

# FUTURE VISION TRANSPORT

# TRAITEMENT D'IMAGES

# POUR SYSTÈMES

# EMBARQUÉS DE

# VOITURES AUTONOMES

Grégoire MUREAU

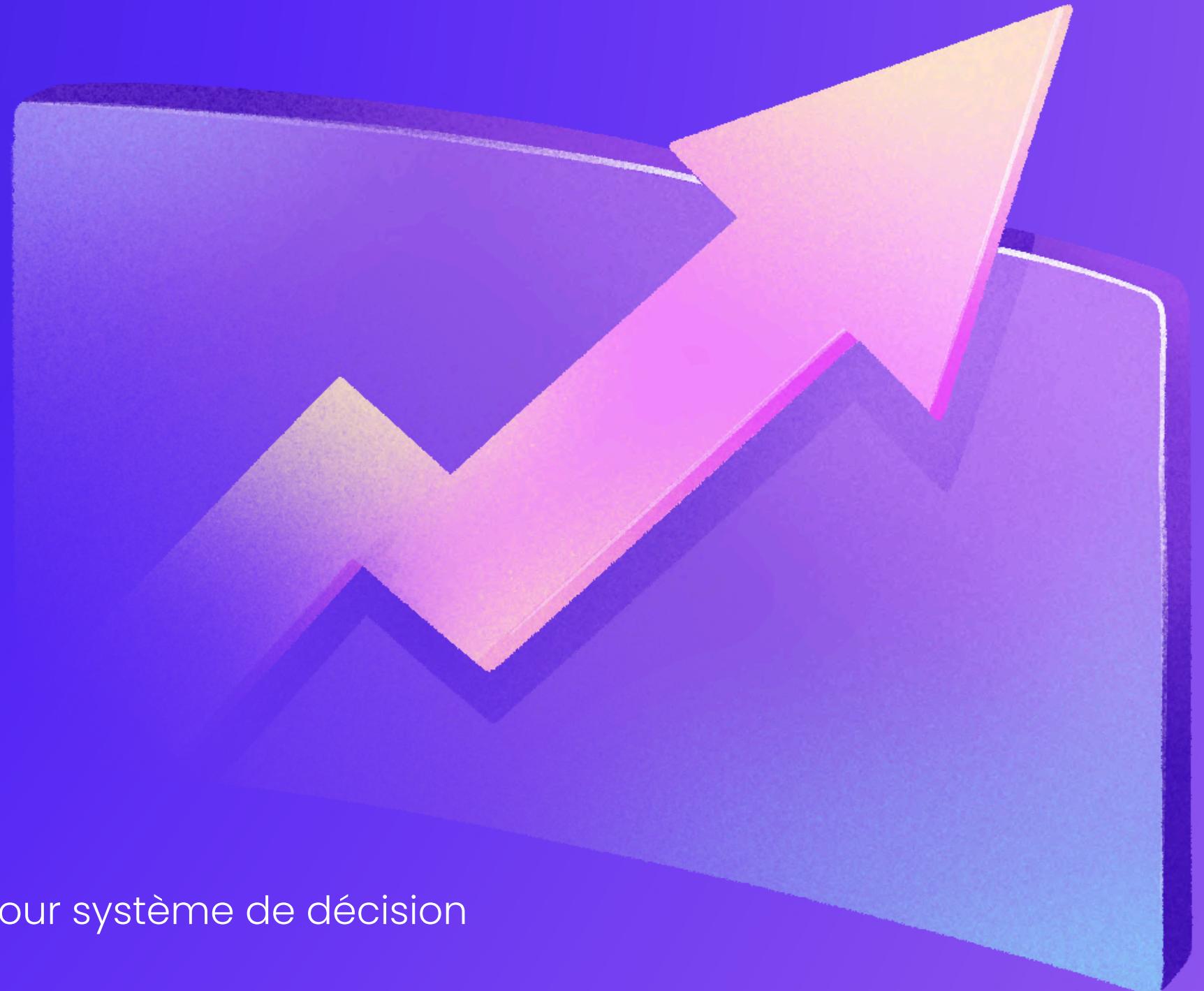
# INTRODUCTION

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



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

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



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

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



## **Déploiement** de l'API



## 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

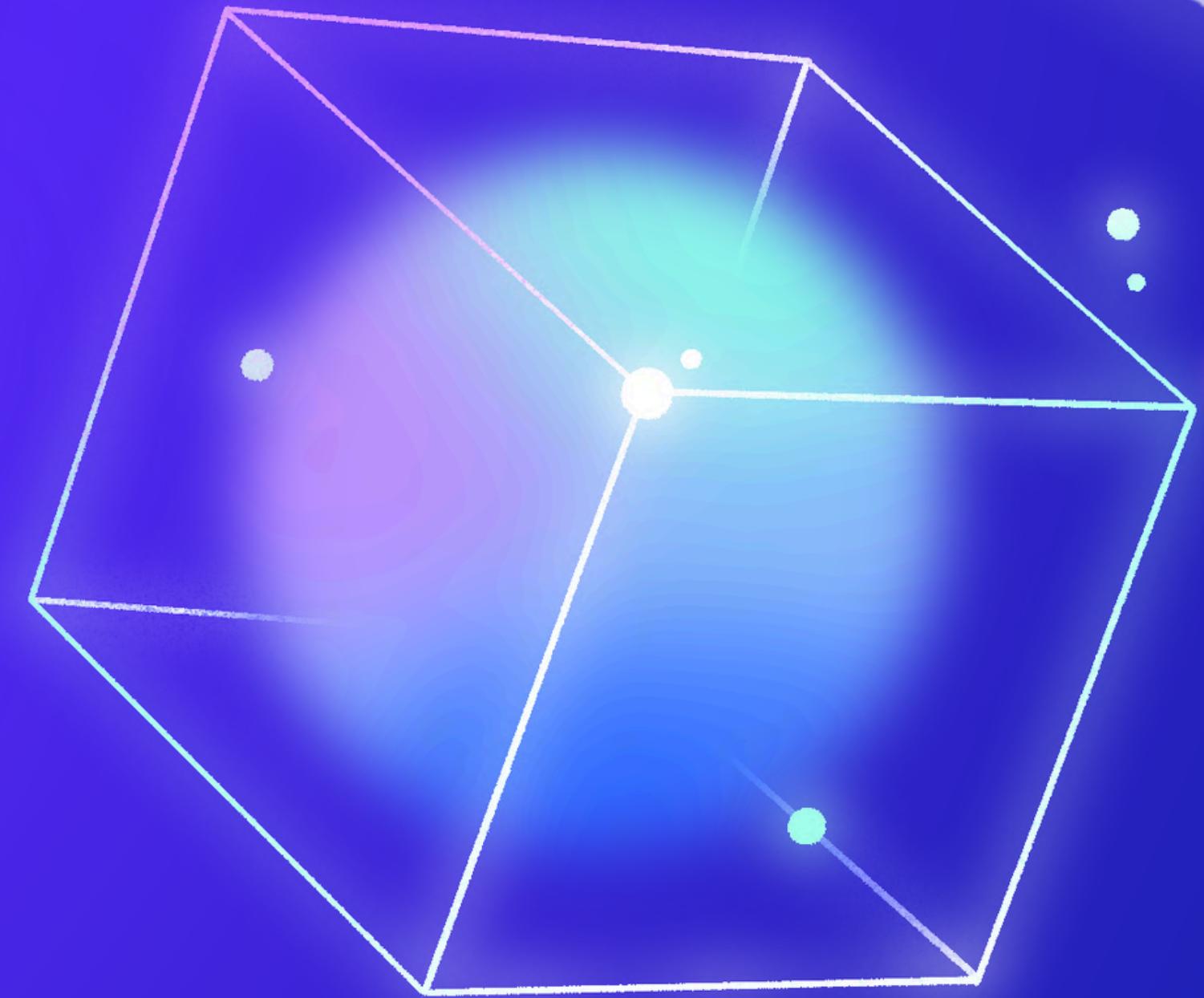
- Développement API de segmentation
- Pipeline CD
- Démonstration (masques, statistiques, comparaisons)

## 3. Règles RGPD

## 4. Conclusion et Limites



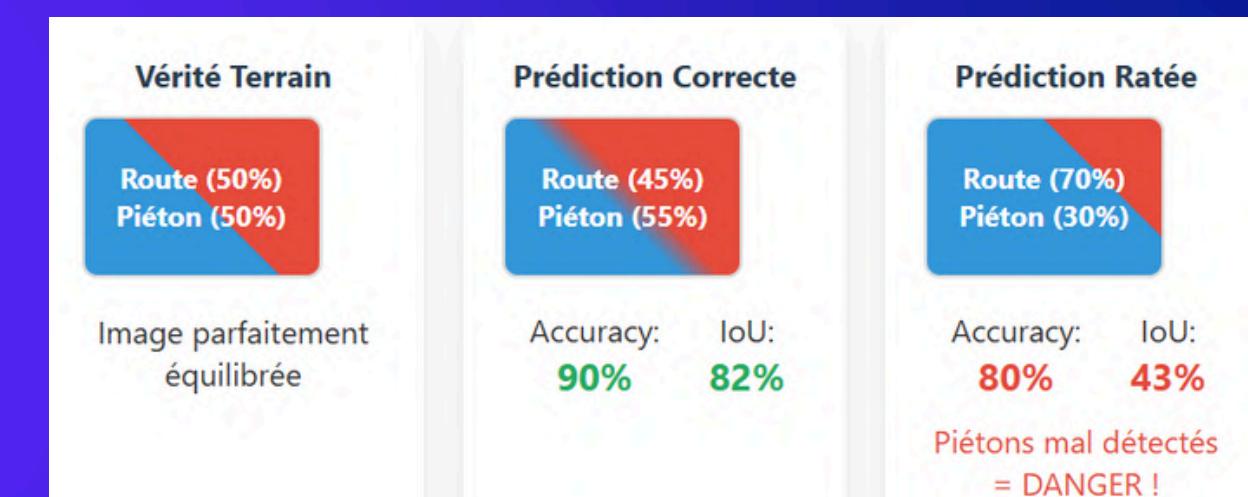
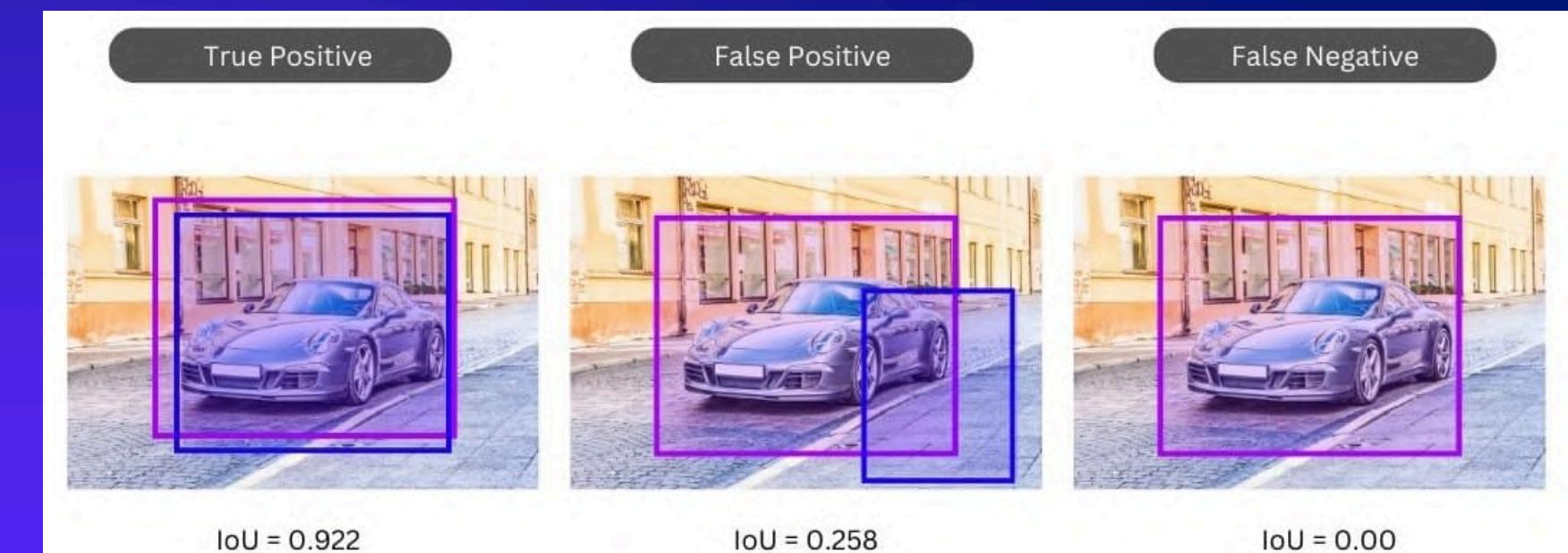
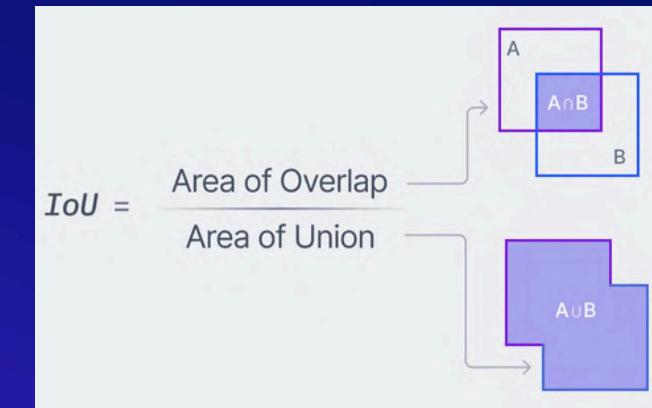
# ENTRAÎNEMENT DE MODÈLES



# 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

## PRÉTRAIEMENT

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

Mapping Cityscapes : 34  $\rightarrow$  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 scratch, 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

## ENTRAÎNEMENT DES MODÈLES: BACKBONES (ENCODEURS)

**VGG16**

**138M**

**BASELINE  
"GROSSE  
PUISANCE"**

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**

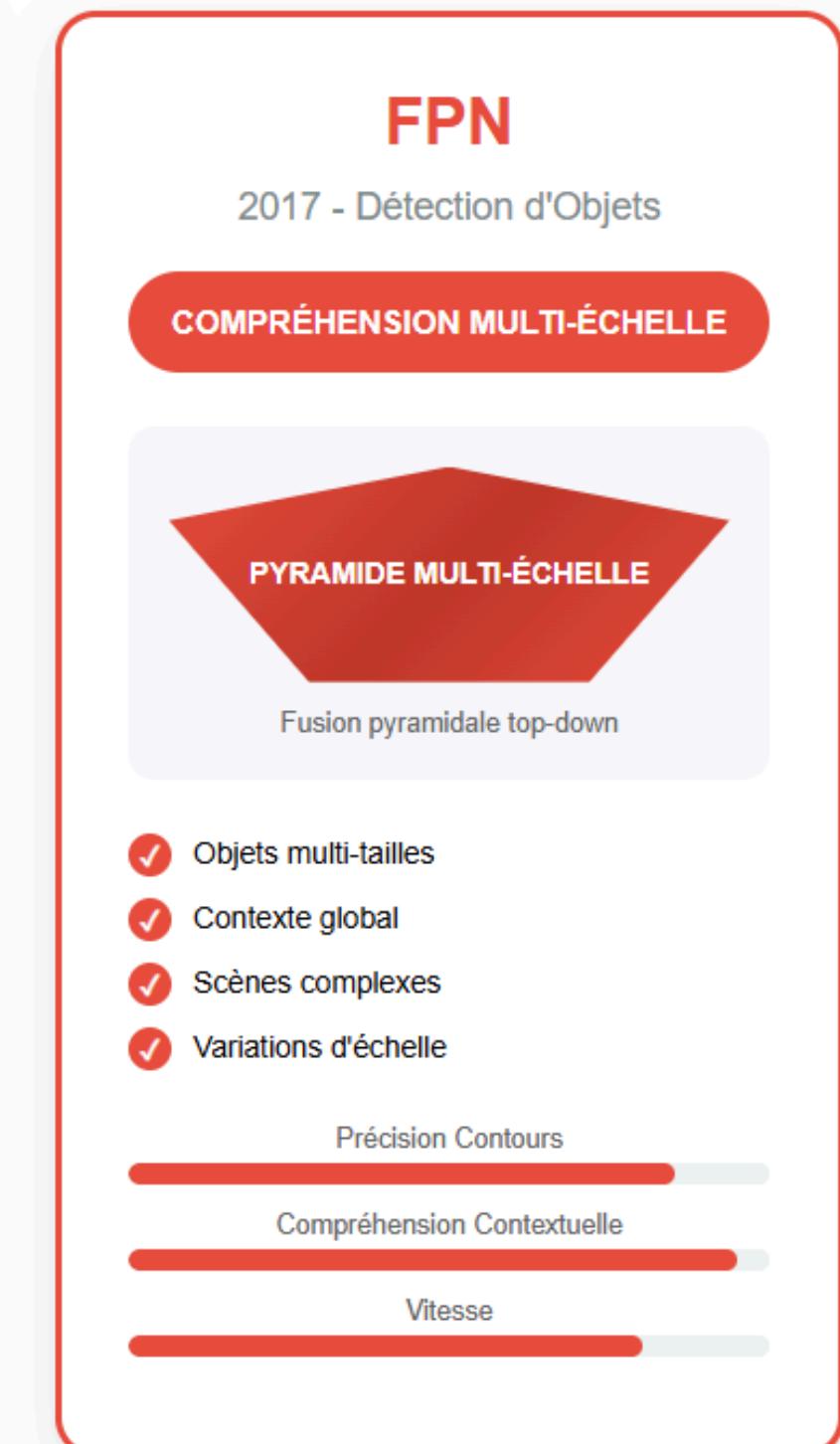
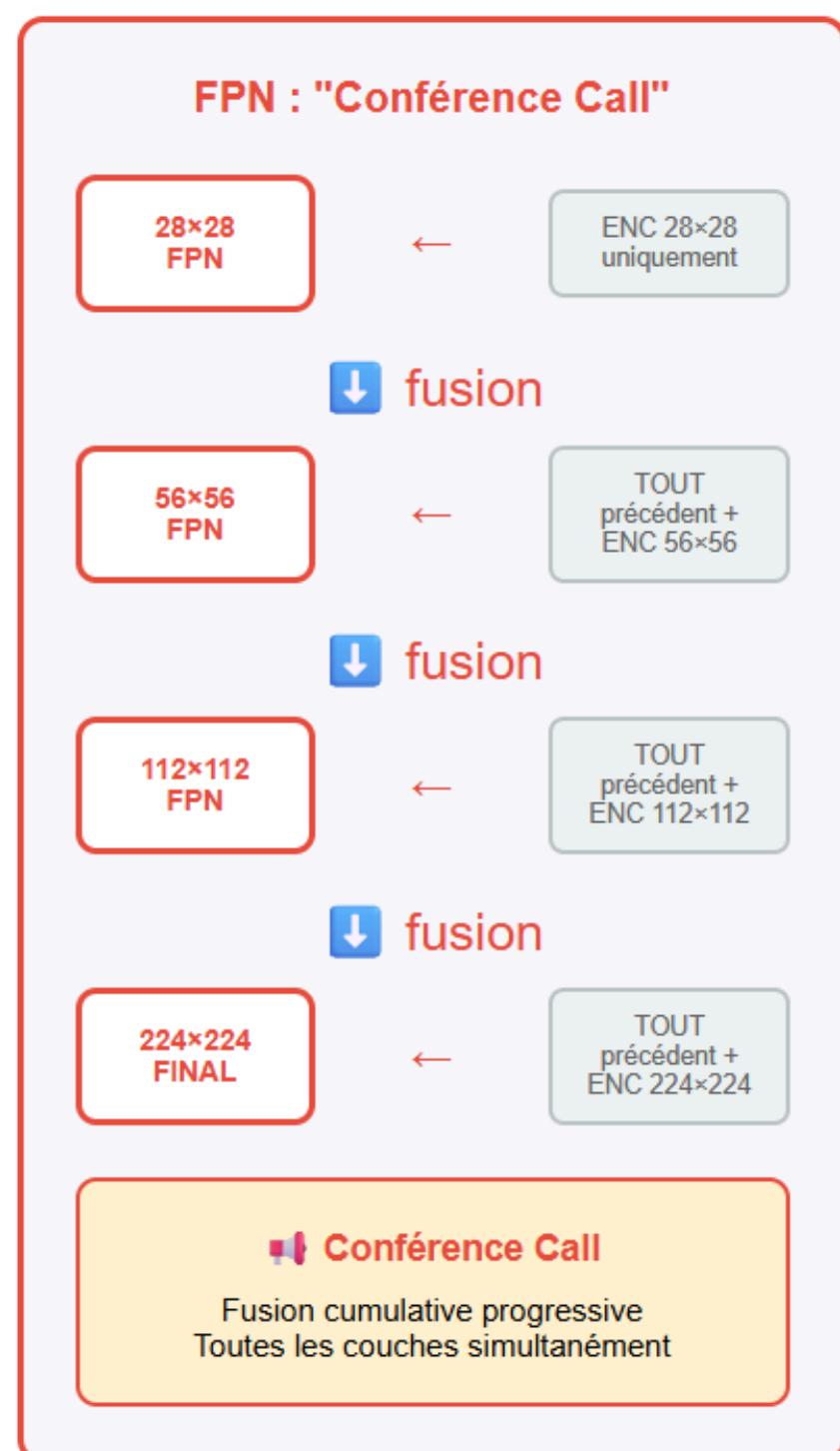
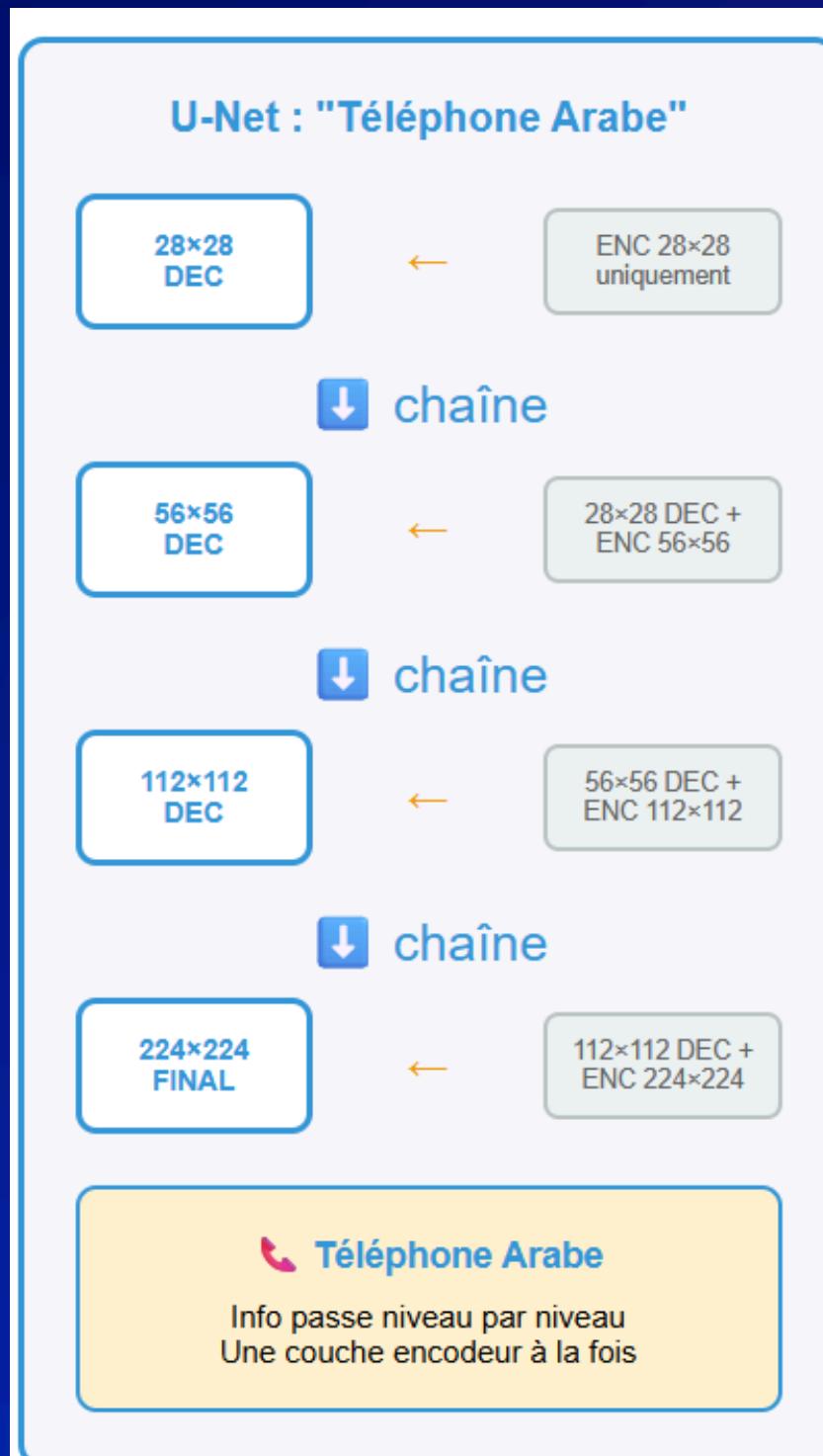
Précision

Efficacité Embarquée

Spécialement conçu  
pour mobile/edge

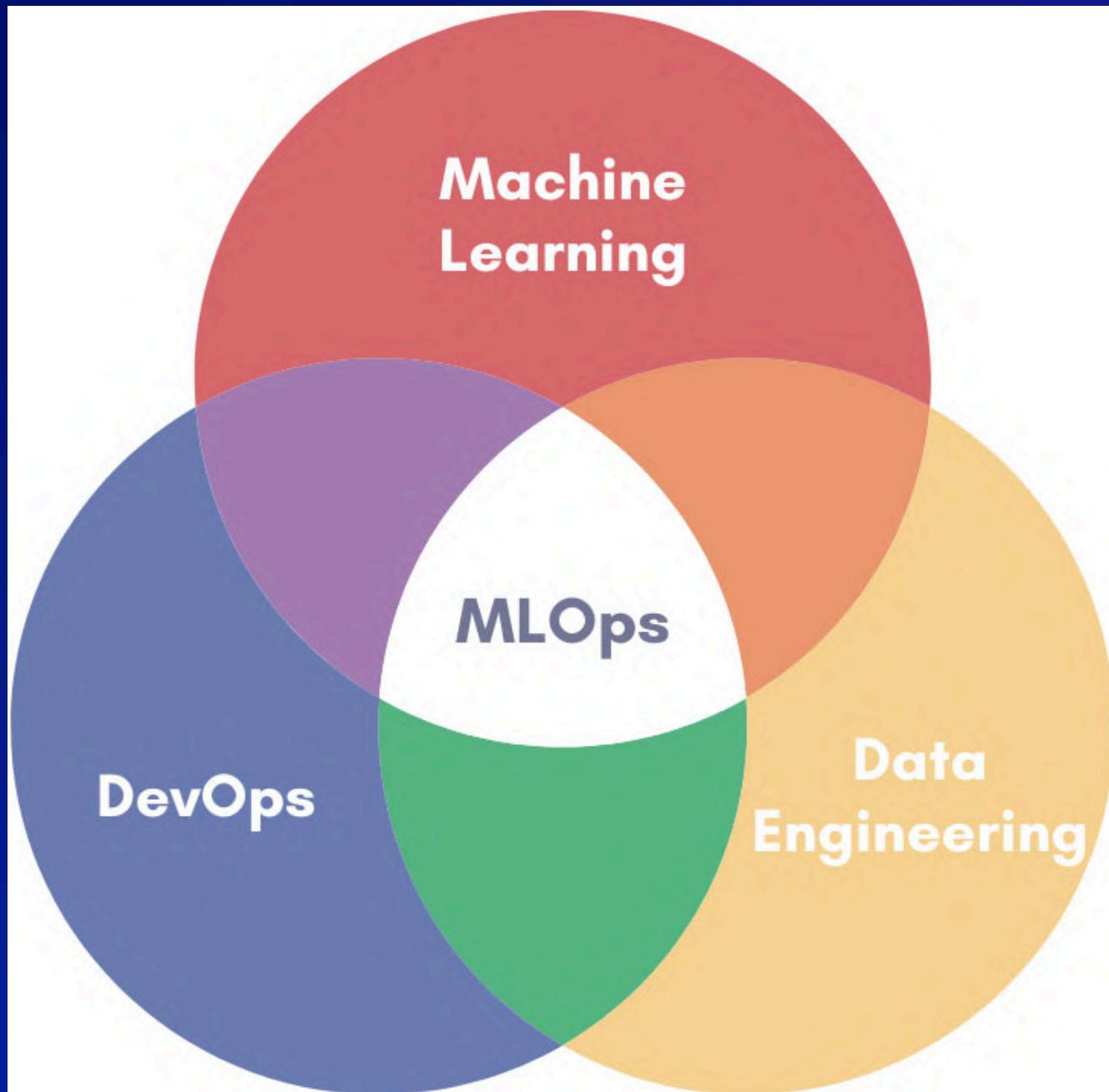
# MÉTHODOLOGIE

## ENTRAÎNEMENT DES MODÈLES: ARCHITECTURES (DÉCODAGE)



# TRAVAIL EN ENVIRONNEMENT MLOPS

## PRINCIPES



Aspect	Machine Learning	Data Engineering	DevOps	MLOps
<b>But principal</b>	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)
<b>Activités clés</b>	Modélisation, entraînement, évaluation	Collecte, nettoyage, feature engineering, pipelines	Gestion du code, CI/CD, monitoring, infrastructure	CI/CD ML, gestion des modèles, surveillance, reproductibilité
<b>Outils courants</b>	TensorFlow, PyTorch, scikit-learn, MLFlow	DVC, Apache Airflow, Kafka, Spark, MinIO	Docker, Kubernetes, Jenkins, Git, Prometheus	MLFlow, Kubeflow, Seldon Core, Airflow, Prometheus
<b>Livrables</b>	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
<b>Compétences requises</b>	Mathématiques, statistiques, programmation	Python/SQL, ingénierie des données, architecture cloud	DevOps, scripting, cloud, sécurité	Connaissances en ML, DevOps et data engineering
<b>Cycle de vie</b>	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

mlflow 3.0.0 Experiments Models Prompts GitHub Docs

### Experiments

OC Projet 8 Provide Feedback Add Description Share

Search experiments

OC Projet 7 Demo PC OC Projet 8

Runs Models Experimental Evaluation Traces

metrics.rmse < 1 and params.model = "tree"

Time created State: Active Datasets Sort: test\_mean\_iou + New run

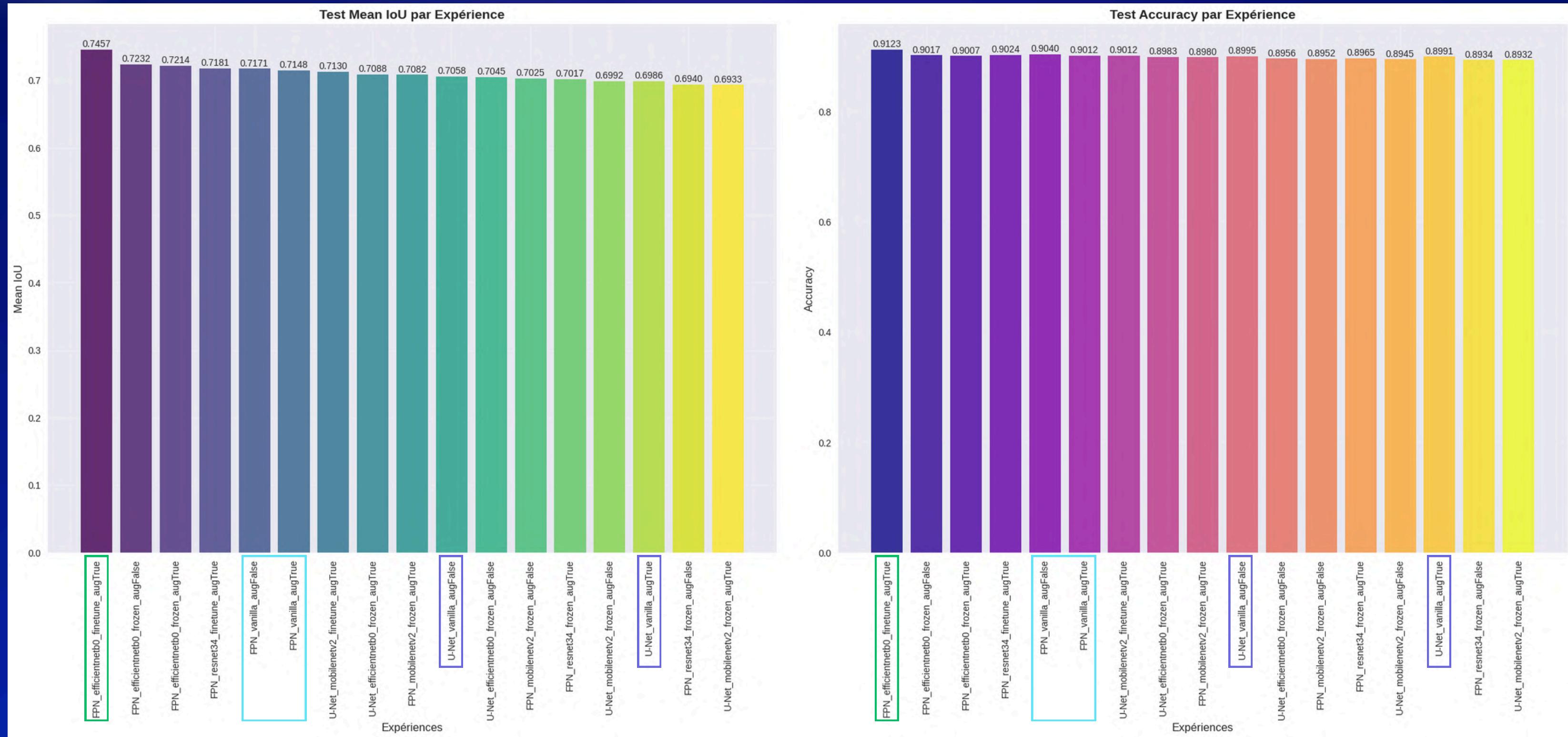
Columns Group by

	Run Name	Created	Duration	Models	mean_iou	test_mean_iou	test_sparse_categorical_accuracy	val_mean_iou	val_sparse_categ
Meilleur modèle	FPN_efficientnetb0_finetune	8 days ago	1.4h	StreetsSegmentation v2	0.828673422...	0.74574315...	0.9122862219810486	0.772921800...	0.927231609...
	FPN_efficientnetb0_froze...	13 days ago	1.5h	-	0.800400495...	0.72319647...	0.901692271232605	0.747867166...	0.923266112...
	FPN_efficientnetb0_froze...	13 days ago	1.4h	-	0.796282827...	0.72138713...	0.9007003903388977	0.745271205...	0.923509180...
	FPN_resnet34_finetune_a...	10 days ago	1.5h	-	0.819609642...	0.71807212...	0.9023549556732178	0.737803757...	0.916531622...
Baseline FPN	FPN_vanilla_augFalse_prod	17 days ago	1.4h	-	0.776753187...	0.71713540...	0.9039961099624634	0.729162454...	0.915708303...
	FPN_vanilla_augTrue_prod	17 days ago	1.5h	-	0.771618783...	0.71483695...	0.9011518955230713	0.733851194...	0.916307270...
	U-Net_mobilenetv2_finet...	7 days ago	1.3h	-	0.775924146...	0.71299183...	0.9011589288711548	0.732845783...	0.912386059...
	U-Net_efficientnetb0_froz...	15 days ago	1.3h	-	0.778255701...	0.70881448...	0.8982968926429749	0.730305075...	0.919434487...
	FPN_mobilenetv2_frozen_...	14 days ago	1.4h	-	0.781289100...	0.70819256...	0.8980445265769958	0.710815012...	0.909752905...
Baseline U-Net	U-Net_vanilla_augFalse_p...	17 days ago	1.4h	-	0.755218327...	0.70584131...	0.8994956016540527	0.712402820...	0.903598189...
	U-Net_efficientnetb0_froz...	16 days ago	1.2h	-	0.780601620...	0.70451716...	0.8955950736999512	0.727331519...	0.915876030...
	FPN_mobilenetv2_frozen_...	14 days ago	1.4h	-	0.781210362...	0.70252742...	0.8952182531356812	0.714657723...	0.911375105...
	FPN_resnet34_frozen_aug...	12 days ago	1.4h	-	0.766723275...	0.70170883...	0.8965063095092773	0.716218173...	0.912843346...
	U-Net_mobilenetv2_froze...	17 days ago	1.3h	-	0.774183452...	0.69917733...	0.8944605588912964	0.717655897...	0.913667142...

34 matching runs

# RÉSULTATS

## SUIVI D'EXPÉRIENCES MLFLOW



# RÉSULTATS

## SUIVI D'EXPÉRIENCES MLFLOW : MEILLEUR MODÈLE

The screenshot shows the MLflow UI interface for an experiment named `FPN_efficientnetb0_finetune_augTrue_prod`. The top navigation bar includes links for `mlflow` 3.0.0, `Experiments`, `Models`, and `Prompts`. On the right, there are GitHub and Docs links.

The main content area displays two tables: `Metrics (18)` and `Parameters (11)`.

**Metrics (18)**

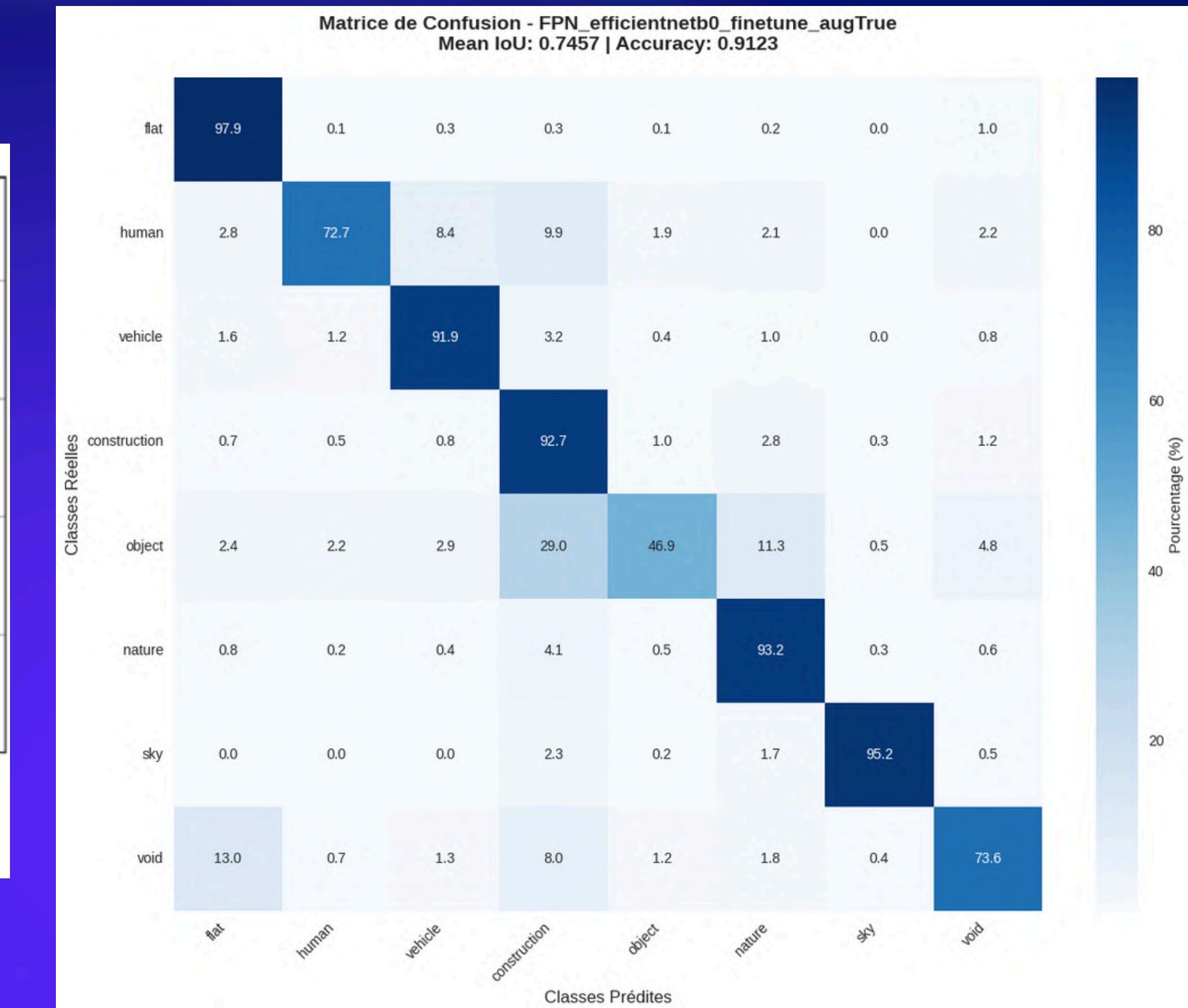
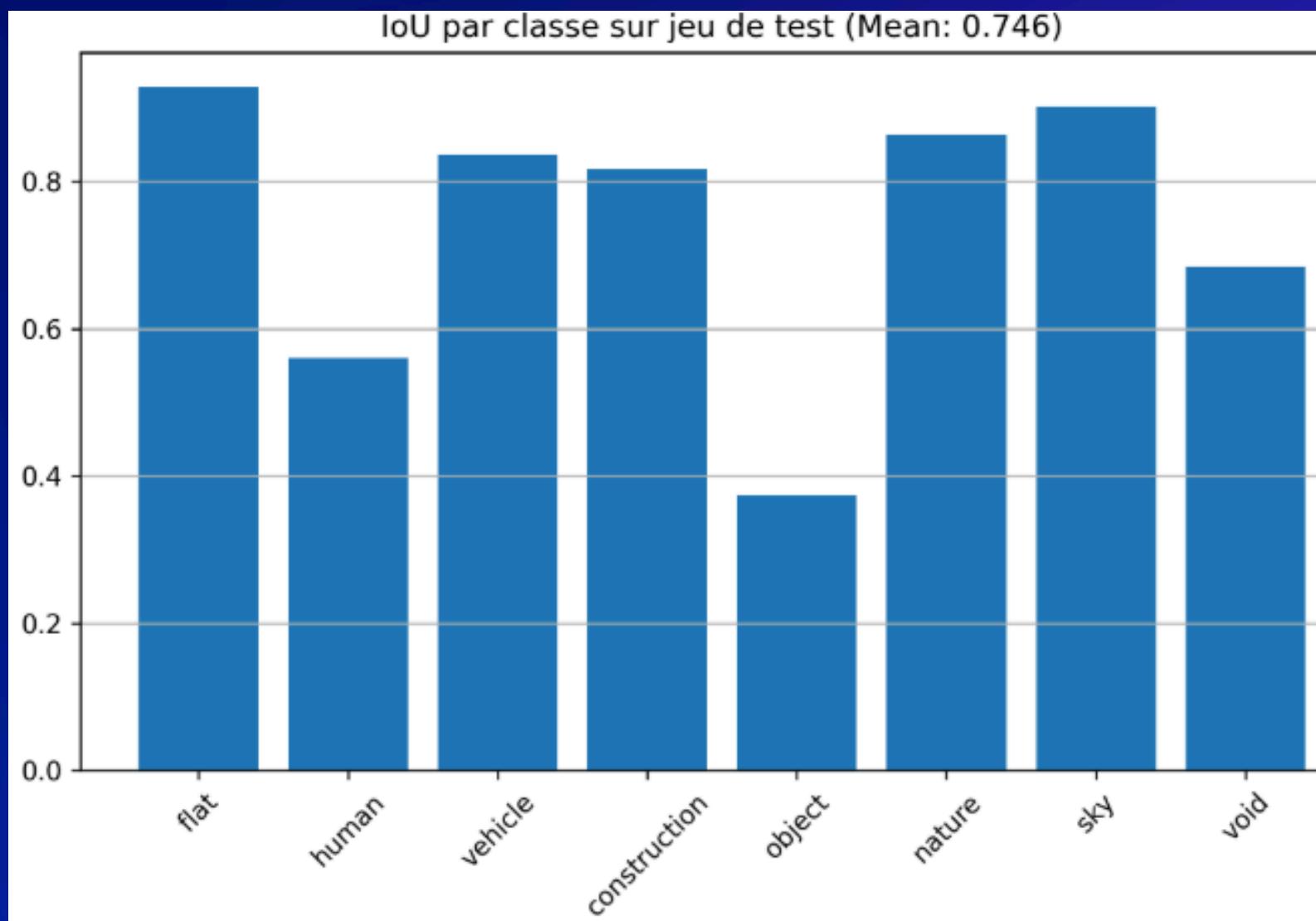
Metric	Value
loss	0.12769649922847748
mean_iou	0.8286734223365784
sparse_categorical_accuracy	0.9550122618675232
val_loss	0.25732511281967163
val_mean_iou	0.7729218006134033
val_sparse_categorical_accuracy	0.9272316098213196
training_time_seconds	4980.868204832077
test_loss	0.3419622480869293
test_sparse_categorical_accuracy	0.9122862219810486
test_mean_iou	0.7457431554794312
class_0_iou	0.9289387023976591
class_1_iou	0.5601579468766423
class_2_iou	0.8366606030906278
class_3_iou	0.8169073828570096
class_4_iou	0.37355104269092737
class_5_iou	0.8632219482620936
class_6_iou	0.9019125503970717
class_7_iou	0.6845946740685235

**Parameters (11)**

Parameter	Value
class_mapping	custom_id_to_group
model_architecture	FPN
model_name	FPN_efficientnetb0_finetune_augTrue
encoder	efficientnetb0
encoder_freeze	False
script_batch_size	8
script_img_size	(224, 224)
test_mode	False
max_epochs	20
device	/GPU:0
subset_size	None

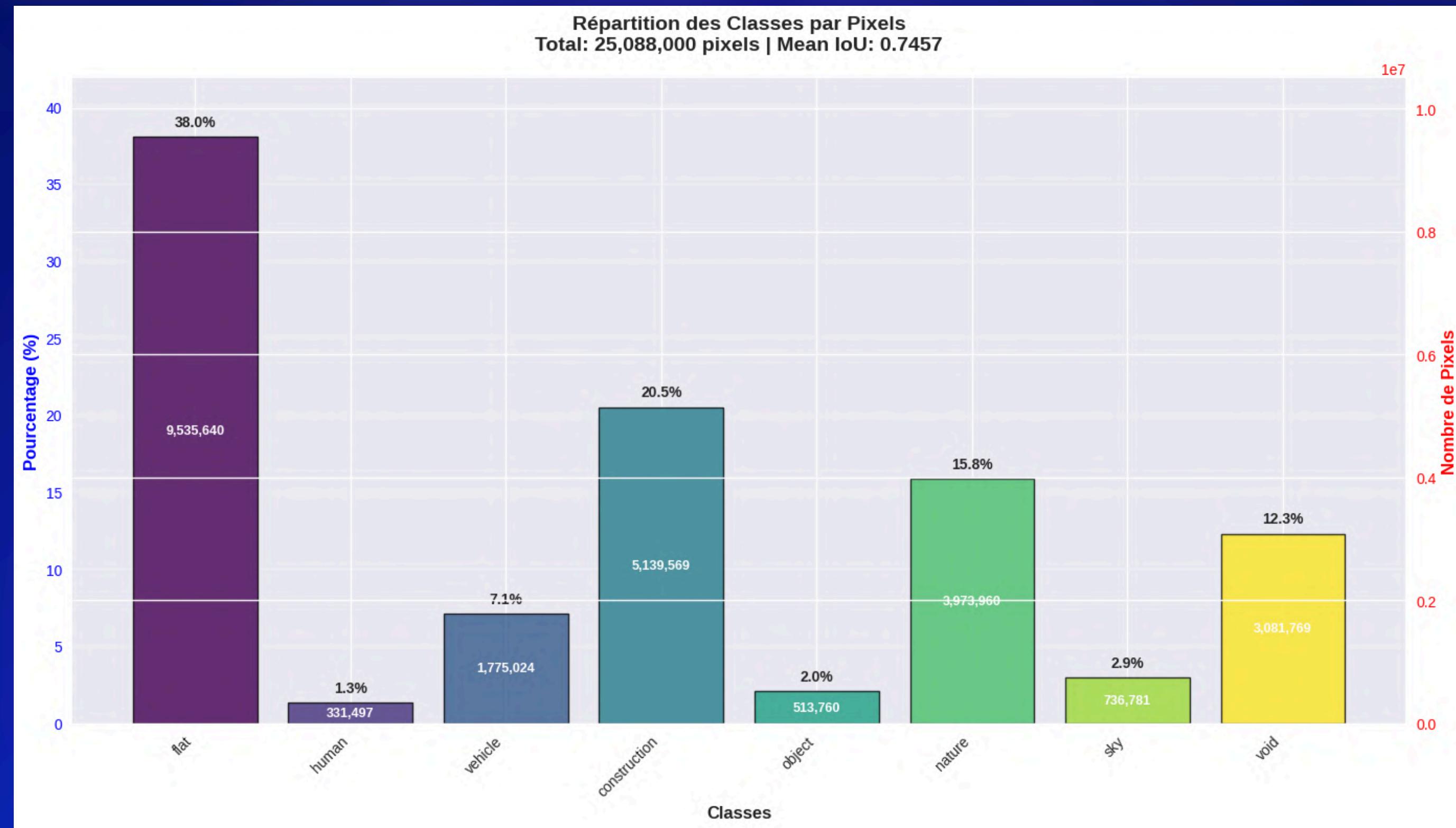
# RÉSULTATS

## SUIVI D'EXPÉRIENCES MLFLOW : MEILLEUR MODÈLE



# RÉSULTATS

## SUIVI D'EXPÉRIENCES MLFLOW : RÉPARTITION CLASSES DATASET TEST



# RÉSULTATS

## SUIVI D'EXPÉRIENCES MLFLOW : HYPERPARAMETERS TUNING

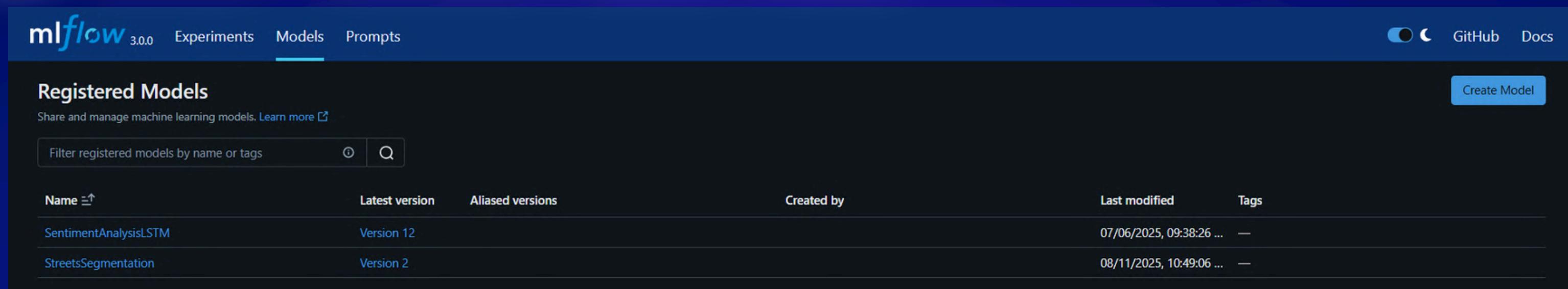




MISE EN  
PRODUCTION

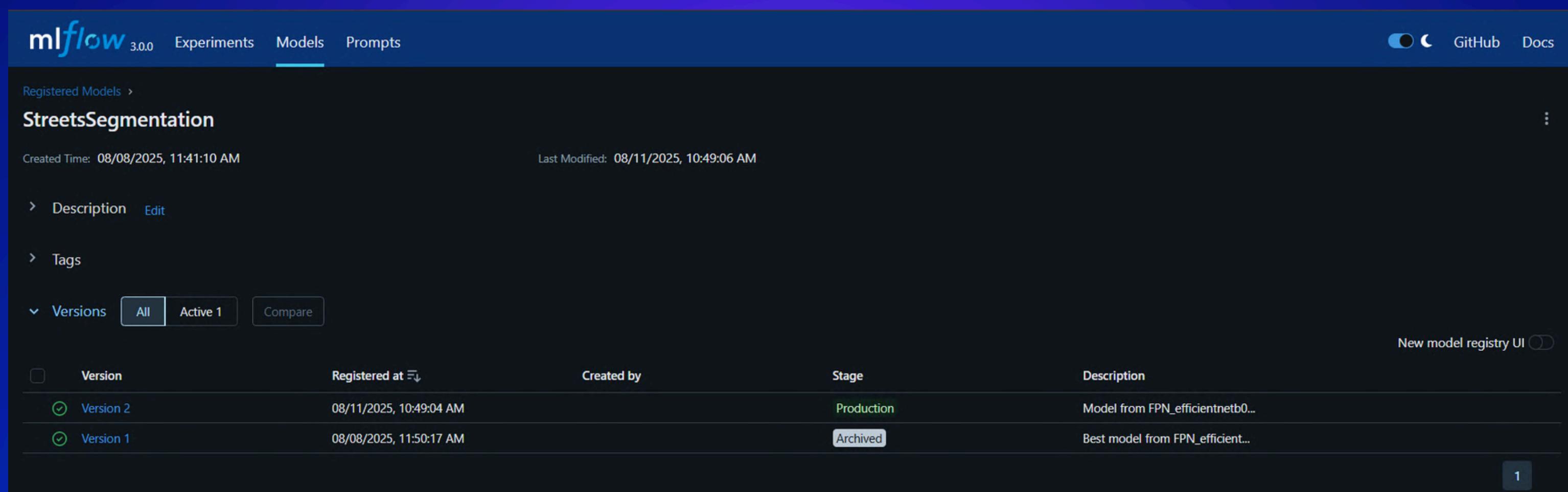
# DÉVELOPPEMENT API

## MLFLOW : MEILLEUR MODÈLE AVANCÉ



The screenshot shows the MLflow UI for the 'Registered Models' page. The top navigation bar includes the 'mlflow' logo, version '3.0.0', and tabs for 'Experiments', 'Models' (which is active), and 'Prompts'. On the right, there are GitHub and Docs links. Below the header, a section titled 'Registered Models' with a subtitle 'Share and manage machine learning models.' and a 'Learn more' link. A search bar with placeholder 'Filter registered models by name or tags' and a 'Create Model' button. The main table lists registered models:

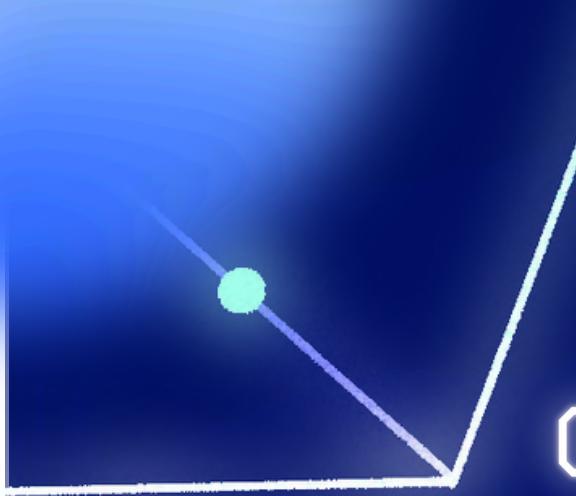
Name	Latest version	Aliased versions	Created by	Last modified	Tags
SentimentAnalysisLSTM	Version 12	—	—	07/06/2025, 09:38:26 ...	—
StreetsSegmentation	Version 2	—	—	08/11/2025, 10:49:06 ...	—



The screenshot shows the MLflow UI for the 'StreetsSegmentation' model details page. The top navigation bar is identical to the previous screen. The page title is 'StreetsSegmentation'. It displays creation and modification times: 'Created Time: 08/08/2025, 11:41:10 AM' and 'Last Modified: 08/11/2025, 10:49:06 AM'. Below this, there are sections for 'Description' (with an 'Edit' link) and 'Tags'. Under the 'Versions' section, there are buttons for 'All', 'Active 1', and 'Compare'. A 'New model registry UI' link is also present. The main table lists model versions:

Version	Registered at	Created by	Stage	Description
Version 2	08/11/2025, 10:49:04 AM	—	Production	Model from FPN_efficientnetb0...
Version 1	08/08/2025, 11:50:17 AM	—	Archived	Best model from FPN_efficient...

A small number '1' is visible in the bottom right corner.



# DÉVELOPPEMENT API

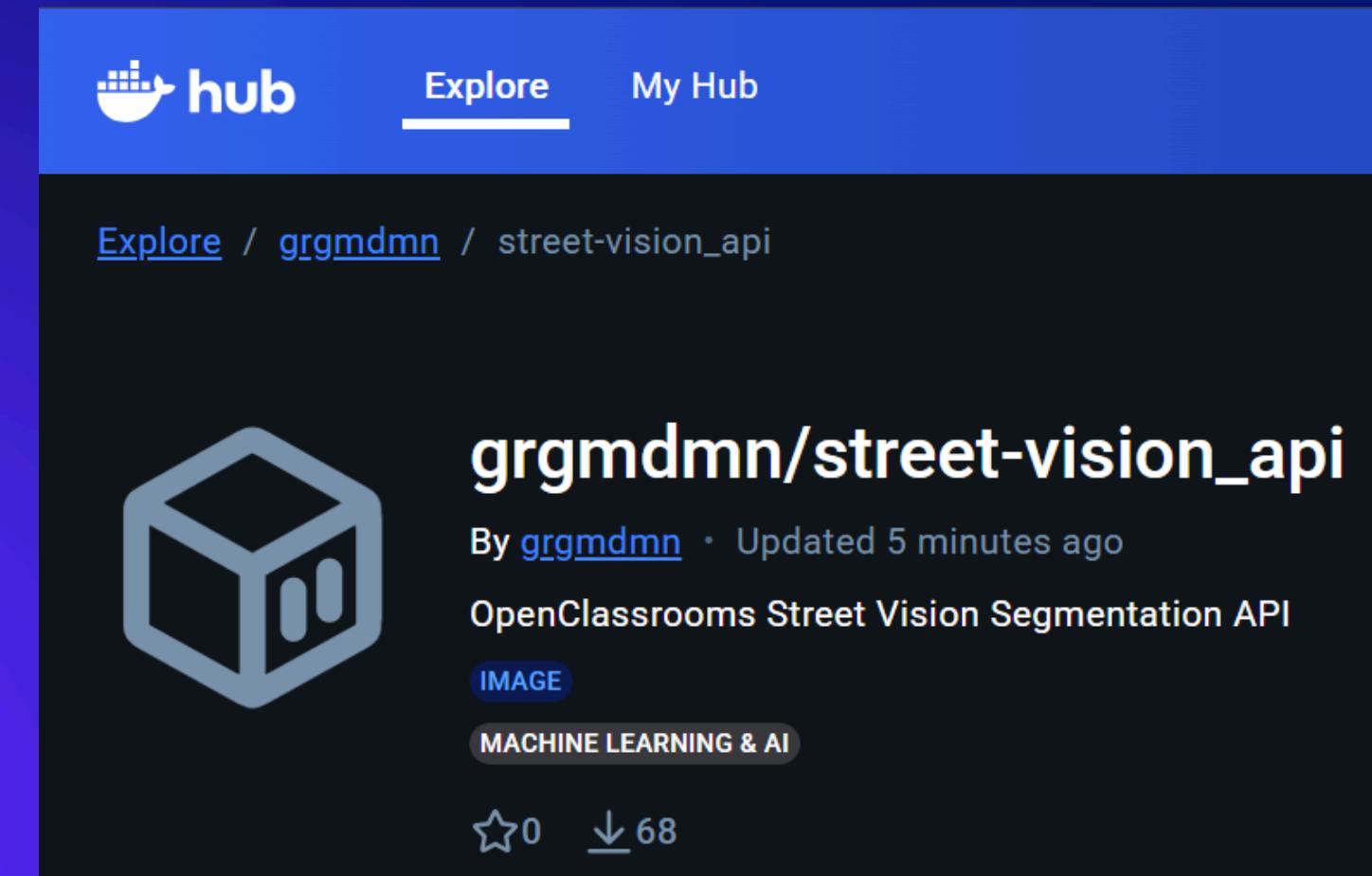
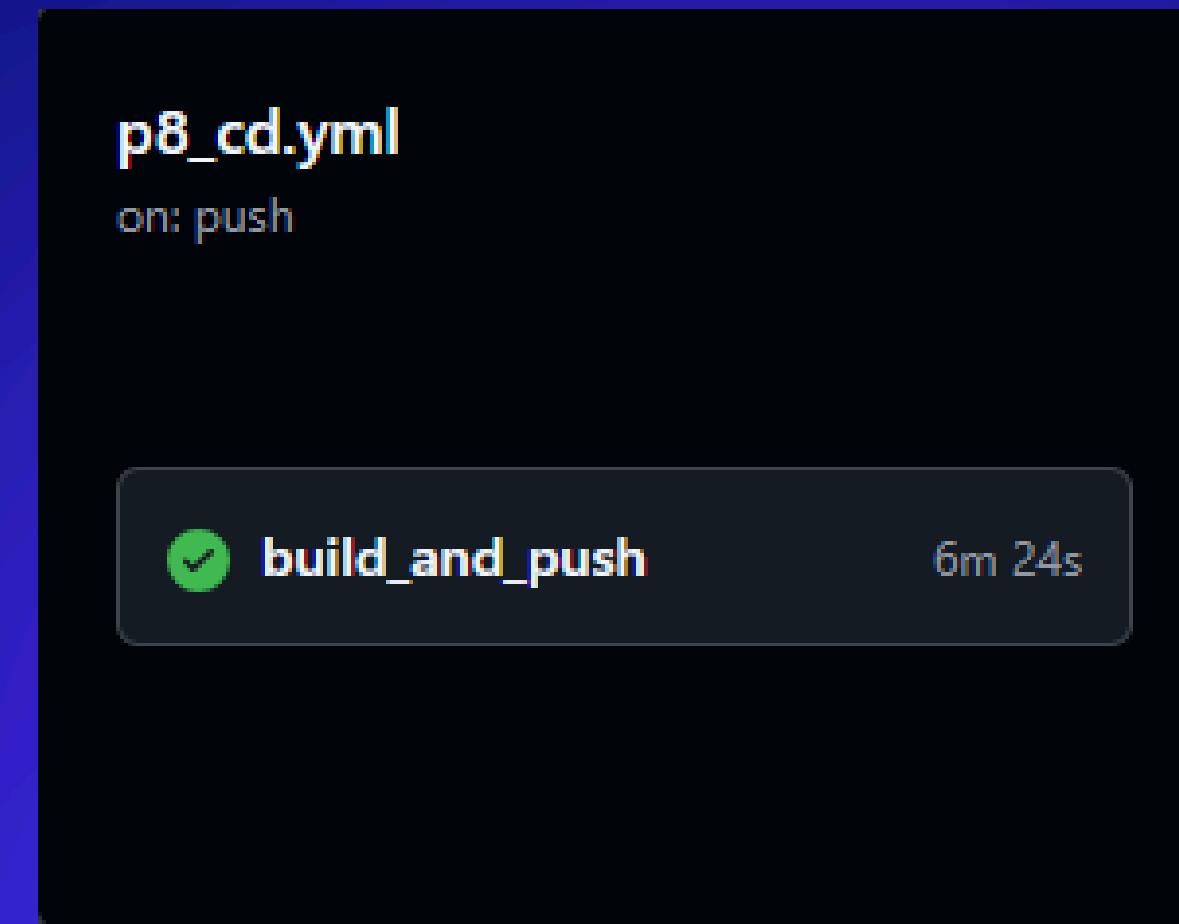
## CONSTRUCTION DE STREET VISION API

- 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

# PIPELINE

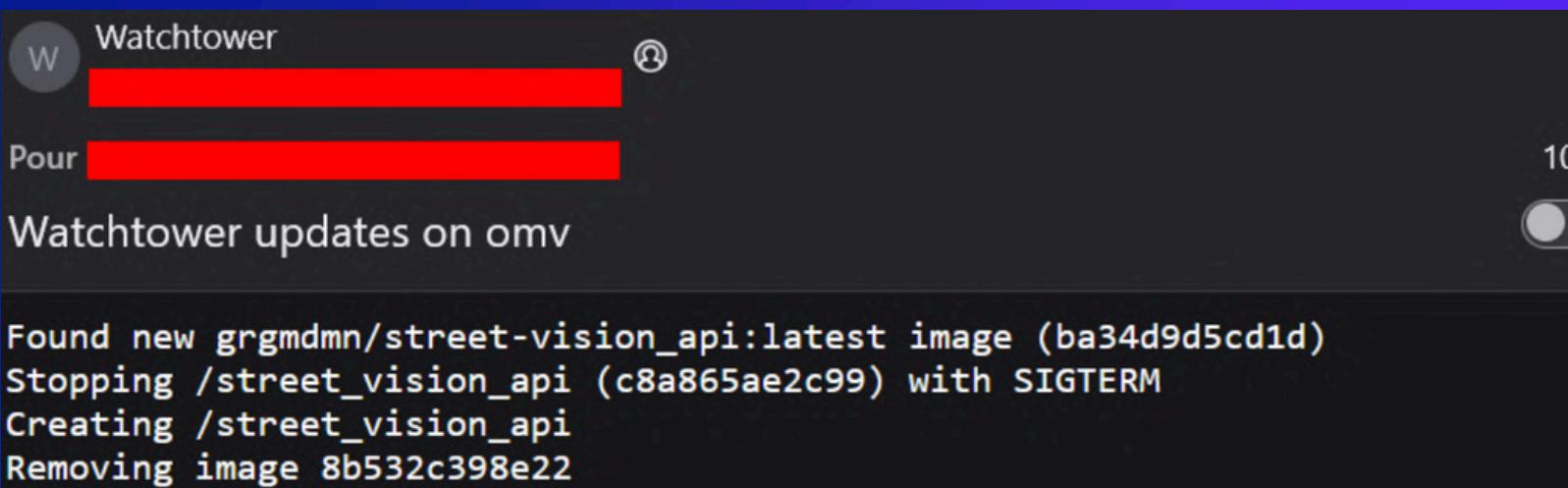
# DÉPLOIEMENT CONTINU

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



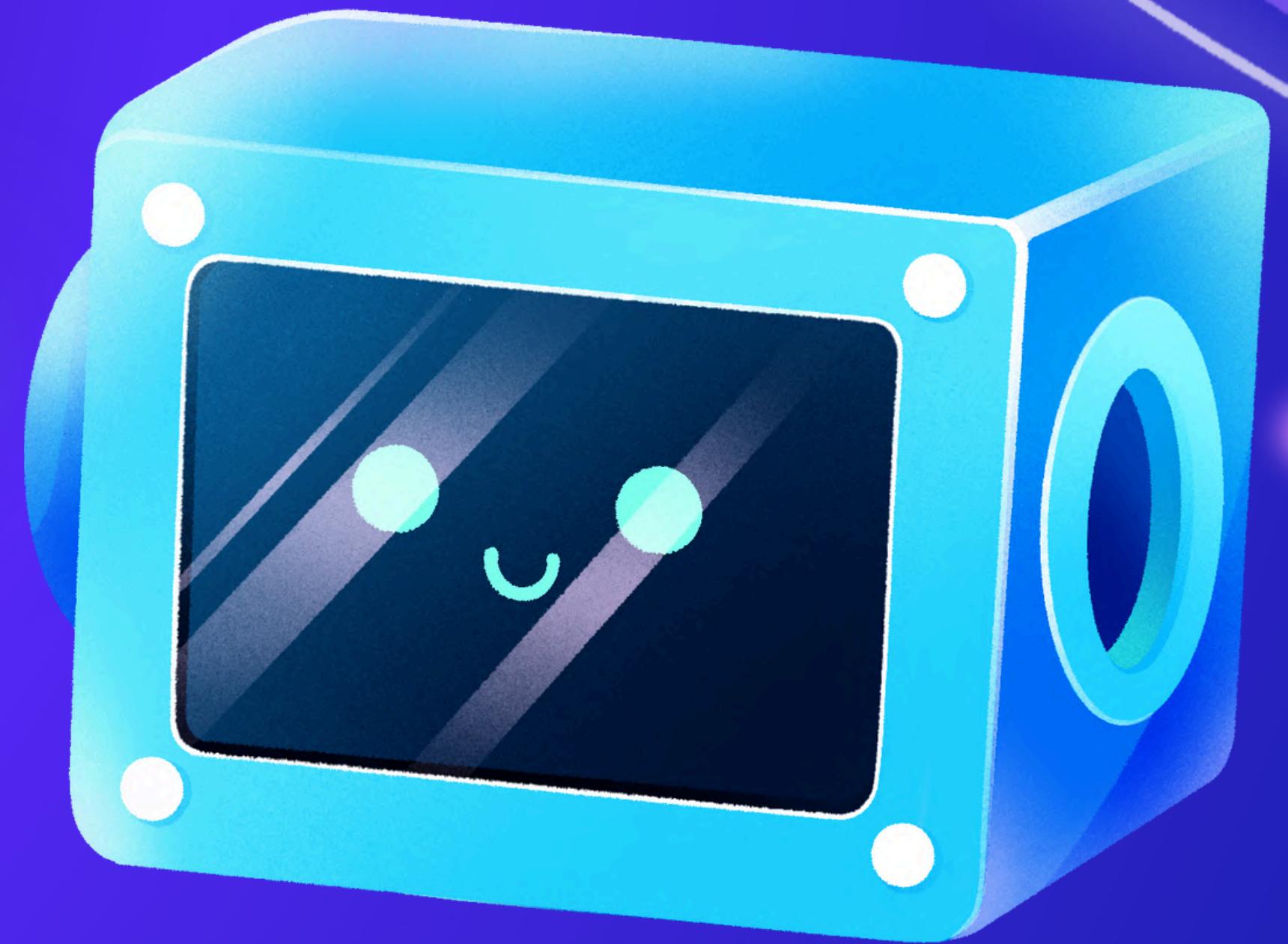
# DÉPLOIEMENT

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

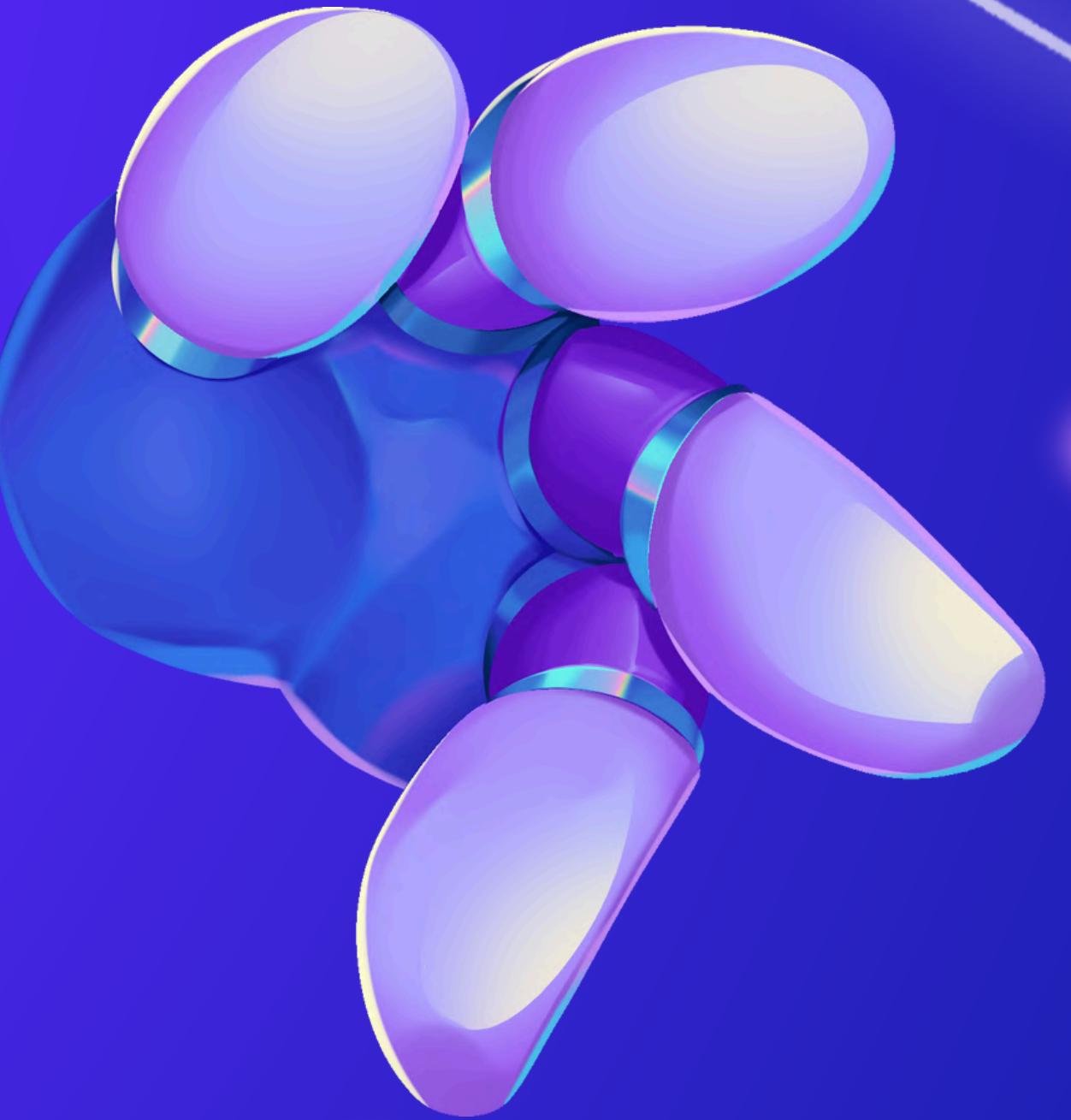


Nom	Description	Etat
MinIO	Creating S3 buckets	Up
MLFlow	MLOps for Machine Learning (OpenClassRooms Project 7)	Up
Nextcloud	Cloud Drive App	Up
Portainer	docker containers catalog	Down
PostgreSQL	Relational Database	Up
Sentiment_API	Tweets Sentiment Prediction : uses Word Embedding + LSTM deep learning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

# DÉMONSTRATION API



**RESPECT RGPD**



# RESPECT RGPD

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

Principe RGPD	Mesures mises en place
<b>Licéité, loyauté, transparence</b>	API publique, pas d'info perso collectée, usage clair segmentation
<b>Limitation des finalités</b>	Segmentation sémantique uniquement, pas d'autre usage
<b>Minimisation des données</b>	Aucune donnée utilisateur stockée, traitement images temporaire
<b>Exactitude des données</b>	Traitement temps réel, pas de stockage persistant données entrée
<b>Sécurité et confidentialité</b>	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

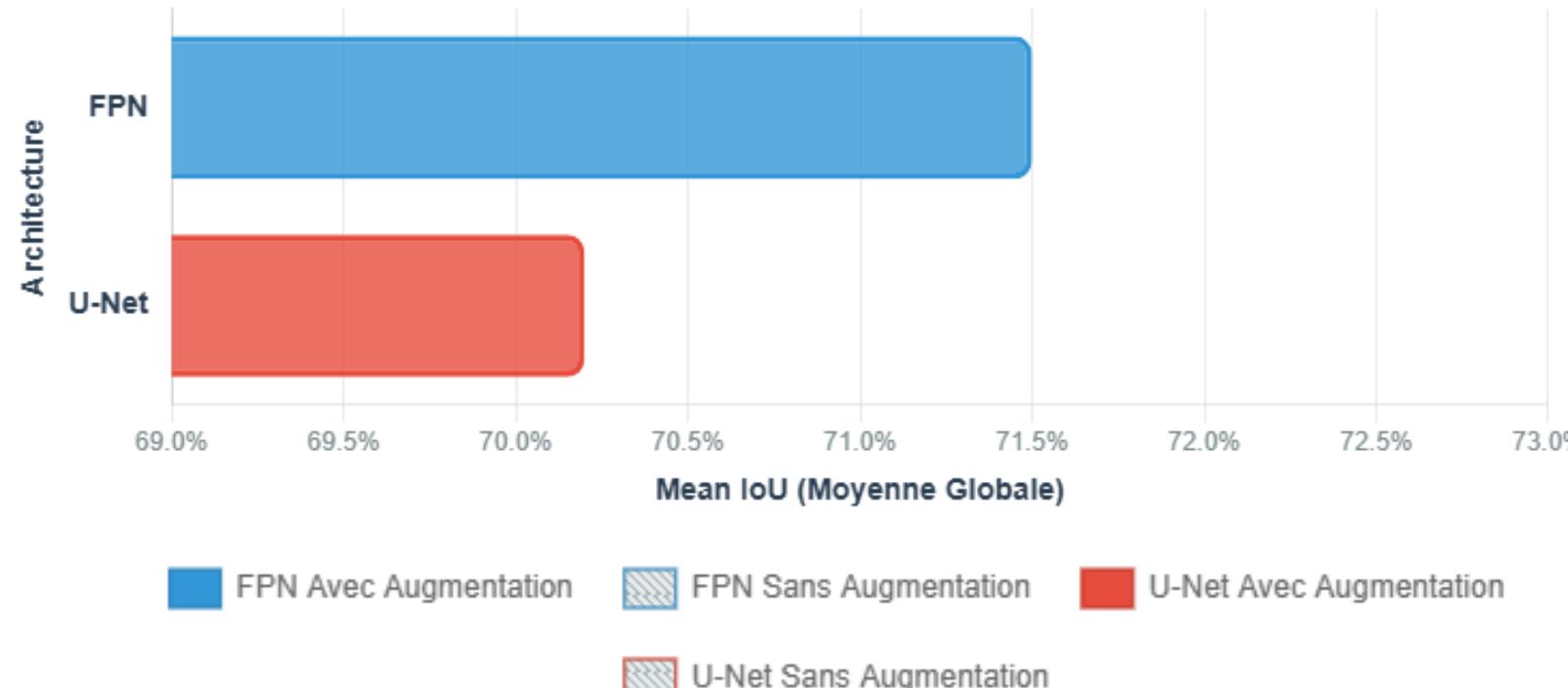
# PERSPECTIVES

- 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

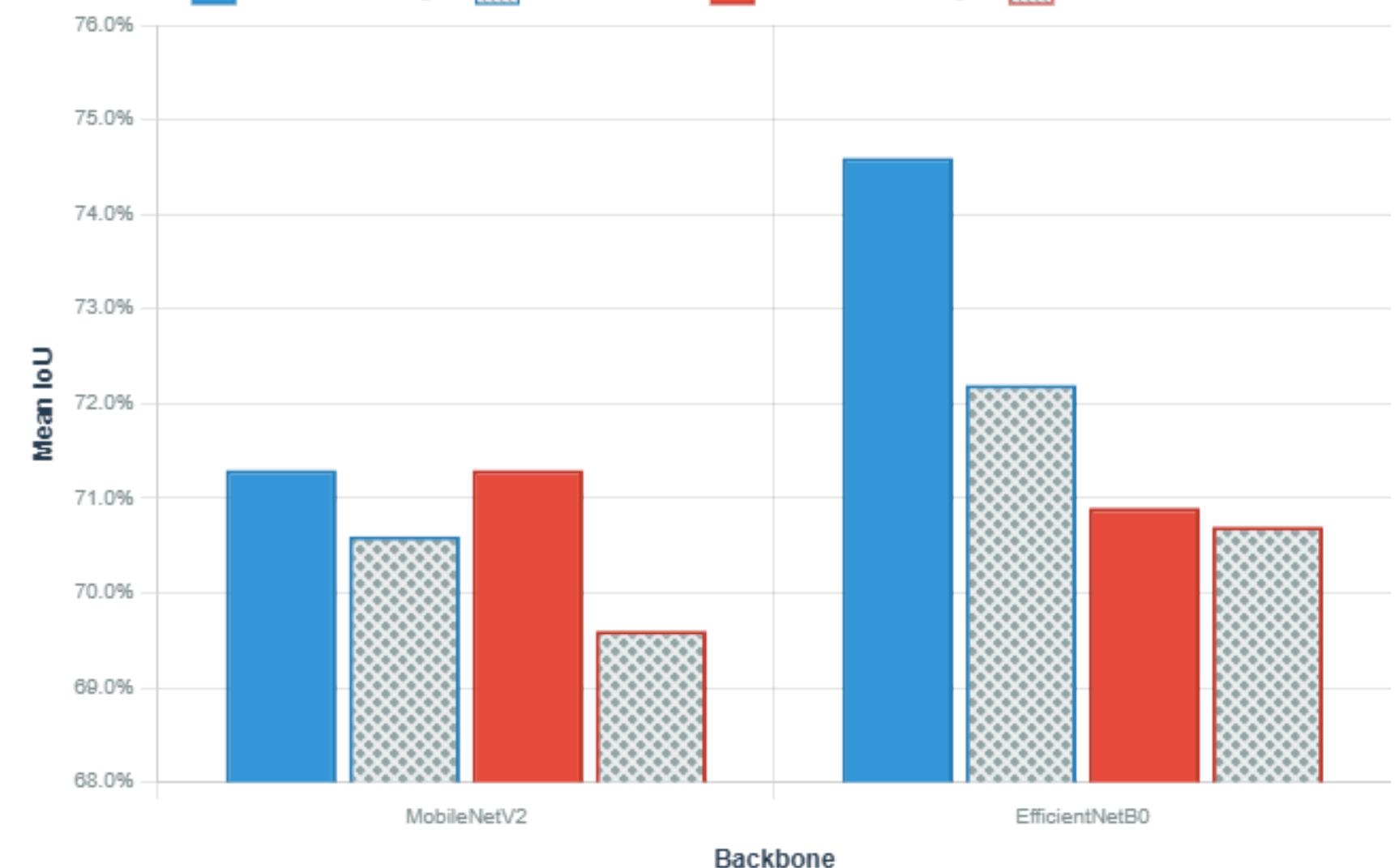
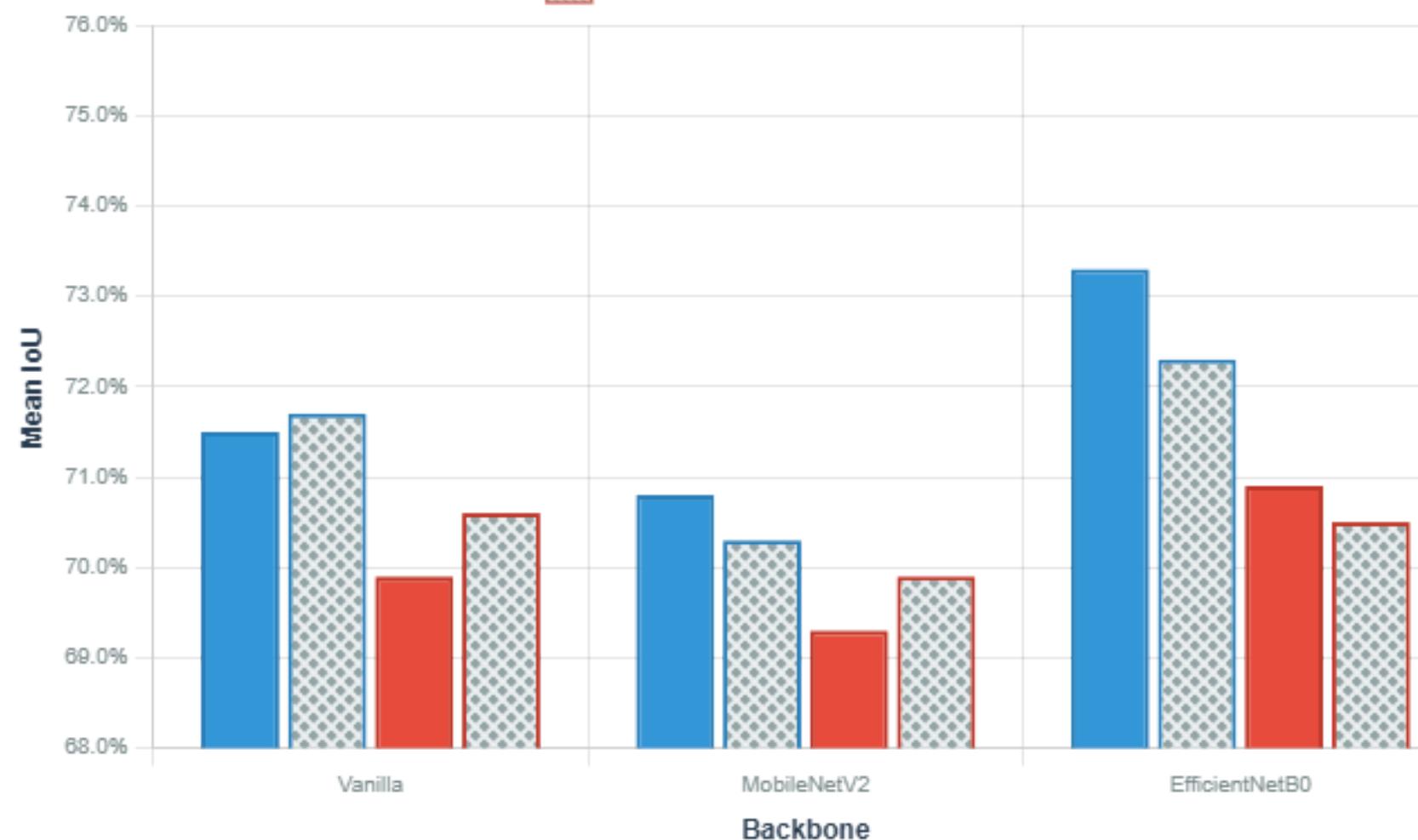
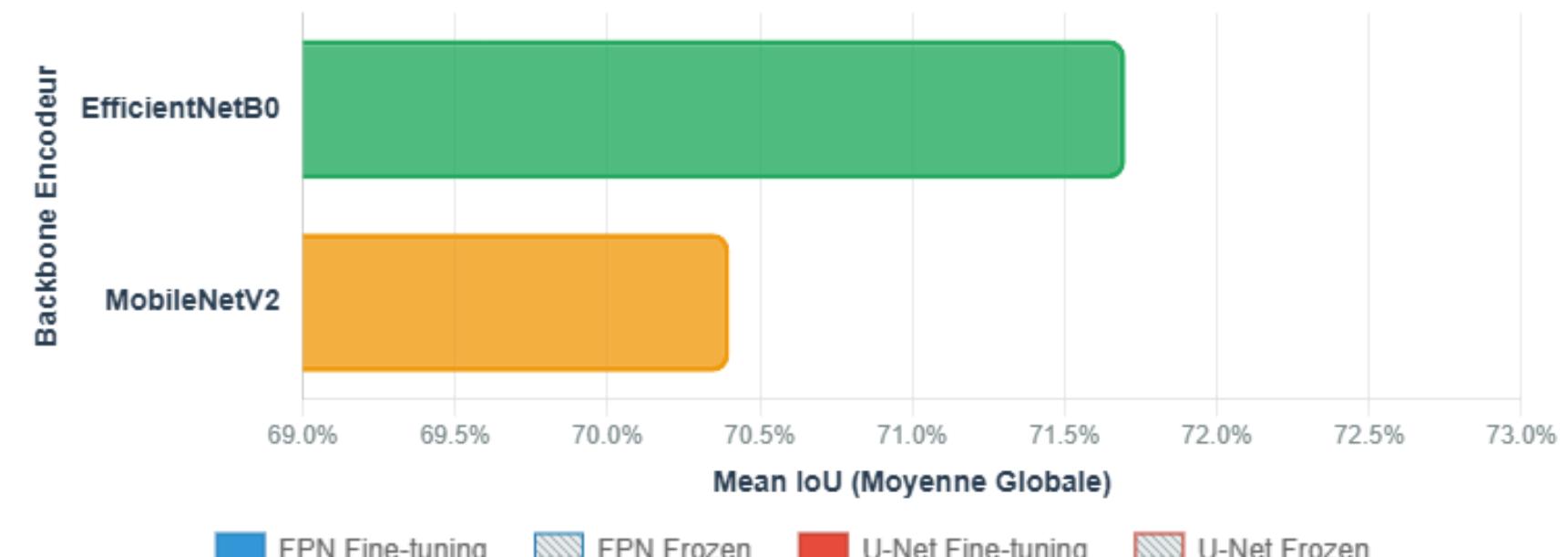
THANK YOU!

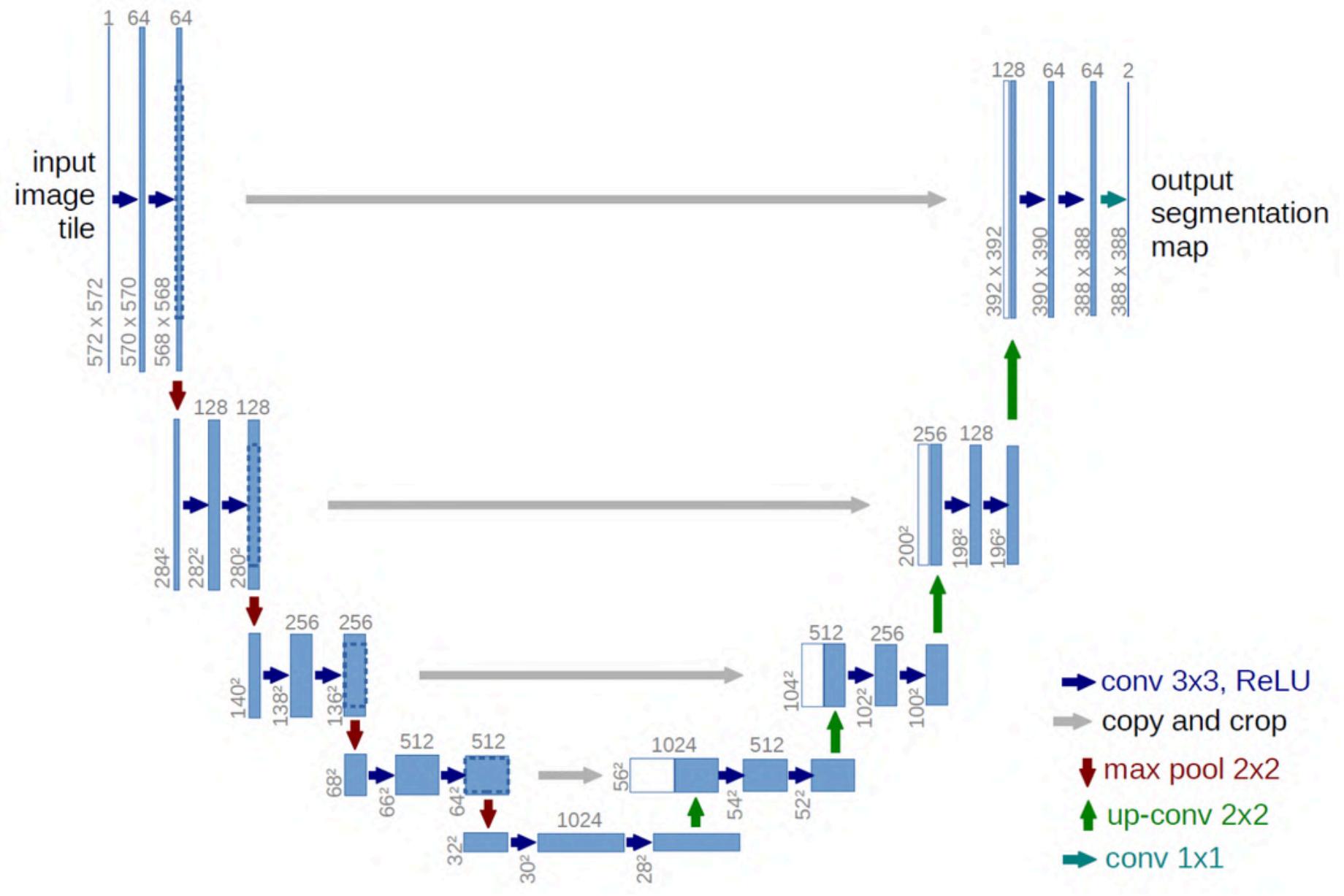


**Performance Globale : FPN vs U-Net**



**Performance Comparative : Backbones Encodeurs**





Feature Pyramid Network (FPN) [3]

