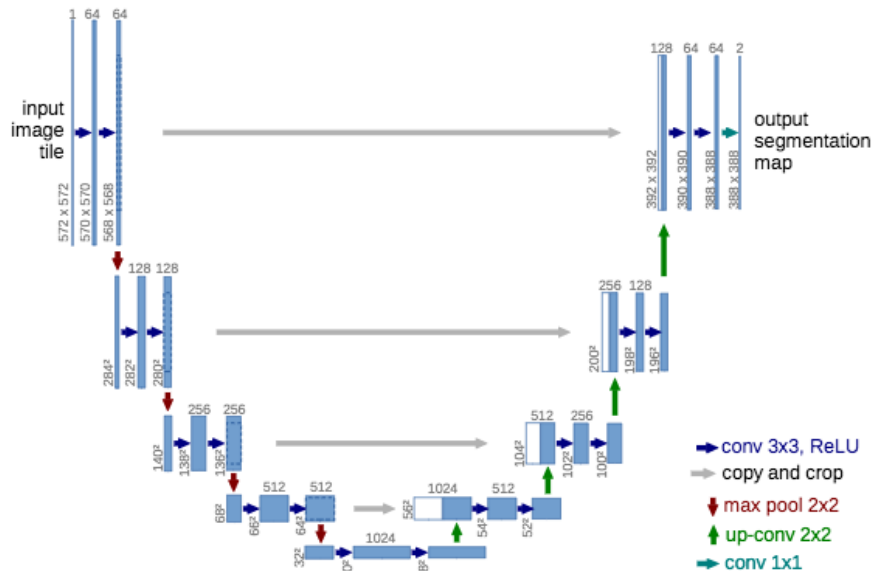
# C- Segmentation d'images: Choix de l'architecture

---

- Pour la suite , nous avons donc fait le choix de nous arrêter sur des modèles qui ont fait leur preuve , et qui ont un ratio performance/cout d'entrainement acceptable pour notre GPU
- Nous avons fait le choix de prendre une architecture encodeur-decodeur relativement classique.
  - Encodeur en VGG16 car
    - assez populaire et moins couteux qu'un resnet 50 ou 101 plus lourds en couches
    - Pré entrainé sur imagenet, parfait en poids de départ pour des images en extérieur comme cityscape
  - Décodeur Unet
    - Permet d'avoir des bords peu grossiers grâce aux skip connections. En effet un des problèmes des architectures de segmentation classiques est la perte d'information entre le passage de la faible résolution vers la haute résolution. Les Skip Connections y apportent une réponse
  - Decodeur PSPnet
    - Apporte de l'information contextuelle globale permettant de mieux définir les objets,

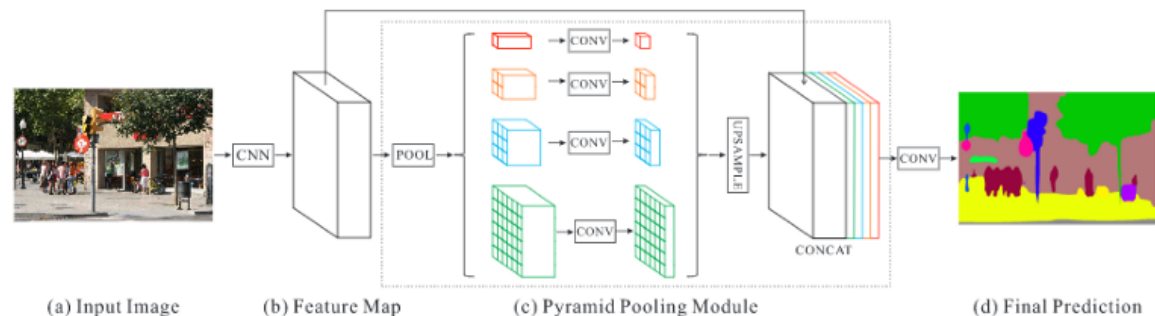


# C- Segmentation d'images: Choix de l'architecture



Vgg-unet avec 256x512 en input ,  
après test de temps de calcul pour  
notre problème cityscape à 8 classes

Vgg-PSPnet



# D-Modèles deep learning de Segmentation d'images

## WorkFlow

CREATION DES LISTES DE PAIRES DE CHEMINS IMAGES/GROUND TRUTH

CREATION DE LA FONCTION D'AUGMENTATION

CREATION DU GENERATEUR QUI SERVIRA D'ENTREE AUX MODELES KERAS

*Le générateur peut gerer plusieurs paramètres: La taille du batch, l'augmentation, le type de jeu (train, test ou validation) , le nombre de classes du ground truth*

CONSTRUCTION DE L'ARCHITECTURE ENCODEUR-  
DECODEUR VGG-UNET

CONSTRUCTION DE L'ARCHITECTURE ENCODEUR-  
DECODEUR VGG-PSPNET

CREATION DES FONCTIONS METRIQUES ADAPTEES: dice, Mean Dice, IoU, Mean IoU

PARAMETRAGE DU CALLBACK avec modele checkpoint ,  
earlystopping sur Max Validation Mean IoU avec patience de 5 epochs et sauvegarde des poids

FIT DU MODELE VGG-UNET

FIT DU MODELE VGG-PSPNET

Rechargement du modèle sauvegardé  
PREDICTION DU TEST AVEC VGG-UNET

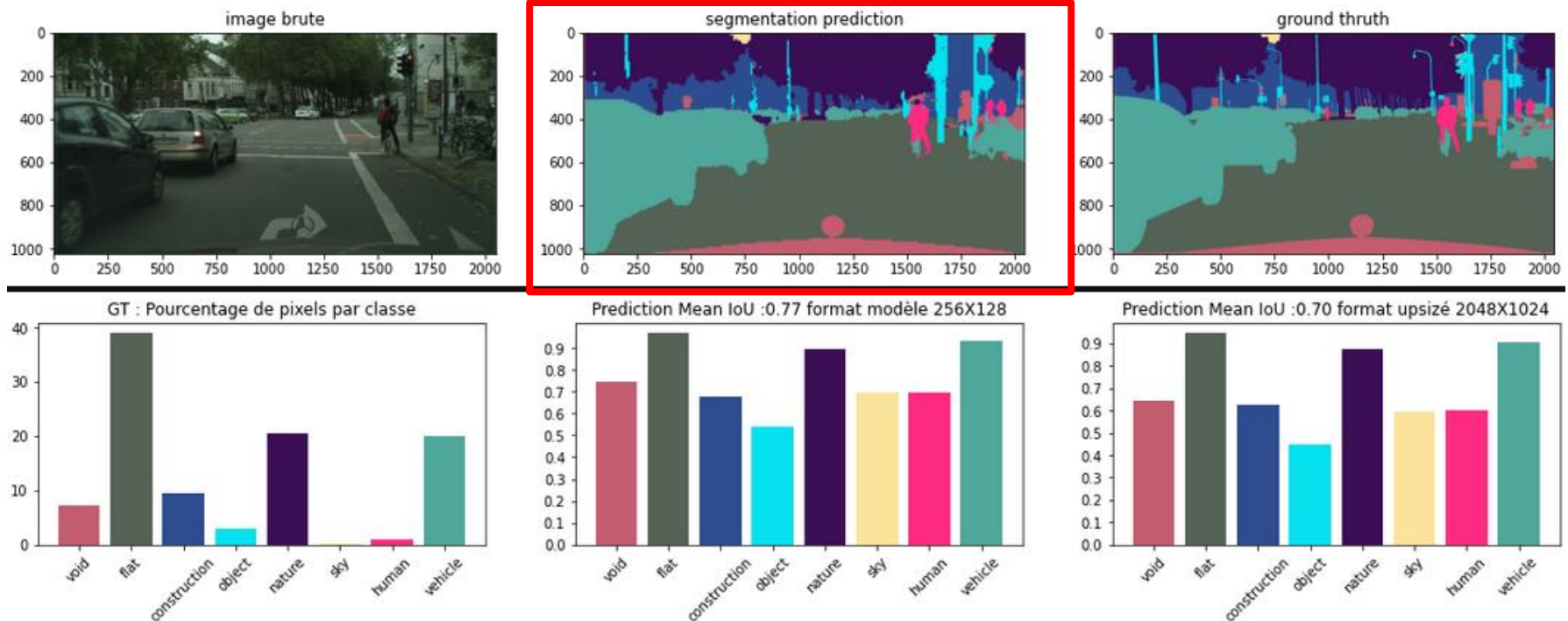
Rechargement du modèle sauvegardé  
PREDICTION DU TEST AVEC VGG-PSPNET

ANALYSE DES METRIQUES SUR LE SAMPLE TEST

AFFICHAGE DE RESULTATS SUR IMAGE TEST

# E- Analyse des résultats

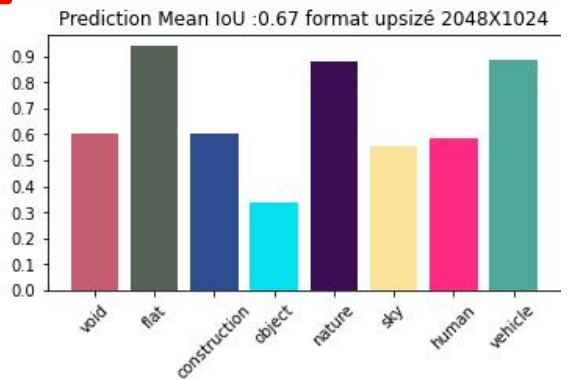
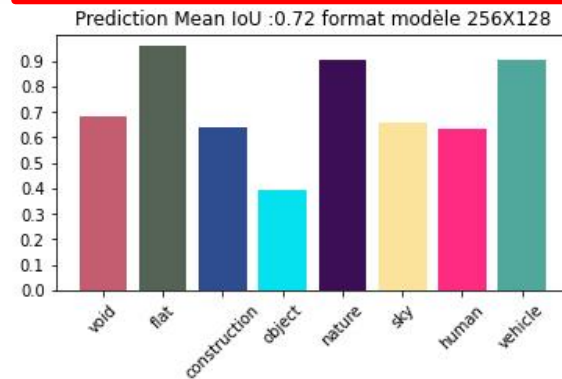
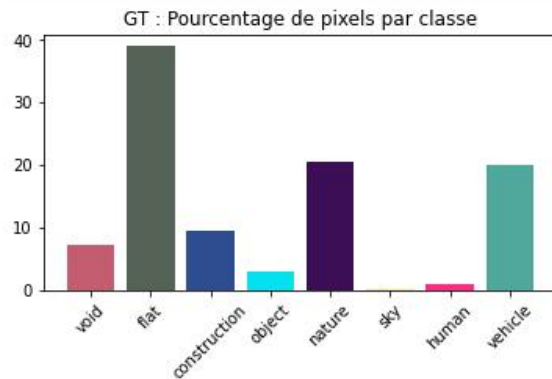
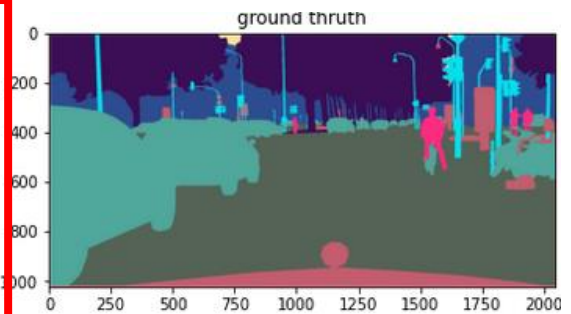
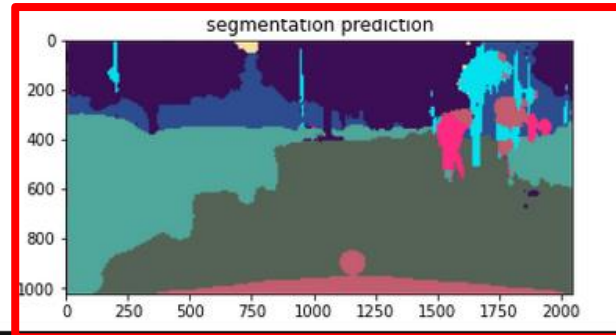
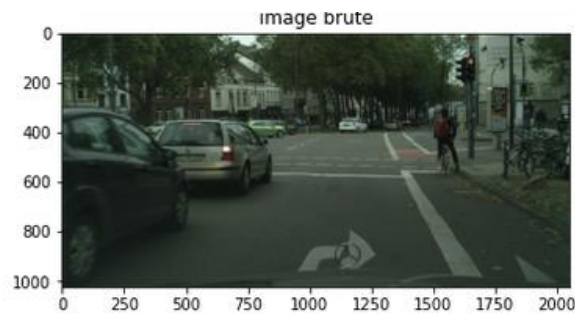
- VGG-UNET
  - Encodeur pré entraîné sur imagenet
  - Sans augmentation : 37 epochs totales de train (700sec par epoch)
  - 8 heures d'entraînement
  - De bon résultats sur des échantillons de test: un val mean IoU autour de 0,7



# E- Analyse des résultats

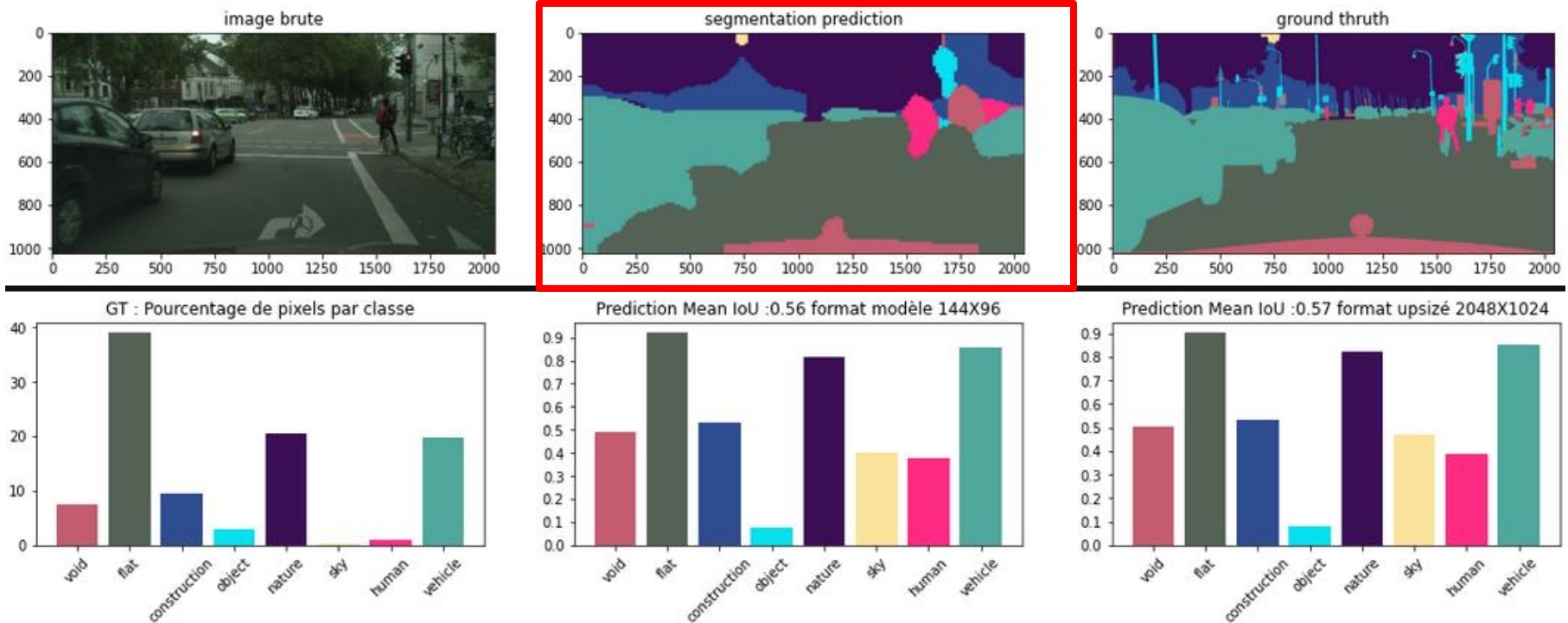
## ■ VGG-UNET avec AUGMENTATION

- Encodeur pré entraîné sur imagenet
- Sans augmentation : 37 epochs totales de train (700sec par epoch)
- 8 heures d'entraînement
- De bon résultats sur des échantillons de test:
- val\_mean\_dice: 0.7823 - val\_mean\_iou: 0.6691 - val\_flat\_dice: 0.8700 - val\_flat\_iou: 0.7703 - val\_accuracy: 0.9110



# E- Analyse des résultats

- PSP-UNET
  - Encodeur pré entraîné sur imagenet
  - Sans augmentation : 30 epochs totales de train (622s sec par epoch)
  - 6 heures d'entraînement
  - De bon résultats sur des échantillons de test mais moins bien définis (pas de skip connections)
  - **Val\_mean\_dice: 0,66 - val\_mean\_iou: 0.5370** - val\_flat\_dice: 0.8186 - val\_flat\_iou: 0.6931 - val\_accuracy: 0.8558



**D**

# DEPLOIEMENT D'UNE API FLASK DE PREDICTION DE SEGMENTATION D'IMAGES BASÉE SUR VGG-UNET VIA AZURE WEB APP

# A-Deploiement de l'API Workflow

CREATION DE L'API FLASK BASEE SUR VGG-UNET ENTRAINE

TEST EN LOCAL

CREATION DU DEPOT GIT EN LOCAL INCORPORANT HEBERGEANT L'APPLICATION

CREATION DU SERVICE WEB APP AZURE

SELECTION DU PLAN B1  
(tarification/materiel)

Récupération de l'adresse du dépôt Git lié au service

1<sup>er</sup> push Git de l'application puis débogage  
L'application doit s'appeler app.py  
Certains requirements de libraires doivent être adaptés pour azure  
Les lieux de stockage d'images temporaires doivent être adaptés

Push avec déploiement réussie  
Application fonctionnelle à l'adresse éphémère suivante:  
<https://p8webapp.azurewebsites.net>

# B-Application déployée

← → ↻ 🏠 <https://p8webapp.azurewebsites.net> ★ 📁 ☰

## BONJOUR, VOUS ÊTES SUR LA PAGE DE TEST DU MODÈLE DE SEGMENTATION DU PROJET 8 DU PARCOURS OCR IA

POUR CELA, NOUS ALLONS UTILISÉ UN MODÈLE VGG\_UNET 256X512 ENTRAÎNÉ SUR DES IMAGES CITYSCAPES

**SELECTIONNER LES FICHIERS IMAGE (IM.PNG) À SEGMENTER ET GROUNDTRUTH (GT.PNG) À PARTIR D'UN MÊME REPERTOIRE**

Aucun fichier sélectionné.


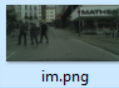
Envoi du fichier

← → ↻ 📁 > nab wind > OCR > P8 > input > pour\_app > test c 🔍 Rechercher dans : test c

Organiser ▾ Nouveau dossier 📄 📁 ?

★ Accès rapide

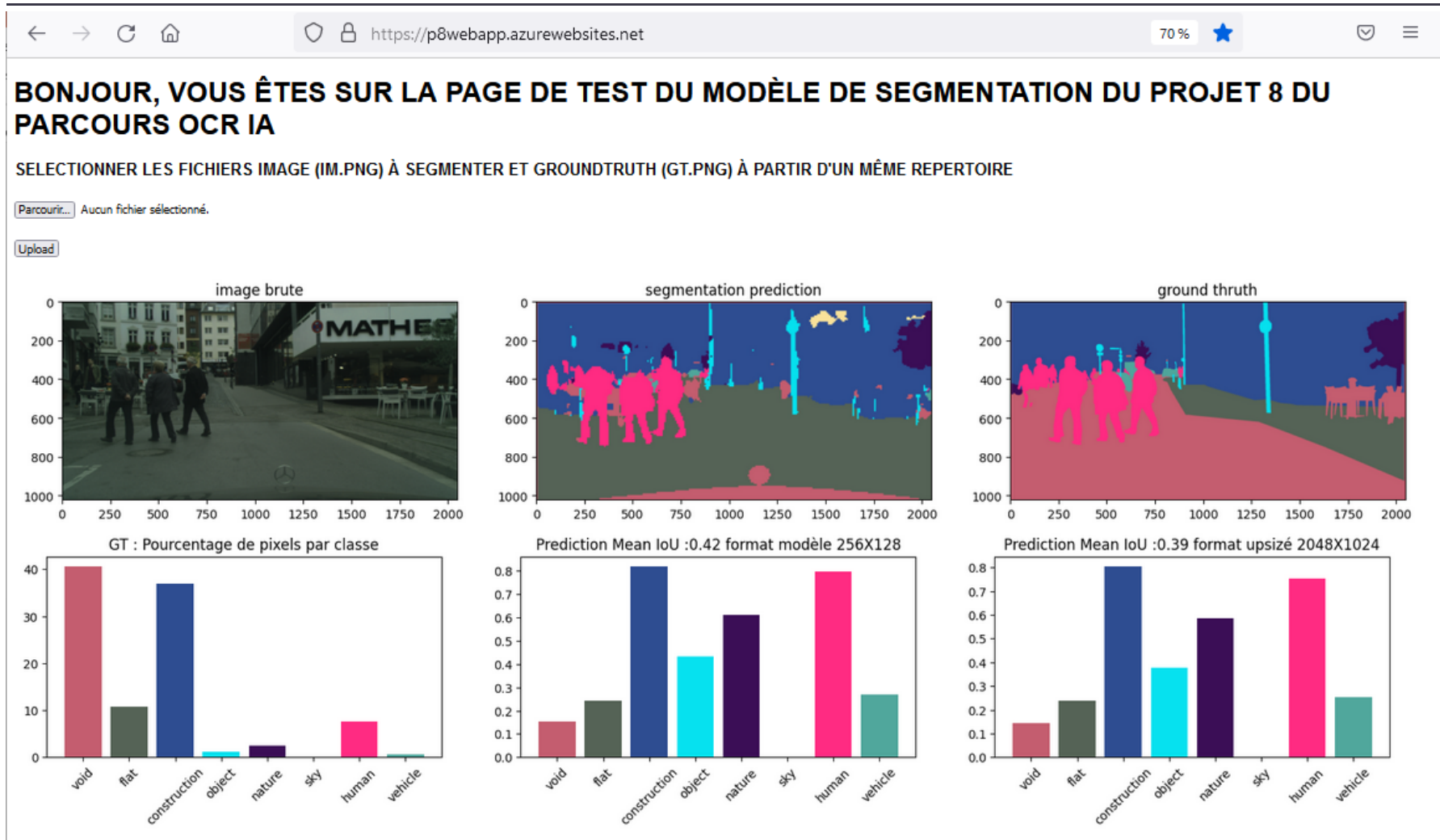
- 📁 Bureau ➡
- 📁 Télécharger ➡
- 📁 Documents ➡
- 📁 Images ➡
- 📁 models
- 📁 P8
- 📁 P8
- 📁 yliess

  
gt.png  
im.png

Nom du fichier : "im.png" "gt.png" Tous les fichiers (\*.\*)



# C-Application déployée - Résultat





# CONCLUSION

# A-Summary

---

Modeles	type	temps d'entrainement	Validation Mean IoU	commentaires
Linear SVC	Machine Learning simple	sans gridsearch: 1 à 10 min pour 100 photos de 35K pixels fonction de la finesse du critère d'arret	0,17	résultats mauvais necessite plus d'entrainement necessite plus de feature engineering
Random Forest Classifier	Machine Learning simple	4,5 min pour 100 photos de 35K pixels	0,26	résultats mauvais necessite plus d'entrainement necessite plus de feature engineering
VGG PSPnet	Deep Learning	600 à 800 seconds par epoch de 2300 photos de 384x576 en fonction de l'augmentation 30 à 50 epochs 6 à 8h	0,54	résultats encourageants
VGG Unet	Deep Learning	600 à 800 seconds par epoch de 2300 photos de 256x512 en fonction de l'augmentation 30 à 50 epochs 6 à 8h	0,69 0,66 AVEC AUGMENTATION	bons resultats bors bien definis