

🚋FUTURE VISION TRANSPORT🚗

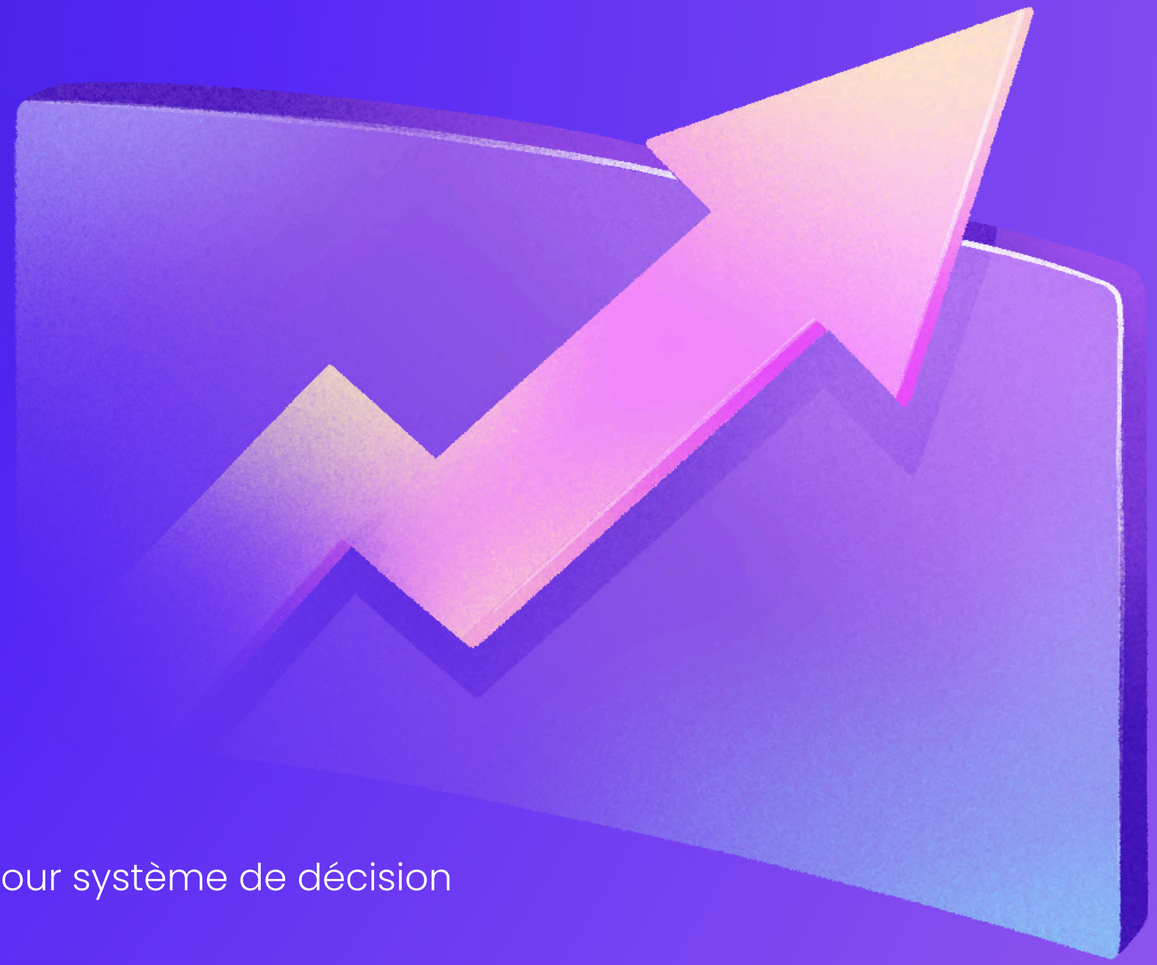
TRAITEMENT D'IMAGES POUR SYSTÈMES EMBARQUÉS DE VOITURES AUTONOMES

Grégoire MUREAU



INTRODUCTION

Problématique: **Future Vision Transport** développe des systèmes embarqués pour véhicules autonomes et doit intégrer un module de segmentation sémantique temps réel.



Solution souhaitée

Modèle de segmentation 8 catégories + API déployée pour système de décision temps réel.

OBJECTIFS DÉTAILLÉS

01

Entraînement d'un **modèle de segmentation** :

- Architectures encoder-décodeur (U-Net, FPN)
- Backbones pré-entraînés (EfficientNet, MobileNet)
- Dans un environnement MLOps

02

Développement d'une **API de segmentation** :

- Entrées : dataset Cityscapes ou images uploadées
- Sorties : masques + statistiques d'analyse
- Conteneurisée (Docker)
- Github CD

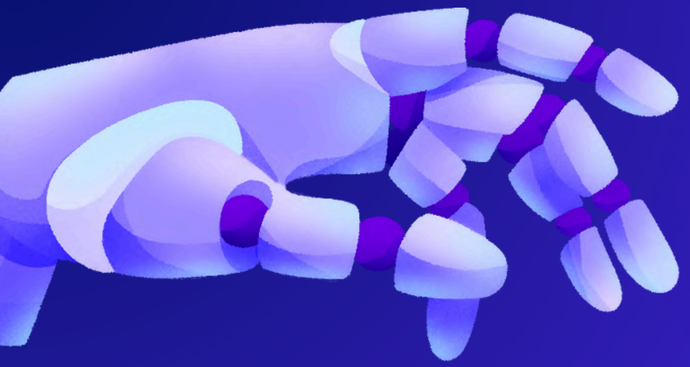
03

Déploiement de l'API

04

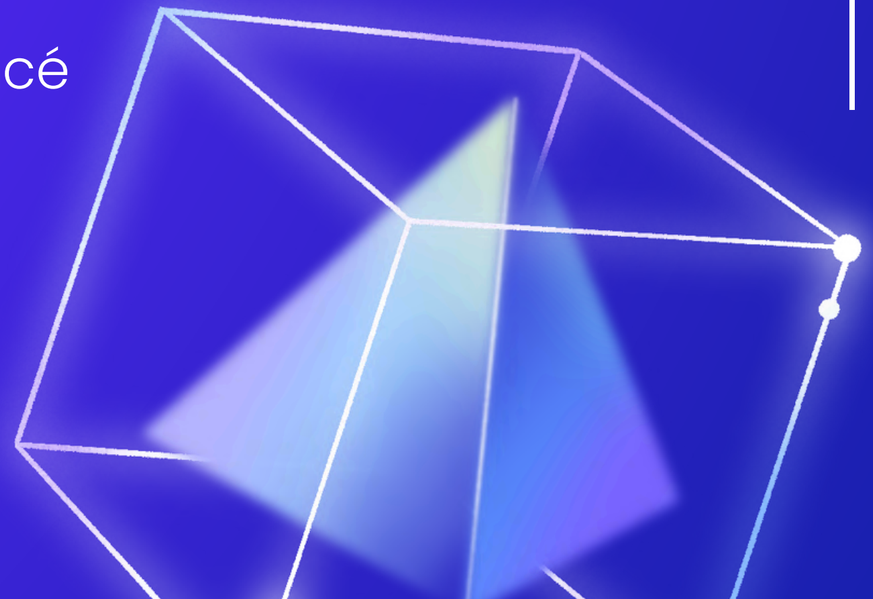
Challenge personnel : **serveur personnel** :

- Minimiser la dépendance aux GAFAM
- Assurer la souveraineté des données
- Garantir l'indépendance vis-à-vis des infrastructures des grandes entreprises

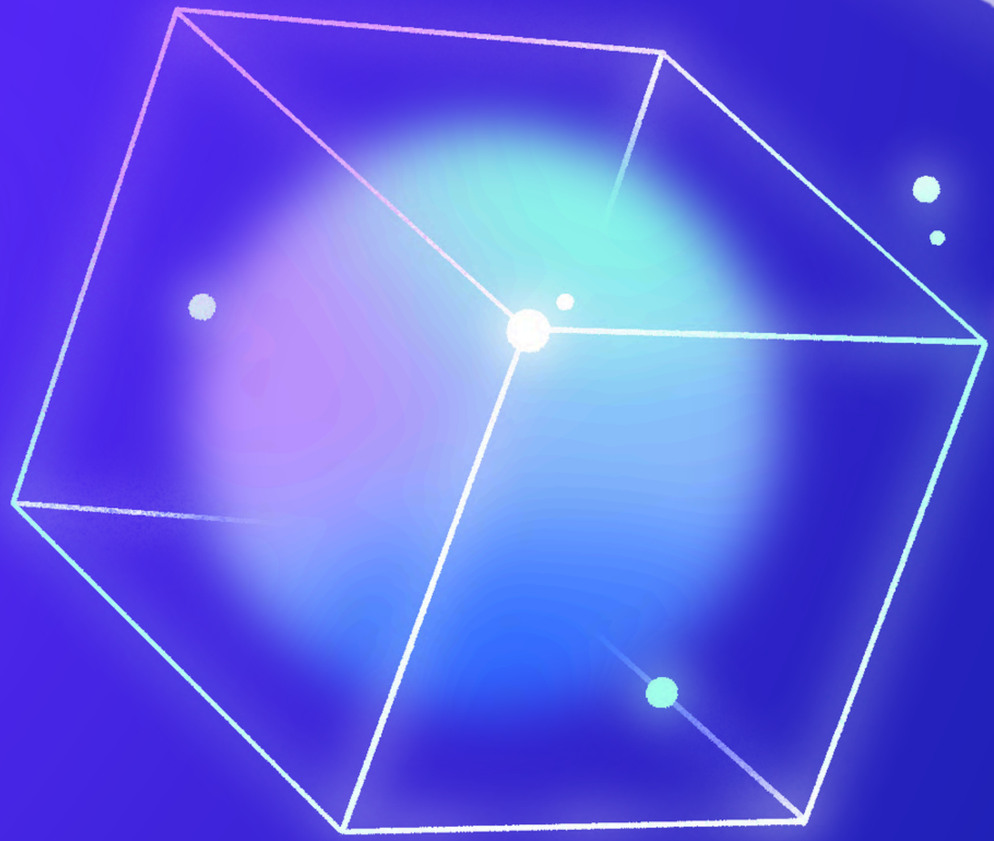


SOMMAIRE

1. Entraînement de modèles dans un environnement MLOps
   - Méthodologie comparative (U-Net vs FPN)
   - **Suivi MLflow des expérimentations**
   - **Résultats et sélection du modèle optimal**
2. Mise en production d'un modèle sur mesure avancé
   - Développement API de segmentation
   - Pipeline CD
   - Démonstration (masques, statistiques, comparaisons)
3. Règles RGPD
4. Conclusion et Limites



ENTRAINEMENT DE MODÈLES



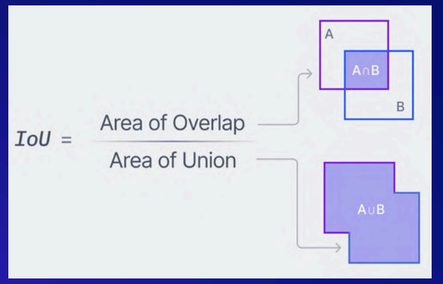
MÉTHODOLOGIE

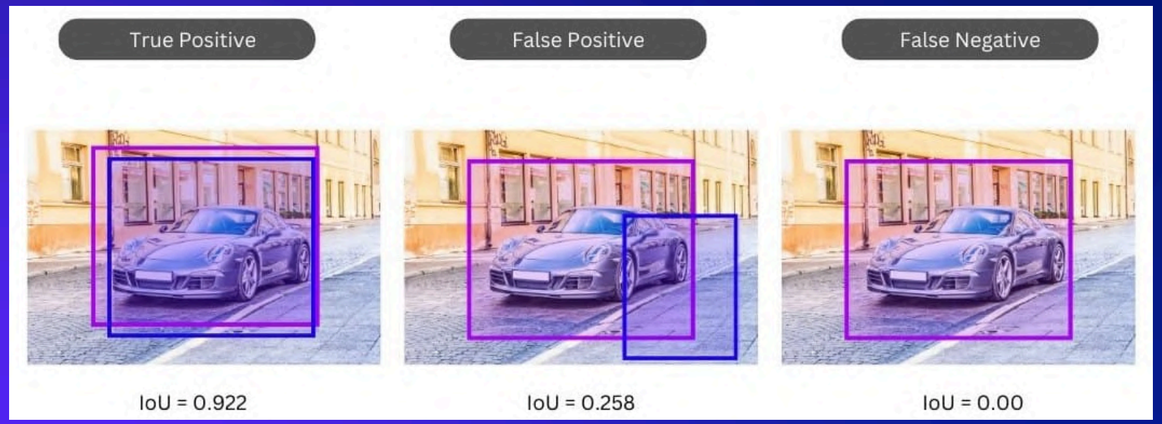
JEU DE DONNÉES, KPI, MODÈLES

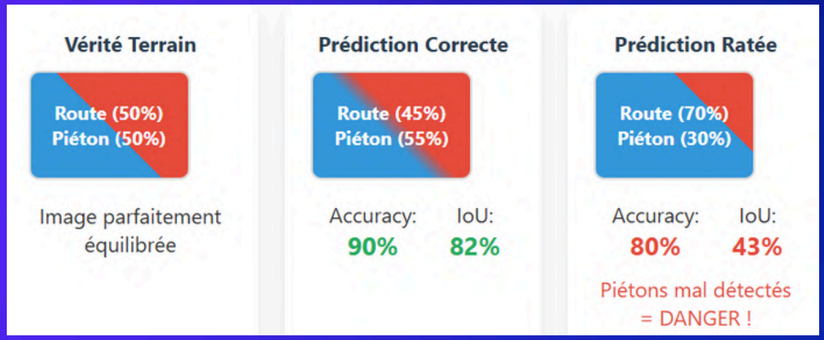
- BDD : Cityscapes
  - 2975 images d'entraînement
    - 2380 images d'entraînement (80%)
    - 595 images de validation (20%)
  - 500 images de test (jeu "val" officiel Cityscapes)
  -
- 34 catégories sémantiques → 8 catégories sémantiques

- Entraînement : Google Colab (GPU T4), sessions limitées 2h

- Objectif : Segmentation précise pour sécurité routière → KPI privilégié : Mean IoU (Intersection over Union) =/= accuracy globale







MÉTHODOLOGIE

PRÉTRAITEMENT

- Redimensionnement : 224×224 pixels (optimisation mémoire GPU)
- Normalisation : Division par 255 (pixels [0,1])
- Regroupement classes : 34 classes Cityscapes → 8 catégories sémantiques
- Augmentation données (1 chance sur 2) :
  - Retournement horizontal aléatoire (p=0.5)
  - Ajustement luminosité aléatoire (δ=0.1)

Mapping Cityscapes : 34 → 8 Catégories

0-Flat | 1-Human | 2-Vehicle | 3-Construction | 4-Object | 5-Nature | 6-Sky | 7-Void

[0] unlabeled	[1] ego vehicle	[2] rectif. border
[3] out of roi	[4] static	[5] dynamic
[6] ground	[7] road	[8] sidewalk
[9] parking	[10] rail track	[11] building
[12] wall	[13] fence	[14] guard rail
[15] bridge	[16] tunnel	[17] pole
[18] pole group	[19] traffic light	[20] traffic sign
[21] vegetation	[22] terrain	[23] sky
[24] person	[25] rider	[26] car
[27] truck	[28] bus	[29] caravan
[30] trailer	[31] train	[32] motorcycle
[33] bicycle		

MÉTHODOLOGIE

ENTRAINEMENT DES MODÈLES : CONFIGURATIONS RETENUES

Choix des **modèles** :

- Encodeurs
  - Pré-entraînés avec *transfer learning* (ImageNet) sauf mention "*From scratch*"
  - Poids :
    - Gelés : plus rapide
    - Dégelés : from sratch, fine-tuning
- Décodeurs: U-Net et FPN
- Contraintes : GPU T4 Google Colab, sessions 2h max

Configuration commune :

- Batch size 8, max 20 époques, Adam optimizer
- LR : 1e-4 (base) / 5e-5 (fine-tuning)
- Early stopping + ReduceLROnPlateau sur Mean IoU
- Checkpoints sur amélioration Mean IoU

Archi	Encoder	Stratégie	Aug
FPN	EfficientNetB0	Fine-tuning	OUI
FPN	ResNet34	Fine-tuning	OUI
FPN	ResNet34	Frozen	OUI
FPN	ResNet34	Frozen	NON
FPN	EfficientNetB0	Frozen	OUI
FPN	EfficientNetB0	Frozen	NON
FPN	MobileNetV2	Frozen	OUI
FPN	MobileNetV2	Frozen	NON
U-Net	EfficientNetB0	Frozen	OUI
U-Net	EfficientNetB0	Frozen	NON
U-Net	MobileNetV2	Frozen	OUI
U-Net	MobileNetV2	Frozen	NON
FPN	Vanilla (VGG16)	From Scratch	OUI
FPN	Vanilla (VGG16)	From Scratch	NON
U-Net	Vanilla	From Scratch	OUI
U-Net	Vanilla	From Scratch	NON

MÉTHODOLOGIE

ENTRAINEMENT DES MODÈLES: BACKBONES (ENCODEURS)

VGG16

138M

BASELINE "GROSSE PUISSANCE"

Précision

Efficacité Embarquée

Référence haute performance, consommation élevée

ResNet34

21M

RÉFÉRENCE STANDARD

Précision

Efficacité Embarquée

Équilibre classique précision/ressources

EfficientNetB0

5.3M

COMPROMIS OPTIMAL

Précision

Efficacité Embarquée

Architecture scientifiquement optimisée

MobileNetV2

3.5M

ULTRA-LÉGER EMBARQUÉ

Précision

Efficacité Embarquée

Spécialement conçu pour mobile/edge

MÉTHODOLOGIE

ENTRAINEMENT DES MODÈLES: ARCHITECTURES (DÉCODAGE)

U-Net : "Téléphone Arabe"

28×28 DEC ← ENC 28×28 uniquement

⬇ chaîne

56×56 DEC ← 28×28 DEC + ENC 56×56

⬇ chaîne

112×112 DEC ← 56×56 DEC + ENC 112×112

⬇ chaîne

224×224 FINAL ← 112×112 DEC + ENC 224×224

Téléphone Arabe

Info passe niveau par niveau
Une couche encodeur à la fois

FPN : "Conférence Call"

28×28 FPN ← ENC 28×28 uniquement

⬇ fusion

56×56 FPN ← TOUT précédent + ENC 56×56

⬇ fusion

112×112 FPN ← TOUT précédent + ENC 112×112

⬇ fusion

224×224 FINAL ← TOUT précédent + ENC 224×224

Conférence Call

Fusion cumulative progressive
Toutes les couches simultanément

U-Net

2015 - Segmentation Biomédicale

PRÉCISION LOCALE



Architecture symétrique en "U"

- Contours ultra-précis
- Détails fins préservés
- Frontières nettes
- Petits objets détectés

Précision Contours

Compréhension Contextuelle

Vitesse

FPN

2017 - Détection d'Objets

COMPRÉHENSION MULTI-ÉCHELLE



Fusion pyramidale top-down

- Objets multi-tailles
- Contexte global
- Scènes complexes
- Variations d'échelle

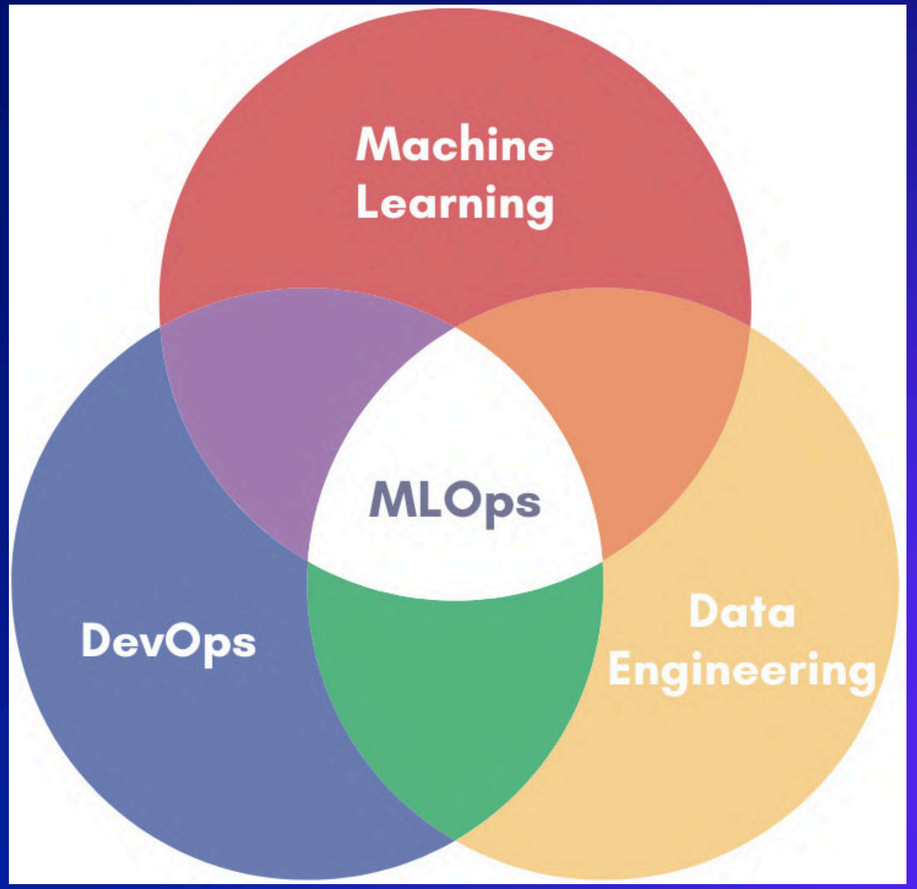
Précision Contours

Compréhension Contextuelle

Vitesse

TRAVAIL EN ENVIRONNEMENT MLOPS

PRINCIPES



Aspect	Machine Learning	Data Engineering	DevOps	MLOps
But principal	Construire et entraîner un modèle	Préparer et transformer les données	Automatiser et fiabiliser les déploiements	Automatiser tout le cycle ML (de la donnée à la production)
Activités clés	Modélisation, entraînement, évaluation	Collecte, nettoyage, feature engineering, pipelines	Gestion du code, CI/CD, monitoring, infrastructure	CI/CD ML, gestion des modèles, surveillance, reproductibilité
Outils courants	TensorFlow, PyTorch, scikit-learn, MLFlow	DVC, Apache Airflow, Kafka, Spark, MinIO	Docker, Kubernetes, Jenkins, Git, Prometheus	MLFlow, Kubeflow, Seldon Core, Airflow, Prometheus
Livrables	Modèle entraîné, métriques de performance	Données prêtes à l'emploi, pipelines automatisés	Code source, containers, scripts de déploiement	Modèles déployés, pipelines d'automatisation, monitoring et alertes
Compétences requises	Mathématiques, statistiques, programmation	Python/SQL, ingénierie des données, architecture cloud	DevOps, scripting, cloud, sécurité	Connaissances en ML, DevOps et data engineering
Cycle de vie	Expérimentation et optimisation	Automatisation des flux de données	Déploiement et maintenance	Orchestration complète du pipeline ML

TRAVAIL EN ENVIRONNEMENT MLOPS

UTILISATION DE MLFLOW

Grands principes :

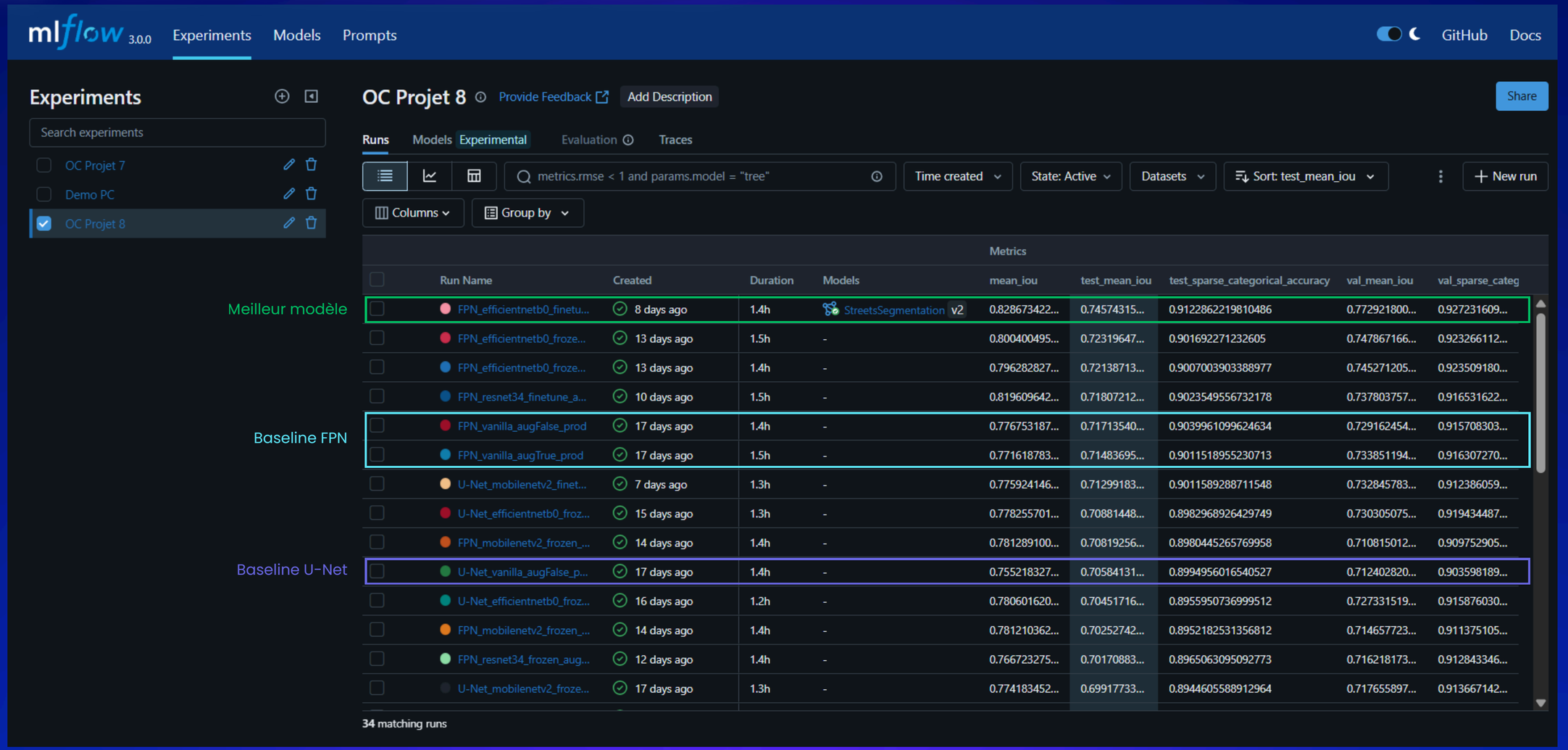
- Suivi des expériences (params, métriques, artefacts)
- Comparaison & reproductibilité des runs
- Stockage artefacts via buckets S3
- Packaging & déploiement modèles
- Interface web simple & efficace

Déploiement:

- Docker sur NAS (OpenMediaVault)
- Backend artefacts : MinIO local (S3-like)
- Reverse proxy via SWAG (Nginx + SSL Let’s Encrypt)
- Accès réseau local + public sécurisé
- Suivi & gestion centralisée des modèles

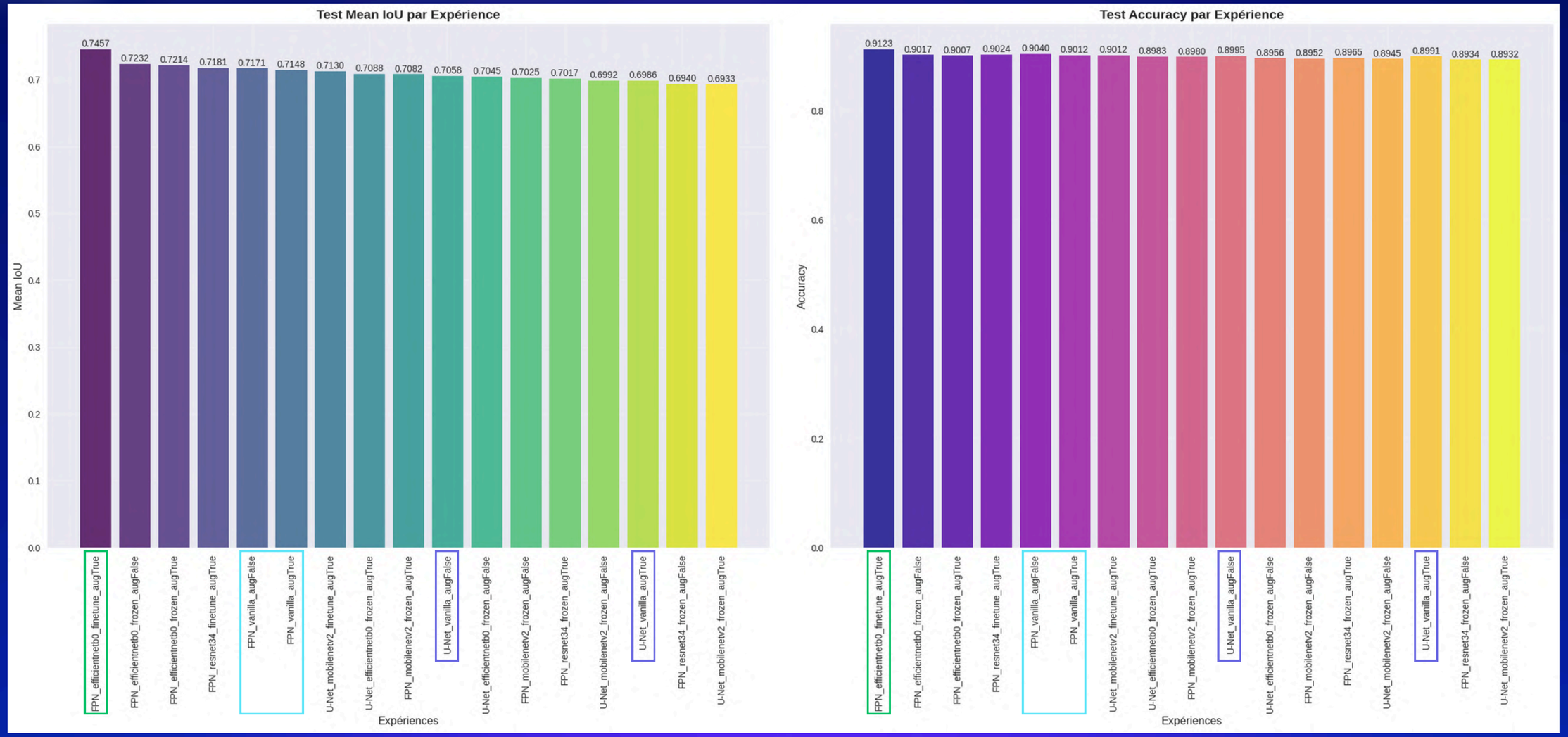
RÉSULTATS

SUIVI D'EXPÉRIENCES MLFLOW



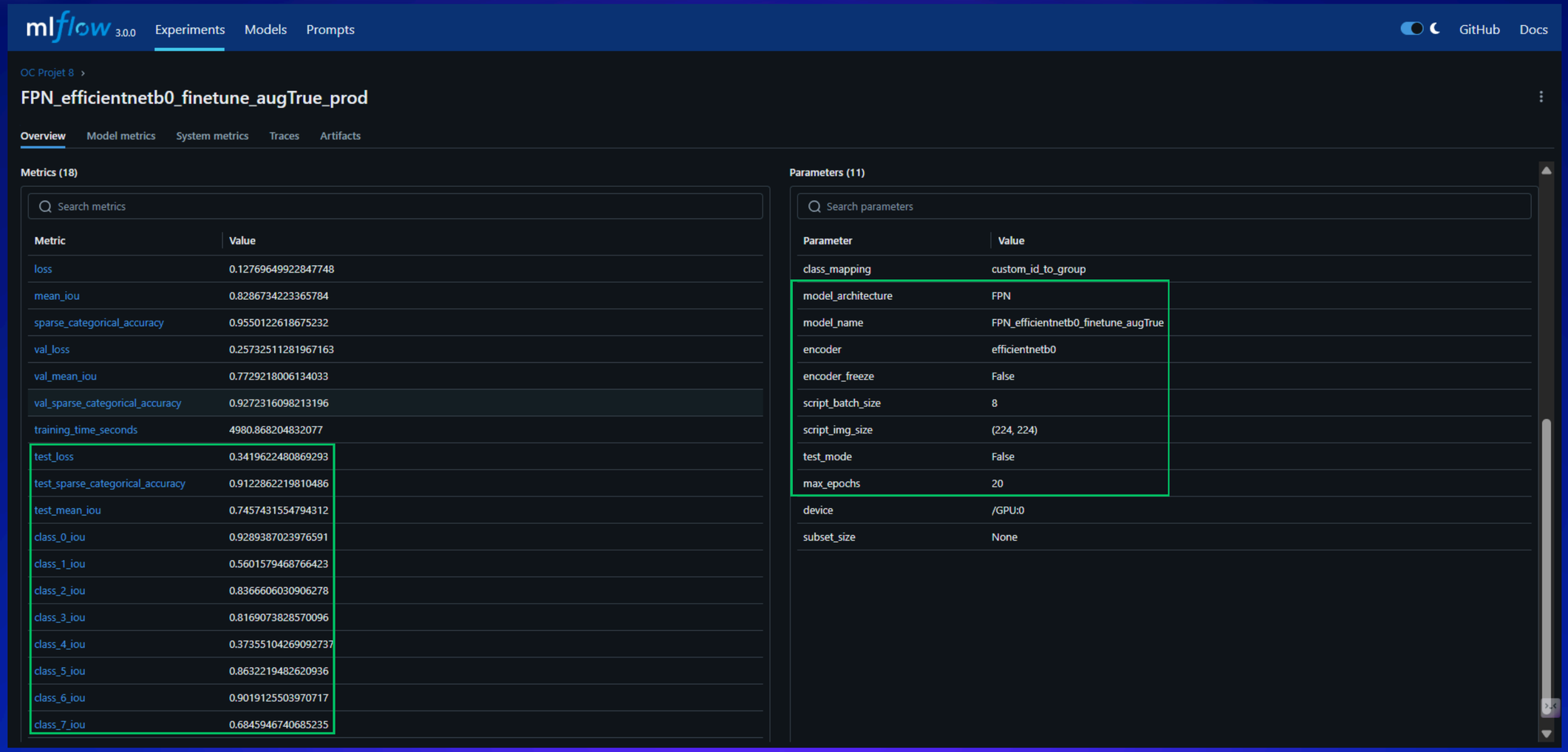
RÉSULTATS

SUIVI D'EXPÉRIENCES MLFLOW



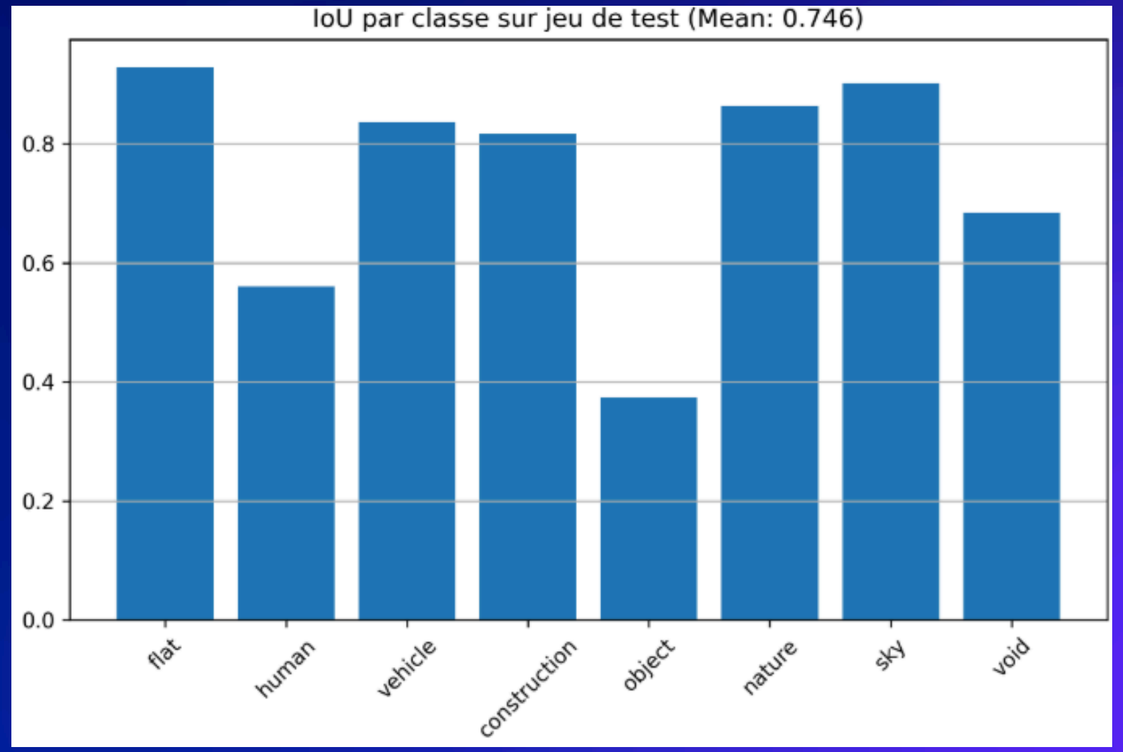
RÉSULTATS

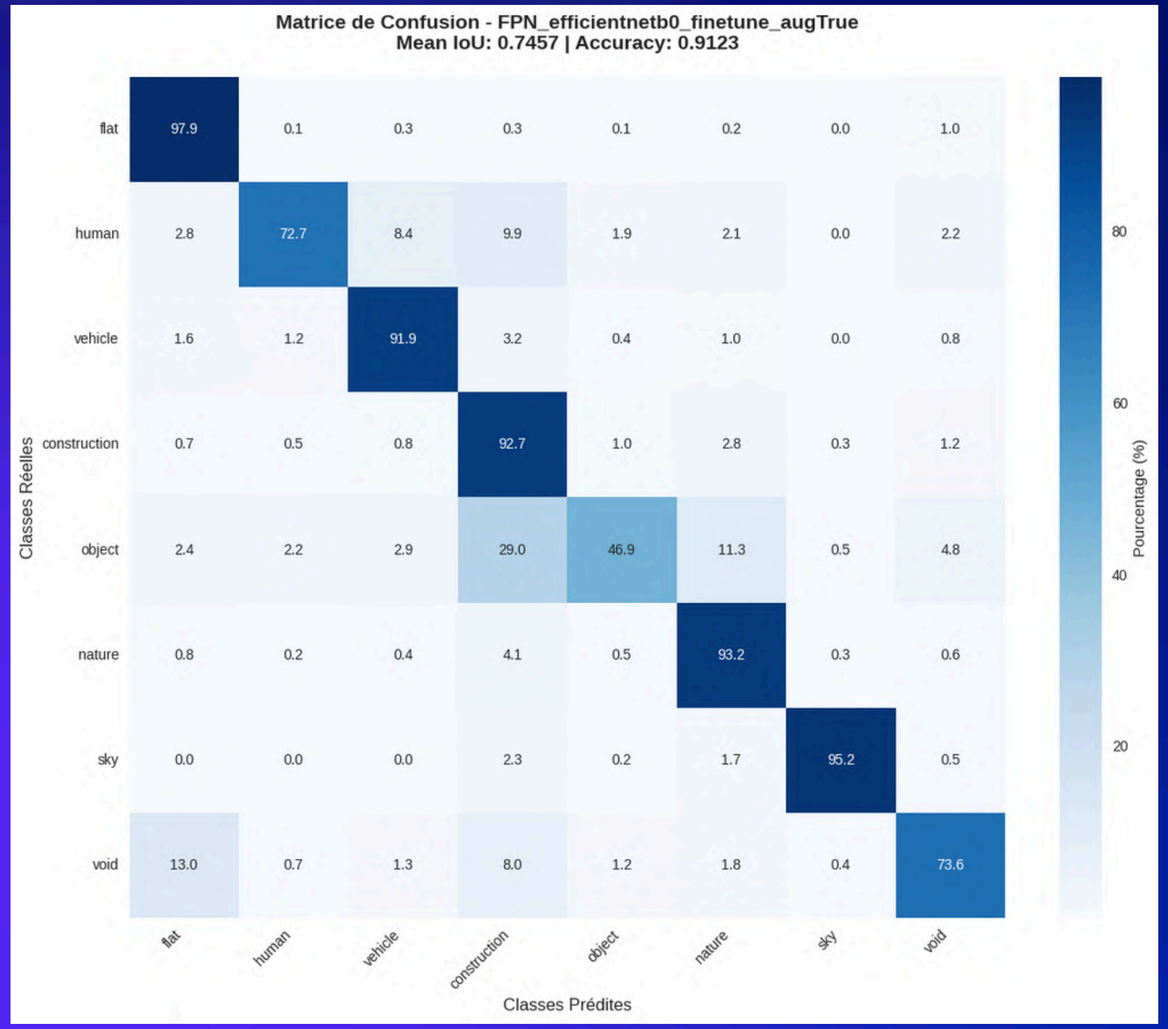
SUIVI D'EXPÉRIENCES MLFLOW : MEILLEUR MODÈLE



RÉSULTATS

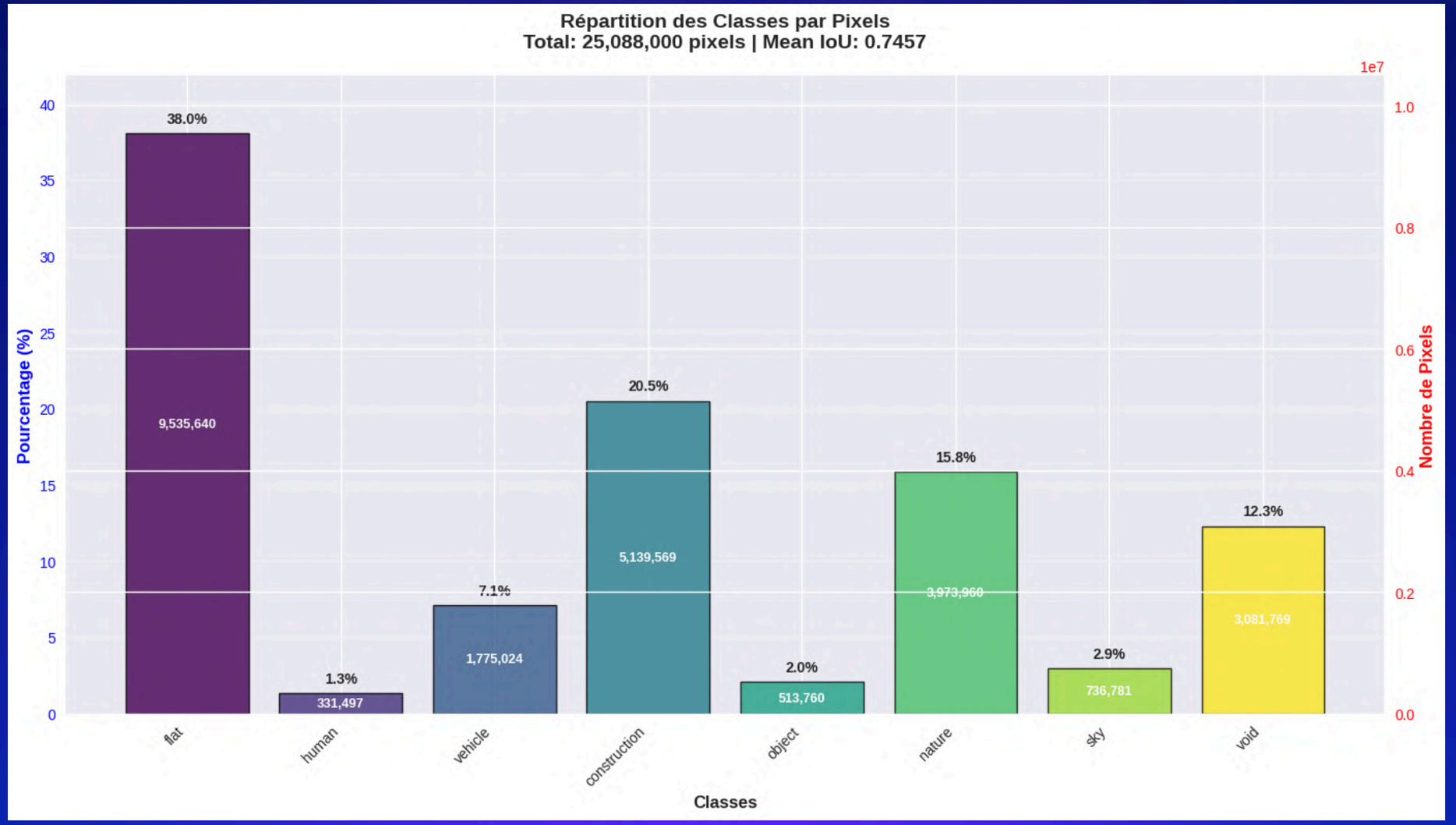
SUIVI D'EXPÉRIENCES MLFLOW : MEILLEUR MODÈLE





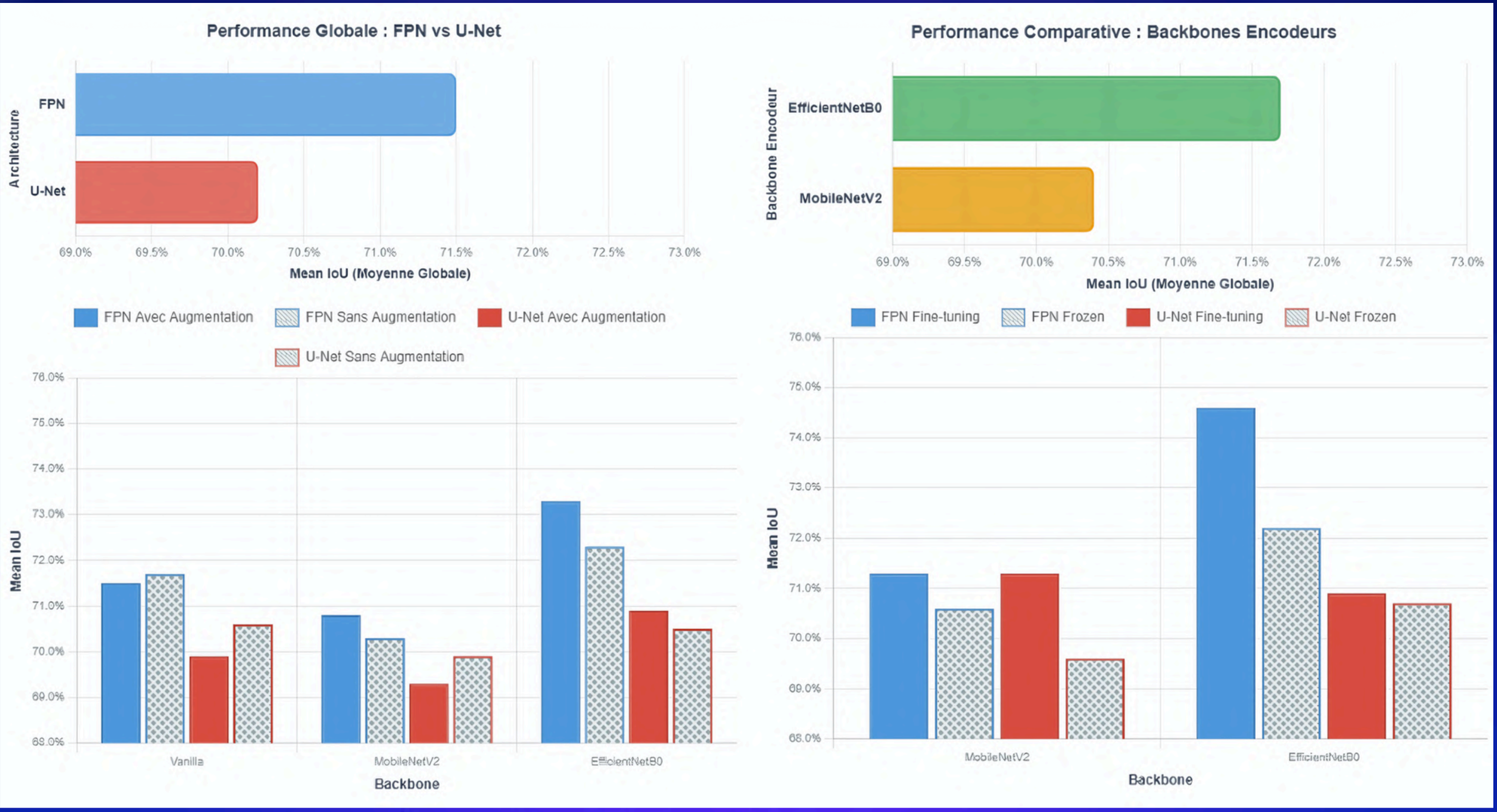
RÉSULTATS

SUIVI D'EXPÉRIENCES MLFLOW : RÉPARTITION CLASSES DATASET TEST



RÉSULTATS

SUIVI D'EXPÉRIENCES MLFLOW : HYPERPARAMETERS TUNING

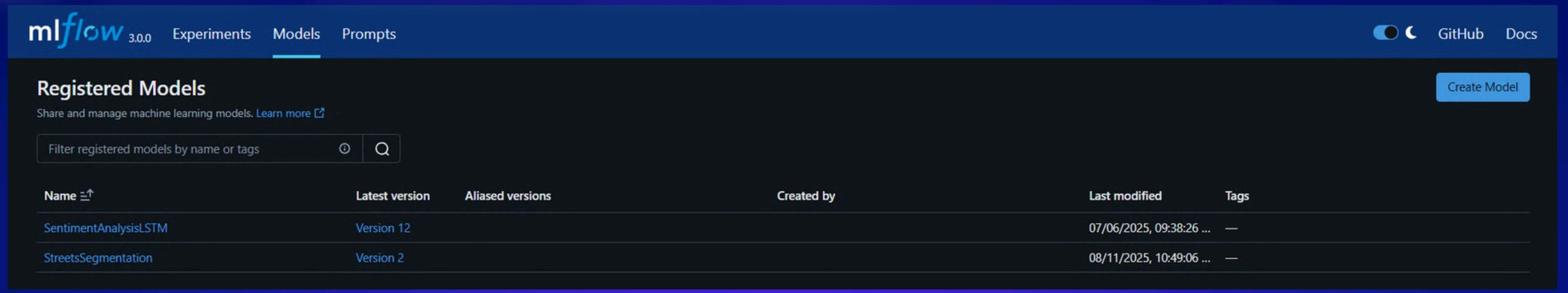


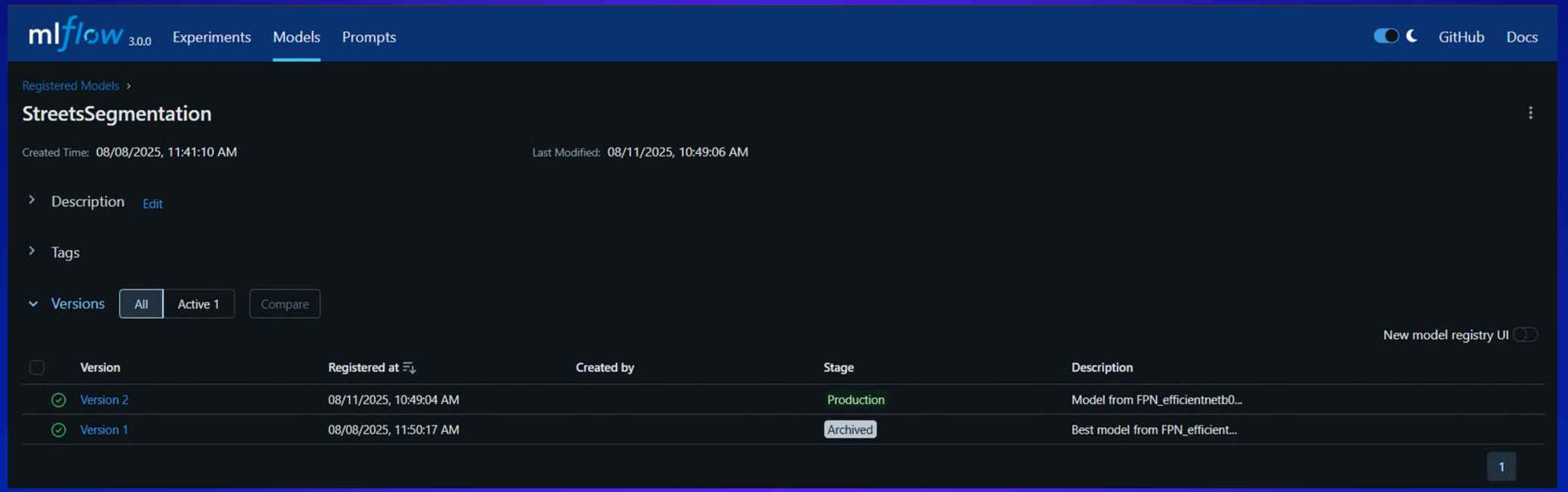


MISE EN PRODUCTION

DÉVELOPPEMENT API

MLFLOW : MEILLEUR MODÈLE AVANCÉ





DÉVELOPPEMENT API

CONSTRUCTION DE STREET VISION API

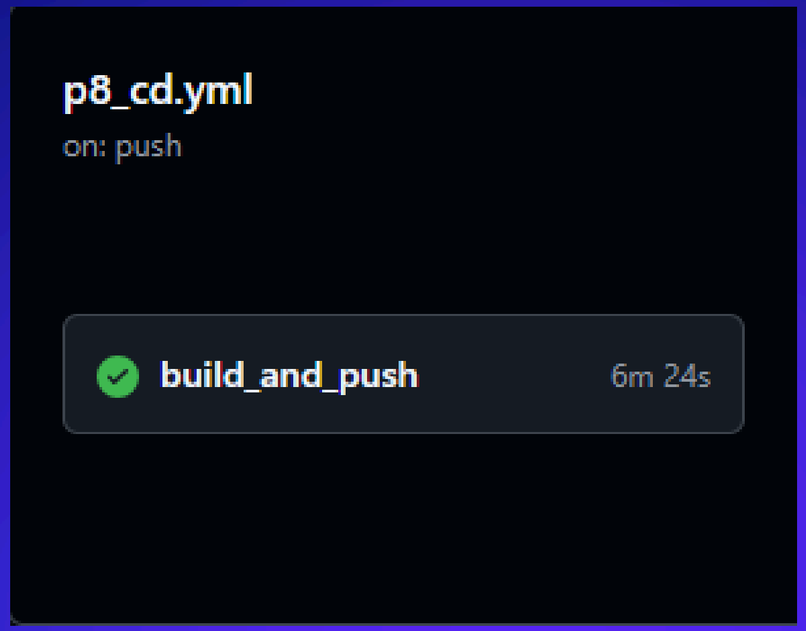
- FastAPI :
  - Requête modèle le plus récent MLFlow ("StreetsSegmentation") + téléchargement + versioning modèles
  - Gestion de plusieurs requêtes de segmentation
  - Entrées : Dataset Cityscapes ou images uploadées utilisateur
  - Sorties : Masques de segmentation + statistiques d'analyse
- Streamlit :
  - Interface graphique pour upload/sélection images
  - Échange de requêtes vers FastAPI
  - Visualisation masques + statistiques

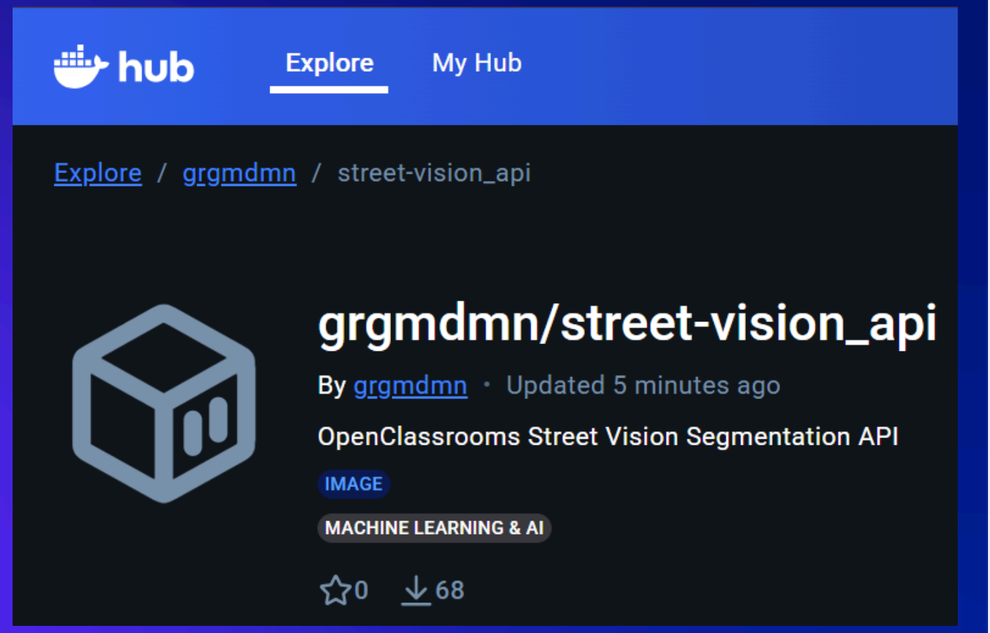
- NGinx reverse proxy : gestion des 2 services sur un seul port

DÉPLOIEMENT CONTINU

PIPELINE

- Dockerisation de l'application
- Github workflow CD :
  - Conteneurisation automatique
  - push vers DockerHub

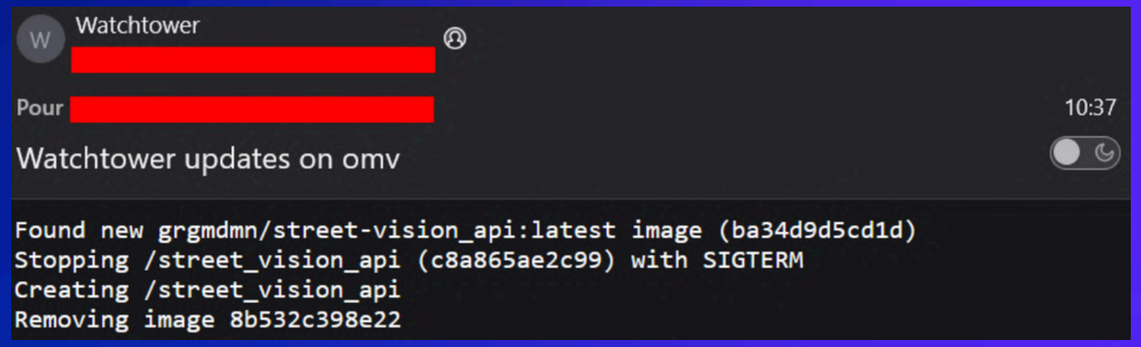




DÉPLOIEMENT CONTINU

DÉPLOIEMENT

- Déploiement sur NAS:
  - Installation via Docker-Compose (configuration yaml)
  - Surveillance et mise à jour via Watchtower



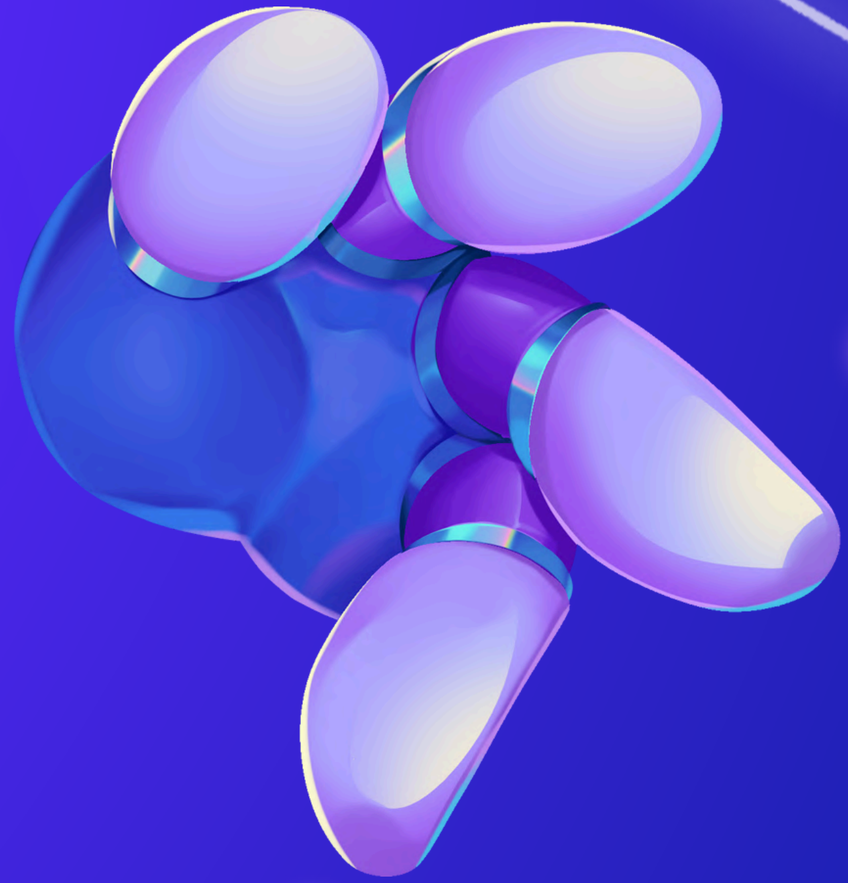
Nom	Description	Etat
MinIO	Creating S3 buckets	Up
MLFlow	MLOps for Machine Learning (OpenClassRooms Project 7)	Up
Nextcloud	Cloud Drive App	Up
Portainer	docker containers catalog	Down
PostgreSQL	Relational Database	Up
Sentiment_API	Tweets Sentiment Prediction : uses Word Embedding + LSTM deep learning	Up
Street Vision API	Multi-segmentation model for street computer vision	Up
SWAG	Secure Web Application Gateway	Up
watchtower	Dockers Updates Monitoring	Up

0 sélectionné / 9 total

DÉMONSTRATION API



RESPECT RGPD



RESPECT RGPD

Contexte : déploiement sur NAS (serveur personnel / petit serveur d’entreprise)

Principe RGPD	Mesures mises en place
Licéité, loyauté, transparence	API publique, pas d'info perso collectée, usage clair segmentation
Limitation des finalités	Segmentation sémantique uniquement, pas d'autre usage
Minimisation des données	Aucune donnée utilisateur stockée, traitement images temporaire
Exactitude des données	Traitement temps réel, pas de stockage persistant données entrée
Sécurité et confidentialité	SSL (Let's Encrypt), NAS sécurisé, suppression immédiate images

CONCLUSION ET PERSPECTIVES



CONCLUSION

- 🏆 Travail effectué :
  - Étude comparative 17 expérimentations (U-Net vs FPN × backbones)
  - Approche MLOps + déploiement complet API/interface

- 🎯 Meilleur modèle obtenu :
  - FPN + EfficientNetB0 + Fine-tuning + Augmentation
  - Mean IoU 0.746 - compromis optimal précision/efficacité

- ⚠️ Limites identifiées :
  - Classes critiques (Mean IoU) : Human (0.56), Object (0.37)
  - Contours ratés : extrémités humaines (cheveux, doigts) - critique sécurité
  - Sous-représentation + complexité formes Object et Human
  - Plateau performance : limitation dataset, pas forcément architecture

PERSPECTIVES

- 🔧 Optimisations techniques :
  - Approches hybrides :
    - FPN (contexte global) + U-Net (apprentissage sur erreurs contours FPN)
    - FPN + Post-processing Edge-aware filtering frontières critiques (CRF + Guided Filtering)
  - Loss hybride : Mean IoU global + terme IoU Human prioritaire

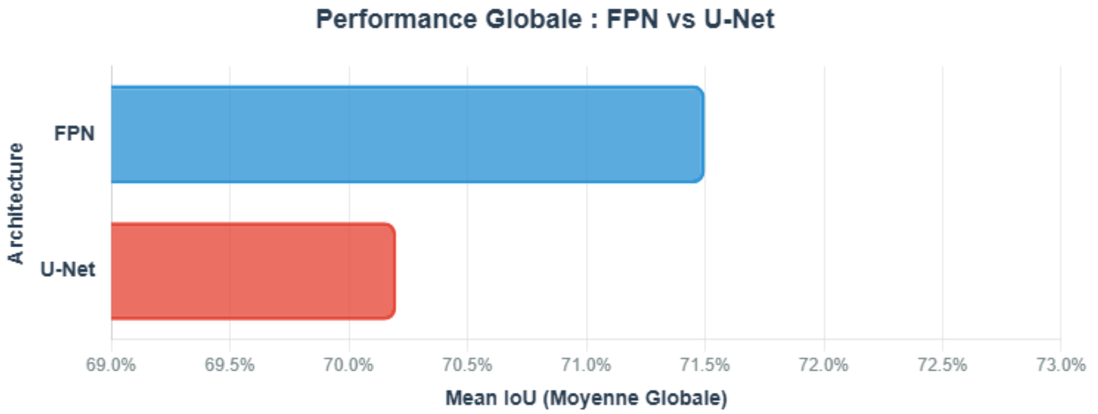
- 📊 Données :
  - Data augmentation ciblée : Classe Human prioritaire
  - Extension dataset : Mapillary Vistas pour diversité (25 000 images classifiées)
  - Rééquilibrage classes : Weighted loss + oversampling classes minoritaires

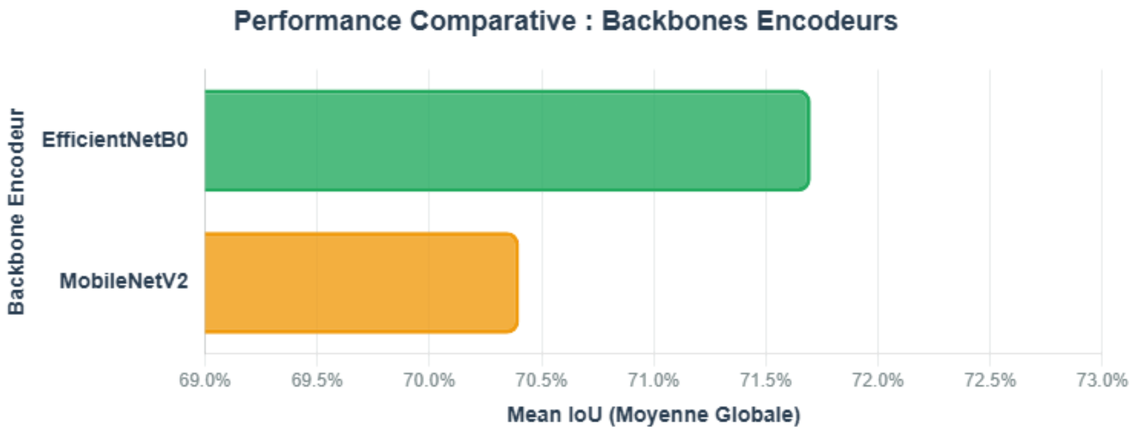
- 🚗 Temps réel embarqué :
  - Quantization : FP16/INT8 pour gains vitesse
  - Hardware spécialisé : Tests GPUs embarqués (Jetson)
  - Latence cible : <100ms décisions critiques

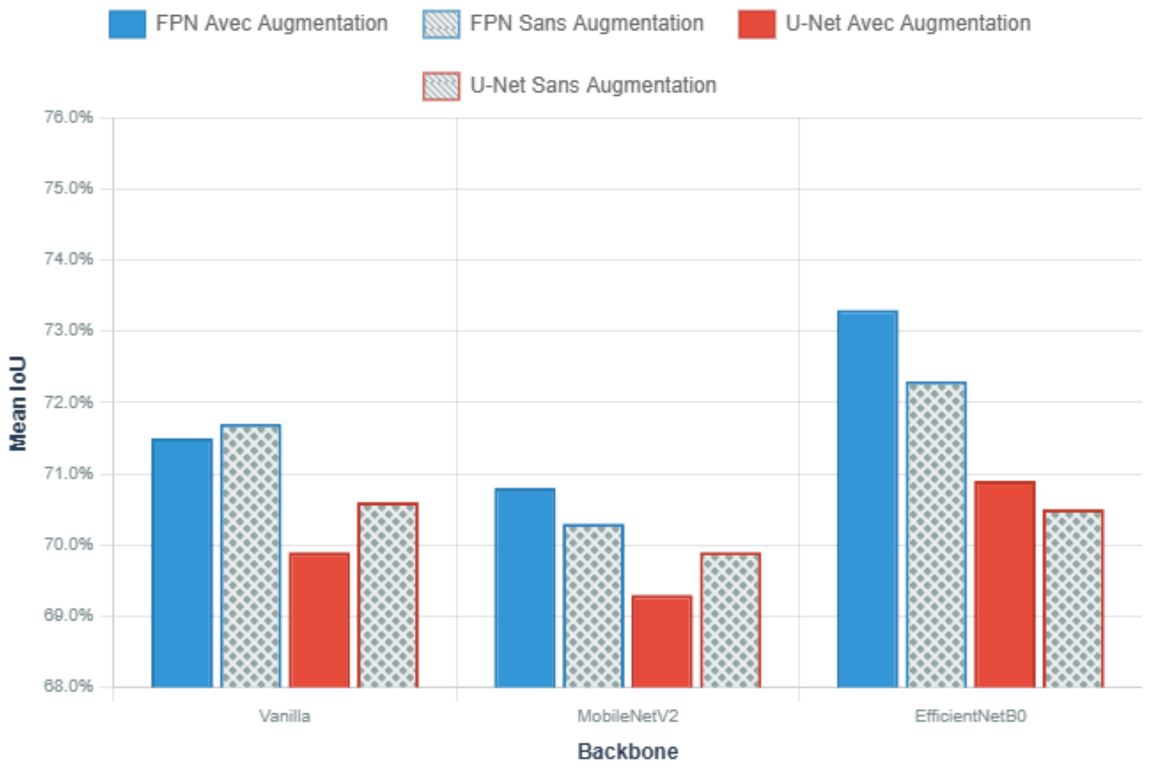
- 🎯 Priorité absolue : Amélioration IoU Human - extrémités ratées = danger mortel

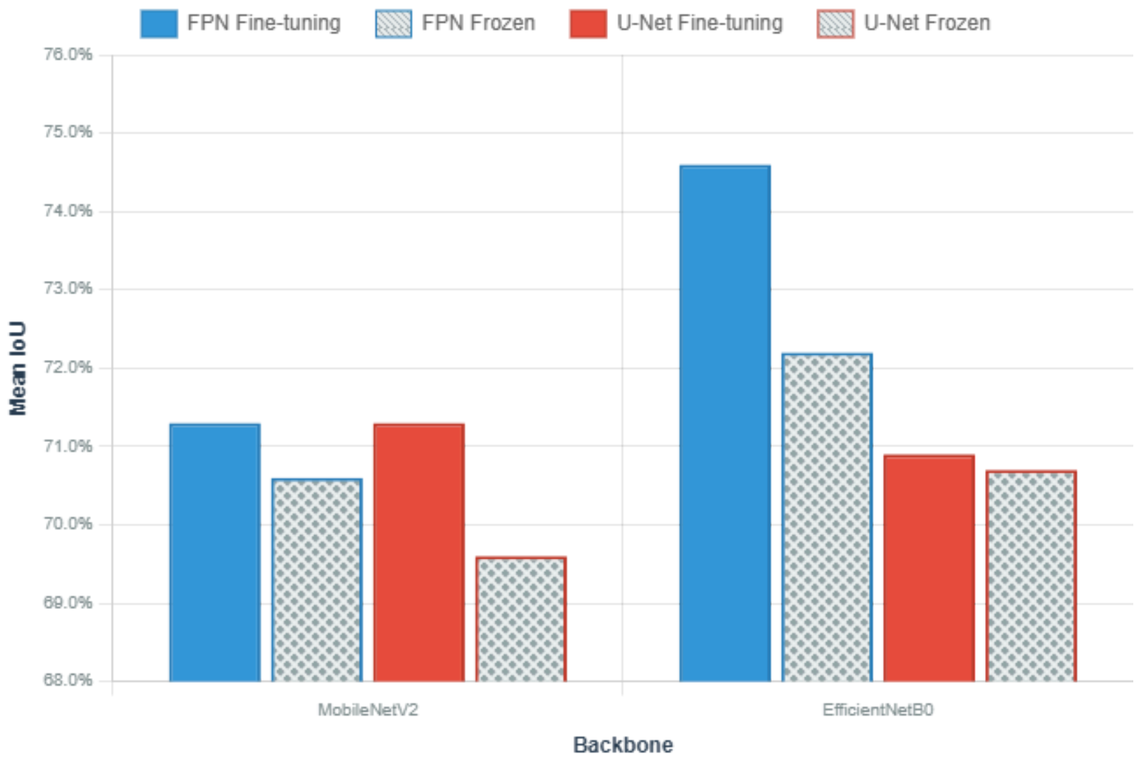
THANK YOU!

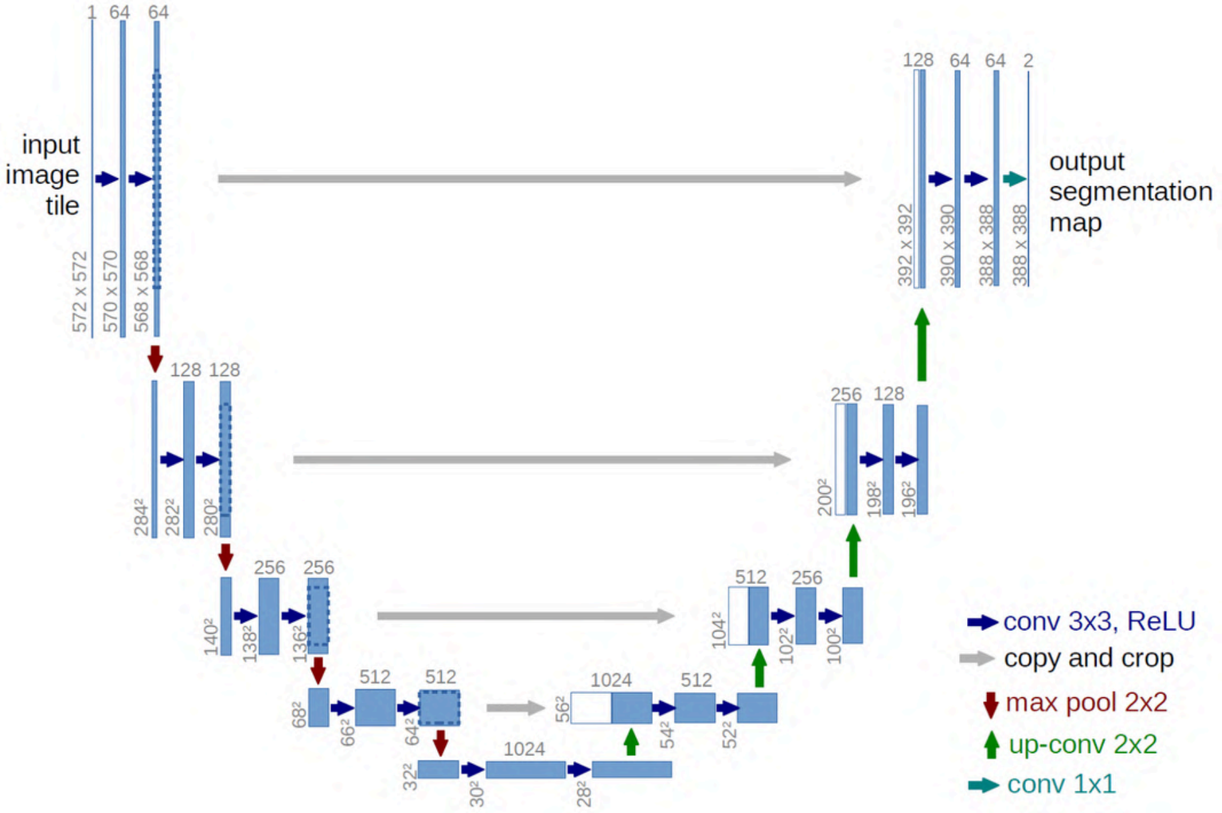


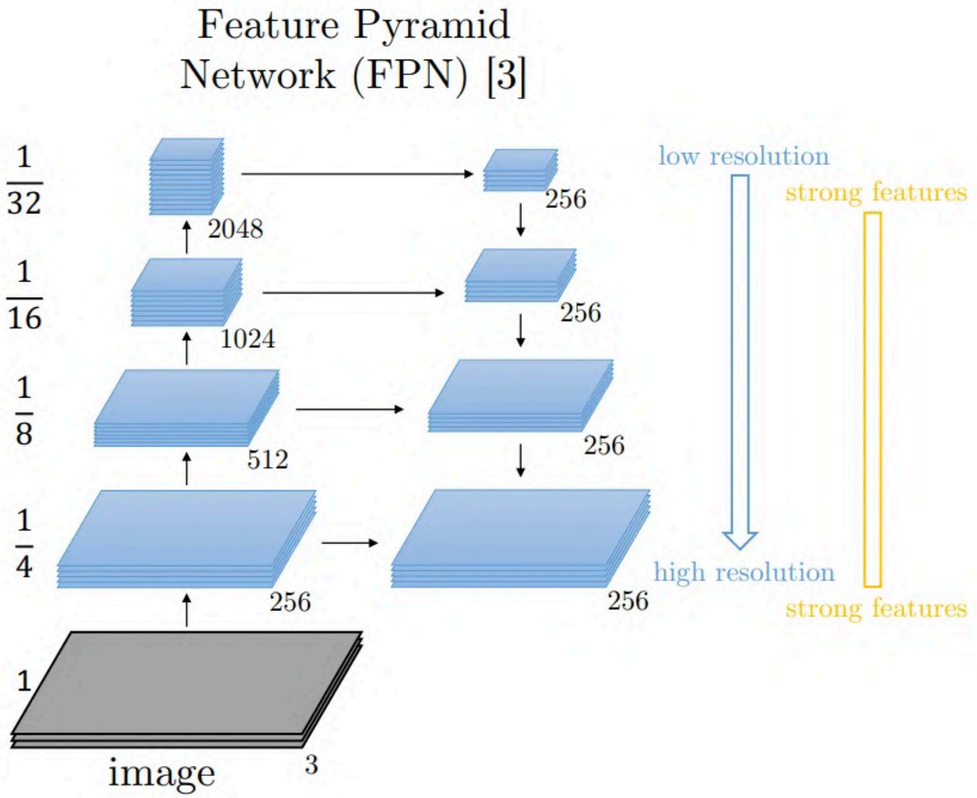

Performance Globale : FPN vs U-Net
Architecture
FPN
U-Net
69.0%
69.5%
70.0%
70.5%
71.0%
71.5%
72.0%
72.5%
73.0%
Mean IoU (Moyenne Globale)


Performance Comparative : Backbones Encodeurs
Backbone Encodeur
EfficientNetB0
MobileNetV2
69.0%
69.5%
70.0%
70.5%
71.0%
71.5%
72.0%
72.5%
73.0%
Mean IoU (Moyenne Globale)


FPN Avec Augmentation
FPN Sans Augmentation
U-Net Avec Augmentation
U-Net Sans Augmentation
Mean IoU
76.0%
75.0%
74.0%
73.0%
72.0%
71.0%
70.0%
69.0%
68.0%
Vanilla
MobileNetV2
EfficientNetB0
Backbone


FPN Fine-tuning
FPN Frozen
U-Net Fine-tuning
U-Net Frozen
Mean IoU
76.0%
75.0%
74.0%
73.0%
72.0%
71.0%
70.0%
69.0%
68.0%
MobileNetV2
EfficientNetB0
Backbone


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1


Feature Pyramid Network (FPN) [3]
low resolution
strong features
high resolution
strong features
image