Note méthodologique :

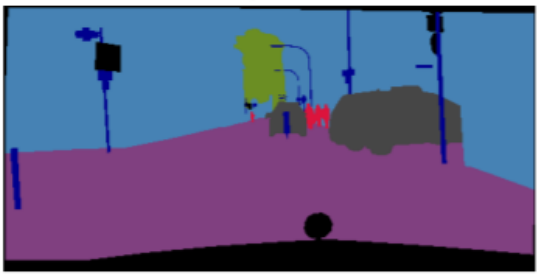
preuve de concept

I - Dataset retenu

Le dataset **Cityscapes** est une référence mondiale pour la segmentation d'images en milieu urbain.
Il propose des images haute résolution (2048×1024 px) capturées dans plus de 50 villes européennes, annotées pixel par pixel pour l'analyse de scènes routières (piétons, véhicules, bâtiments, routes…).

Image

Masque (GT)



1.Taille et structure du dataset original

Cityscapes fournit :

- **5000 images finement annotées**, réparties en :
    - 2975 images d'entraînement
    - 500 images de validation
    - 1525 images de test
- **34 classes** au total, dont **19 classes officielles** utilisées pour l'évaluation standard.
- Des **masques très détaillés**, annotés manuellement.

Ces caractéristiques font de Cityscapes l'un des datasets les plus complets et exigeants pour les systèmes de vision embarqués.

2. Adaptation du dataset pour la preuve de concept

Afin de rendre l'entraînement compatible avec les contraintes du POC (temps de calcul, cohérence du pipeline, reproductibilité), un prétraitement a été appliqué.
Deux modifications principales ont été effectuées :

A. Réduction du nombre de classes à 8 catégories

Les 19 classes officielles ont été regroupées en **8 classes principales**, correspondant aux éléments les plus structurants d'une scène urbaine :

Classe (POC)	Description
flat	route, trottoir, surfaces roulables
human	piétons, silhouettes
vehicle	voitures, bus, camions, motos
construction	bâtiments, murs, structures
object	poteaux, panneaux, mobilier urbain
nature	arbres, herbes, végétation
sky	ciel
void	zones ignorées / non pertinentes

Ce regroupement permet :

- de **simplifier la tâche de segmentation**,
- de **réduire l'impact des classes rares**,
- de **conserver les grandes familles d'objets essentielles** à la conduite autonome.

B. Redimensionnement à 256 pixels de hauteur

Les images Cityscapes étant très grandes, elles ont été uniformisées en **256px** pour la hauteur.

Ces dimensions réduites permettent :

- d'accélérer significativement l'entraînement,
- de réduire l'usage mémoire GPU,
- de conserver les structures globales nécessaires à la segmentation sémantique.

3.Dataset final utilisé dans le POC

Après prétraitement, la version utilisée est :

- **2975 images** pour l'entraînement
- **500 images** pour la validation
- Chaque image possède son **masque associé**, remappé vers les **8 classes définies**
- L'ensemble des données est normalisé et homogène, compatible avec PyTorch et les deux modèles testés (DeepLabV3+ et Mask2Former)

Cette version simplifiée de Cityscapes conserve la diversité visuelle du dataset original, tout en permettant un entraînement rapide et reproductible dans le cadre du POC.

II - La modélisation

Cette section présente le cadre méthodologique mis en place pour entraîner et comparer deux modèles de segmentation sémantique — **DeepLabV3+** et **Mask2Former** — à partir du dataset Cityscapes réduit à 8 classes. L'objectif était de garantir une expérimentation reproductible, équilibrée et adaptée à un contexte de preuve de concept.

1. Préparation et structuration des données

A. Sélection et regroupement des classes

Cityscapes comporte initialement 30 classes annotées. Pour optimiser le temps d'expérimentation et concentrer l'analyse sur les objets urbains clés, seule une version **regroupée en 8 classes** a été retenue . Ce remapping a été appliqué aux masques originaux pour obtenir un jeu de labels homogène et compatible avec les deux architectures.

B. Prétraitement des images

Pour réduire la charge de calcul, toutes les images RGB et leurs masques ont été :

- **redimensionnés en 256×256 px**,
- **normalisés selon les standards ImageNet**,
- **convertis au format tensor PyTorch**,
- **alignés train/val** via un split fixe :
    - **2975 images train**
    - **500 images validation**

Ce pipeline a permis de constituer un dataset **léger, cohérent, reproductible**.

2. Mise en place des architectures

Deux modèles ont été entraînés selon une procédure identique.

A. Modèle de référence — DeepLabV3+ (ResNet50)

```python
def build_model(aux_loss=True):
    """
    Build a DeepLabV3 model with a ResNet50 backbone and a custom classifier head
    adapted to the number of segmentation classes.

    Args:
        aux_loss (bool): Whether to enable the auxiliary loss branch.

    Returns:
        nn.Module: The configured DeepLabV3 model.
    """
    w  = DeepLabV3_ResNet50_Weights.COCO_WITH_VOC_LABELS_V1
    m  = deeplabv3_resnet50(weights=w, aux_loss=aux_loss)
    in_ch_main = m.classifier[4].in_channels
    m.classifier[4] = nn.Conv2d(in_ch_main, NUM_CLASSES, kernel_size=1)

    if aux_loss:
        in_ch_aux = m.aux_classifier[4].in_channels
        m.aux_classifier[4] = nn.Conv2d(in_ch_aux, NUM_CLASSES, kernel_size=1)

    return m
```

Utilisé comme baseline robuste.
Pipeline mis en place :

- backbone ResNet50 pré-entraîné
- module ASPP
- encodeur/décodeur complet
- head de segmentation adapté aux 8 classes

B. Modèle récent — Mask2Former (Swin Transformer Small)

```python
checkpoint = "facebook/mask2former-swin-small-coco-panoptic"
model = Mask2FormerForUniversalSegmentation.from_pretrained(
    checkpoint,
    num_labels=NUM_CLASSES,
    id2label=id2label,
    label2id=label2id,
    ignore_mismatched_sizes=True,
)
model.to(DEVICE)
print("Modèle prêt :", type(model).__name__)
from transformers import Mask2FormerImageProcessor
processor = Mask2FormerImageProcessor.from_pretrained(checkpoint, reduce_labels=False)
```

Implémenté via HuggingFace Transformers :

- backbone Swin Transformer
- pixel decoder multi-échelle
- Mask Transformer Decoder
- head de mask classification configuré sur 8 classes

Aucune modification structurelle n’a été apportée : seules les couches finales ont été adaptées aux classes Cityscapes regroupées.

3. Optimisation des hyperparamètres : Random Search

Pour assurer une comparaison équitable entre les deux modèles, un **random search indépendant** a été réalisé pour chacun, sur un sous-dataset réduit (**300 train / 50 val**).
Chaque modèle a été testé sur **10 configurations aléatoires**, avec :

- **learning rate**
- **weight decay**
- **optimiseur (Adam, AdamW, SGD) pour DeepLabV3+**
- **usage ou non de l'auxiliary loss** (pour DeepLabV3+)
- **nombre d'epochs = 10** (fixe)

À l'issue des 10 essais, la configuration obtenant le meilleur **mIoU** a été sélectionnée pour l'entraînement final sur 2975/500 images.

Hyperparamètres retenus:

- DeepLabV3+
- opt = Adam
- lr = 1.5169e-04
- wd = 1.2455e-06
- aux_loss = True
- epochs = 10

- Mask2Former
- lr = 5.6844e-05
- wd = 1.5127e-04
- epochs = 10

Ces configurations optimisées ont ensuite servi à l'entraînement complet des deux modèles avec suivi MLflow.

4. Entraînement final et pipeline technique

Les modèles sélectionnés ont été entraînés selon un processus identique :

- chargement du dataset prétraité
- application du modèle avec ses hyperparamètres optimisés
- suivi automatique des métriques et paramètres via **MLflow**
- sauvegarde des meilleurs checkpoints
- export des masques colorisés pour intégration API/Streamlit

Cette standardisation assure des résultats **reproductibles**, comparables, et facilement intégrables dans un pipeline de déploiement.

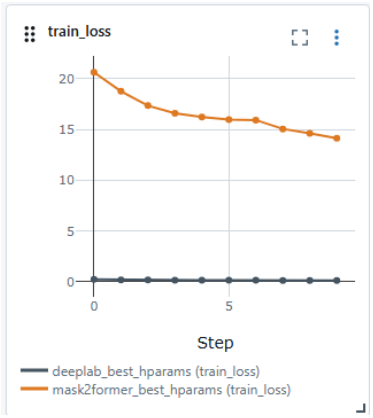
III - Synthèse des résultats

L'objectif de cette preuve de concept était de comparer deux modèles de segmentation — **DeepLabV3+ (ResNet50)** et **Mask2Former (Swin Transformer Small)** — sur un sous-ensemble du dataset Cityscapes réduit à **8 classes**.
Deux phases d'entraînement ont été menées :

- un **échantillon réduit** (300 / 50) pour la recherche d'hyperparamètres
- un **jeu complet** (2975 / 500) pour l'entraînement final

L'ensemble des métriques a été suivi et centralisé via **MLflow**.

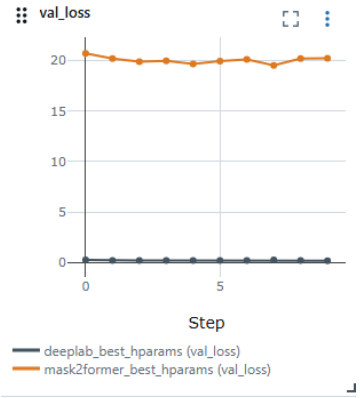
1. Performances pendant l'entraînement



- **DeepLabV3+** présente une train_loss très faible (≈ 0.3 → 0.1).
- **Mask2Former** affiche des pertes plus élevées (≈ 20 → 15).

Ces valeurs **ne sont pas comparables** entre les modèles : leurs fonctions de perte sont différentes.
La train_loss n'est pas pertinente pour départager les architectures.

2. Loss de validation

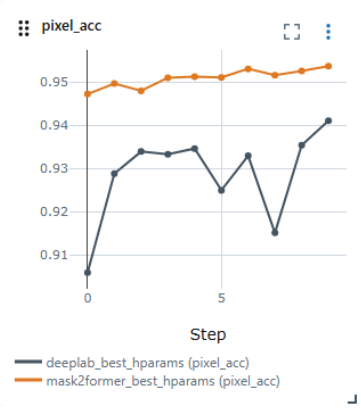


Même constat que pour la loss d'entraînement :

- DeepLabV3+ : val_loss très faible
- Mask2Former : val_loss élevée mais stable

La validation loss ne permet pas d'évaluer correctement lequel des deux modèles segmente le mieux.

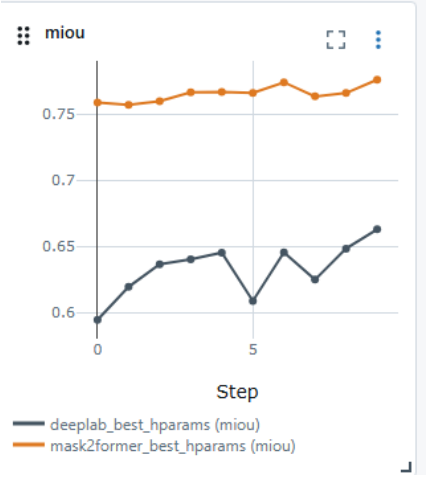
3. Précision pixel (Pixel Accuracy)



- **DeepLabV3+** : ~0.91 – 0.94
- **Mask2Former** : ~0.94 – 0.95

Mask2Former obtient systématiquement de meilleurs résultats pixel par pixel.

4. Qualité de segmentation (mIoU)



- **DeepLabV3+** : 0.59 – 0.66
- **Mask2Former** : 0.75 – 0.78

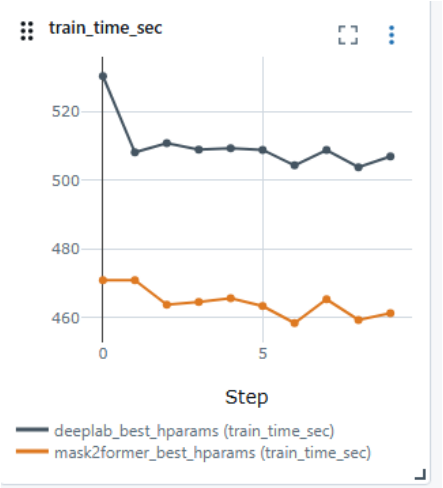
La différence est très nette :
Mask2Former dépasse DeepLabV3+ de 12 à 18 points de mIoU.
Les frontières d'objets, surfaces larges et structures urbaines sont mieux segmentées grâce à l'attention globale du Transformer.

5. Vitesse d’inférence

- DeepLabV3+ : ~5.6 – 5.9 imgs/s
- Mask2Former : ~6.3 – 6.5 imgs/s

Mask2Former est légèrement plus rapide à l’inférence, malgré son architecture plus récente.

6. Temps d’entraînement



- DeepLabV3+ : ~520 sec/epoch
- Mask2Former : ~465 sec/epoch

Mask2Former s’entraîne plus vite, notamment grâce à un pipeline d’entraînement mieux optimisé sur les architectures Transformers modernes.

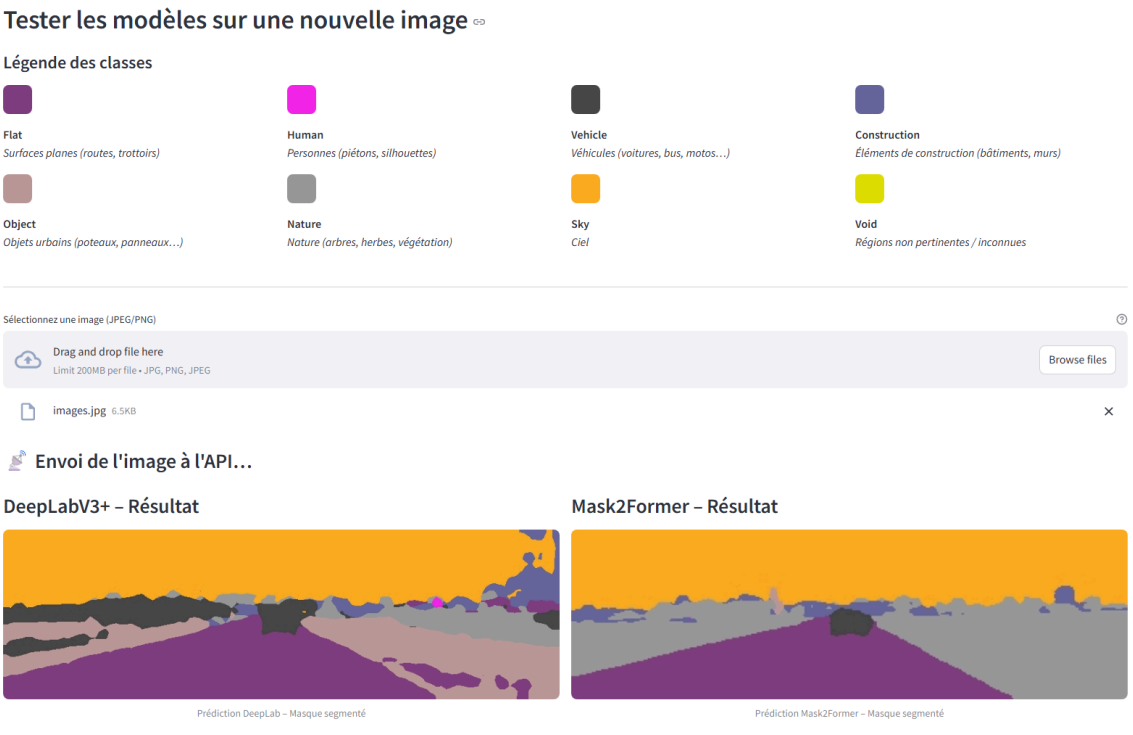
7. Conclusion générale

Mask2Former se distingue nettement comme **le meilleur modèle** dans ce POC :

- meilleure **qualité de segmentation** (mIoU),
- meilleure **cohérence pixel-wise**,

- entraînement **plus rapide**,
- inférence **plus rapide**,
- architecture **plus robuste** pour les scènes urbaines complexes.

Dans une perspective d'API ou de système embarqué en vision temps réel, **Mask2Former est le modèle le plus adapté**, plus performant et plus moderne que DeepLabV3+.



IV - Analyse de la feature importance globale et locale

Dans cette preuve de concept, aucune méthode d'interprétabilité dédiée (Grad-CAM, SHAP…) n'a été utilisée.
L'analyse repose donc uniquement sur l'observation qualitative des comportements des modèles lors de la segmentation.

1. Importance globale

- Mask2Former semble exploiter une compréhension globale de la scène : grandes zones homogènes, continuité spatiale, relations à longue distance.

- DeepLabV3+ repose davantage sur des informations locales : textures, variations fines, motifs proches.

Ces différences découlent directement de leurs architectures (Transformers vs CNN).

2. Importance locale

- DeepLabV3+ capte bien les détails locaux mais peut perdre en cohérence dans les zones complexes.

- Mask2Former produit des zones plus régulières mais lisse parfois certains petits détails.

3. Erreurs typiques

- DeepLabV3+ : confusions sur textures proches, perte de détails fins.

- Mask2Former : contours parfois trop lissés, objets très petits parfois ignorés.

V. Limites et améliorations possibles

1. Limites du POC

- **Réduction à 8 classes**
  Le regroupement des classes simplifie la tâche et accélère le prototypage, mais supprime les classes rares et les structures fines. Cela peut artificiellement augmenter certaines métriques.

- **Dataset limité**
  Le jeu réduit (300 images) rend les modèles sensibles au surapprentissage et aux variations de scènes. Le dataset complet (2975 images) améliore la stabilité mais reste modeste pour exploiter pleinement un Transformer.

- **Coût computationnel**
  Même optimisé, Mask2Former reste plus coûteux qu'un CNN classique : mémoire GPU élevée, entraînement et inférence plus lourds pour un déploiement embarqué.

- **Interprétabilité limitée**
  Comme pour la plupart des modèles de vision profonde, il est difficile d'expliquer précisément quelles caractéristiques influencent la prédiction sans outils avancés d'analyse.

2. Améliorations possibles

- **Augmentation du dataset et de la diversité**
  Utilisation de transformations (contraste, crop, bruit, luminosité, rotation) ou de stratégies automatiques (RandAugment, AutoAugment) pour enrichir la variabilité visuelle.

- **Entraînement plus long et réglage fin**
  Étendre le fine-tuning (30–50 epochs) pour stabiliser le mIoU et améliorer la régularité des frontières.

- **Exploration d'architectures plus récentes**
  SegFormer, InternImage ou variantes plus légères de Mask2Former pour un meilleur compromis performance / coût.

- **Amélioration du pipeline de déploiement**
  Micro-batching, traitement asynchrone, et optimisation des conteneurs pour réduire les temps de réponse dans une API temps réel.

3. Ouverture : vers une segmentation embarquée et panoptique

Dans une perspective d'intégration embarquée pour les véhicules autonomes, une extension naturelle de ce POC serait d'exploiter **la segmentation panoptique** — une approche qui unifie segmentation sémantique et instance.
Mask2Former, par sa conception, **supporte nativement la segmentation panoptique sans changer d'architecture**, ce qui en fait un candidat idéal pour :

- différencier chaque instance de véhicule ou de piéton,
- gérer simultanément les surfaces et les objets,
- offrir une perception plus proche des besoins réels d'un système embarqué.

Cette évolution constituerait une étape clé vers une solution complète de perception visuelle embarquée.

Sources:

https://arxiv.org/abs/2112.01527
https://huggingface.co/docs/transformers/model_doc/mask2former
https://mask2former.com/how-accurate-is-mask2former/
https://medium.com/@HannaMergui/maskformer-per-pixel-classification-is-not-all-you-need-for-semantic-segmentation-1e2fe3bf31cb
https://www.researchgate.net/figure/nference-results-of-Mask2Former-U-Net-and-DeepLabV3-tested-on-PV01-PV03-PV08_fig2_380300222
https://debuggercafe.com/fine-tuning-mask2former/
https://blog.roboflow.com/what-is-mask2former/