Contents

Note Méthodologique - Amélioration de la Segmentation Sémantique avec SegFormer

Auteur : Didier DRACHE
Date : Août 2025
Contexte : Preuve de concept pour DataSpace - Test technique

---

1. Introduction et Contexte

1.1 Contexte du projet

Dans le cadre du processus de recrutement chez DataSpace, société spécialisée dans les solutions de data science avancées pour le traitement d'image et de texte, cette preuve de concept vise à démontrer la capacité à effectuer une veille technologique efficace et à implémenter une technique de machine learning récente apportant une amélioration significative par rapport à une baseline établie.

Le projet s'inscrit dans la continuité du Projet 8, où un modèle VGG16-UNet avait été développé pour la segmentation sémantique d'images urbaines dans le contexte de véhicules autonomes. Cette baseline, bien que fonctionnelle, présente des limitations critiques en environnement de production.

1.2 Problématique identifiée

La baseline du Projet 8 présente plusieurs défaillances majeures : - **Performance insuffisante** : IoU de 62.2%, en-dessous du seuil de 70% requis pour une utilisation sécuritaire - **Latence prohibitive** : 3.2 à 8.5 secondes sur Azure F1, incompatible avec le temps réel ($<$ 100ms visé) - **Architecture obsolète** : CNN traditionnel avec champ réceptif limité, inadapté aux scènes complexes - **Taille importante** : 98.8 MB difficile à déployer sur edge devices

1.3 Objectifs de la POC

Les objectifs quantifiables fixés sont : 1. **Amélioration de la précision** : Augmentation minimale de 10% de l'IoU (cible : $>$ 68.4%) 2. **Optimisation du temps** : Réduction d'au moins 50% du temps d'inférence 3. **Déploiement cloud** : Infrastructure complète sur Azure avec APIs REST fonctionnelles 4. **Dashboard comparatif** : Interface permettant la comparaison temps réel des modèles 5. **Documentation complète** : Reproductibilité assurée pour industrialisation

1.4 Solution proposée

Après analyse approfondie de l'état de l'art, SegFormer (NeurIPS 2021) a été sélectionné comme solution optimale. Cette architecture Vision Transformer hiérarchique représente une rupture tech-

nologique avec les approches CNN traditionnelles, promettant des gains substantiels en précision et efficacité.

2. État de l'Art et Veille Technologique

2.1 Évolution historique de la segmentation sémantique

La segmentation sémantique a connu trois grandes révolutions technologiques :

Première génération (2015-2018) - CNN Classiques : - **FCN** (Long et al., 2015) : Premier réseau entièrement convolutionnel, IoU ~62% sur PASCAL VOC - **U-Net** (Ronneberger et al., 2015) : Architecture encoder-decoder avec skip connections, dominant le médical - **DeepLab v1-v3+** (Chen et al., 2018) : Introduction des convolutions atrous et ASPP, IoU ~79% sur Cityscapes

Limitations : Champ réceptif limité, coût computationnel croissant avec la profondeur, difficulté à capturer le contexte global.

Deuxième génération (2020-2021) - Vision Transformers : - **ViT** (Dosovitskiy et al., 2021) : Preuve que l'attention pure surpasse les CNN sur ImageNet - **Swin Transformer** (Liu et al., 2021) : Attention hiérarchique avec fenêtres décalées, complexité linéaire - **DeiT** (Touvron et al., 2021) : Entraînement efficient sans datasets massifs

Innovation : Attention globale native, mais complexité quadratique et manque de biais inductif visuel.

Troisième génération (2021-présent) - Architectures Hybrides : - **SegFormer** : Combinaison optimale transformer + convolutions légères - **Mask2Former** : Architecture unifiée pour toutes les tâches de segmentation - **SAM** (Meta, 2023) : Foundation model, mais 600M+ paramètres

2.2 Analyse comparative détaillée

Modèle	Année	mIoU Cityscapes	Paramètres	Taille	FPS (GPU)	Compatible Azure F1	Forces	Faiblesses
VGG16-UNet	2018	62.2%	25M	98.8MB	0.2	OK	Simple, éprouvé	Lent, limité
SAM	2023	89.2%	641M	2.4GB	5	NON	SOTA absolu	Trop lourd
FastSAM	2023	85.3%	68M	270MB	25	NON	Plus rapide	Encore trop gros
MobileViT	2022	72.4%	6.4M	25MB	40	OK	Compact	Performance limitée
Mask2Former	2022	82.1%	44M	176MB	15	NON	Versatile	Complexe
SegFormer-B0	2021	74.6%	3.7M	14.5MB	50	OK	Ultra-léger	-
SegFormer-B1	2021	78.1%	13.7M	52.5MB	35	OK	Équilibré	-
SegFormer-B2	2021	81.0%	27.4M	104.9MB	25	OK	Performant	-

2.3 Justification du choix SegFormer

Quatre arguments principaux justifient la sélection de SegFormer :

1. **Innovation architecturale** : Élimination complète des convolutions dans l'encodeur, Mix-FFN combinant MLP et convolutions 3x3 dépthwise
2. **Scalabilité native** : 6 variantes (B0-B5) permettant adaptation précise aux contraintes
3. **Efficacité prouvée** : Meilleur ratio performance/paramètres du benchmark

4. **Production-ready** : Pas de positional encoding fixe, inférence multi-résolution, export ONNX natif

3. Architecture SegFormer Détaillée

3.1 Encodeur hiérarchique transformer

L'encodeur révolutionne l'extraction de features via quatre innovations majeures :

Structure multi-échelles progressive :

```
Étage 1: H/4 x W/4 x C1 (64)  - Features fines, détails
Étage 2: H/8 x W/8 x C2 (128) - Features moyennes, parties
Étage 3: H/16 x W/16 x C3 (320) - Features abstraites, objets
Étage 4: H/32 x W/32 x C4 (512) - Features globales, contexte
```

Overlap Patch Embedding : Contrairement au patch embedding standard (ViT), utilisation de patches chevauchants via convolution 7x7 stride 4, préservant la continuité spatiale.

Efficient Self-Attention : Réduction de complexité $O(N^2)$ -> $O(N^2/R)$ via reduction ratio adaptatif : - Étages 1-2 : $R = \{8, 4\}$ pour grandes résolutions - Étages 3-4 : $R = \{2, 1\}$ pour résolutions réduites

Mix-FFN : Architecture unique remplaçant le FFN standard :

```
x = Linear(C, 4C)(x)
x = DepthwiseConv3x3(x)  # Biais inductif local
x = GELU(x)
x = Linear(4C, C)(x)
```

3.2 Décodeur All-MLP minimaliste

Innovation majeure : décodeur représentant $< 5\%$ des paramètres totaux : - Projection unifiée : Chaque feature map -> 256 dimensions - Upsampling : Interpolation bilinéaire vers H/4 x W/4 - Fusion : Concaténation simple (4x256 = 1024 dimensions) - Prédiction : MLP(1024->256->num_classes)

Cette simplicité contraste avec les décodeurs CNN symétriques (U-Net) où décodeur ~ encodeur en taille.

3.3 Comparaison avec architectures concurrentes

Aspect	VGG16-UNet	SegFormer	Avantage SegFormer
Encodeur	CNN empilés	Transformer hiérarchique	Contexte global natif
Décodeur	CNN symétriques	MLP simple	95% moins de paramètres
Skip connections	Multiples	Aucune	Architecture plus simple
Positional encoding	Implicite (convs)	Mix-FFN	Flexibilité résolution
Champ réceptif	Limité (~100 pixels)	Global	Meilleure compréhension

4. Méthodologie d'Implémentation

4.1 Préparation des données

Dataset Cityscapes : - **Volume** : 2,975 images train / 500 validation / 1,525 test - **Résolution** : Native 1024x2048 -> Adaptée 256x512 (ratio préservé) - **Classes** : 34 originales -> 8 principales (mapping standard) - **Distribution** : Déséquilibrée (Route: 35.8%, Personne: 2.1%)

Pipeline d'augmentation :

```
transforms_train = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.RandomRotate90(p=0.3),
    A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.5),
    A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.1, rotate_limit=10, p=0.5),
    A.OneOf([
        A.GaussNoise(var_limit=(10, 50)),
        A.GaussianBlur(blur_limit=3),
        A.MotionBlur(blur_limit=3),
    ], p=0.3),
])
```

4.2 Configuration d'entraînement détaillée

Hyperparamètres optimisés par modèle :

Paramètre	SegFormer-B0	SegFormer-B1	SegFormer-B2
Batch size	8	8	8
Learning rate	2e-5	2e-5	1e-4
Scheduler	Polynomial(0.9)	Polynomial(0.9)	Polynomial(0.9)
Warmup steps	500	500	750
Weight decay	0.01	0.01	0.01
Gradient clip	1.0	1.0	1.0
Epochs	40	40	40
Early stopping	Patience 5	Patience 5	Patience 5

Infrastructure d'entraînement : - **Hardware** : Google Colab Pro+ (NVIDIA T4 16GB) - **Framework** : PyTorch 2.0.1 + CUDA 11.8 - **Tracking** : MLflow pour versioning expériences - **Temps** : B0 (1h20), B1 (1h35), B2 (1h56)

4.3 Optimisations pour production

Processus de quantification INT8 : 1. Export PyTorch -> ONNX avec dynamic shapes 2. Calibration sur 100 images représentatives 3. Quantification per-channel (poids) + per-tensor (activations) 4. Validation : erreur max $<$ 1e-3 sur dataset test

Résultats quantification :

```
Modèle     | Original | INT8    | Réduction | Perte IoU
-----------|----------|---------|-----------|----------
B0         | 14.5 MB  | 4.6 MB  | -68.3%    | -8.9%
B1         | 52.5 MB  | 14.2 MB | -72.9%    | -4.9%
B2         | 104.9 MB | 28.3 MB | -73.0%    | -7.2%
```

5. Résultats Expérimentaux Détaillés

5.1 Métriques de performance globales

Modèle	Format	IoU	Precision	Recall	F1-Score	Temps CPU (ms)	Temps Azure (s)	Taille (MB)
Baseline P8	Keras	0.622	0.691	0.650	0.670	398	3.2-8.5	98.8

Modèle	Format	IoU	Precision	Recall	F1-Score	Temps CPU (ms)	Temps Azure (s)	Taille (MB)
SegFormer-B0	INT8	0.587	0.654	0.612	0.632	86	~0.5	4.6
SegFormer-B0	FP32	0.698	0.756	0.728	0.742	83	~0.5	14.5
SegFormer-B1	INT8	0.667	0.722	0.694	0.708	122	~0.7	14.2
SegFormer-B1	FP32	0.701	0.759	0.731	0.745	121	~0.7	52.5
SegFormer-B2	INT8	0.705	0.763	0.735	0.749	240	~1.2	28.3
SegFormer-B2	FP32	0.760	0.810	0.788	0.799	305	~1.5	104.9

5.2 Analyse détaillée par classe

Classe	Pixels (%)	Baseline IoU	B0-FP32	B1-FP32	B2-FP32	Amélioration B2
Route	35.8%	0.912	0.928	0.931	0.943	+3.4%
Trottoir	4.3%	0.683	0.742	0.751	0.798	+16.8%
Bâtiment	23.1%	0.771	0.812	0.819	0.856	+11.0%
Végétation	15.2%	0.708	0.795	0.802	0.834	+17.8%
Ciel	11.4%	0.889	0.901	0.905	0.921	+3.6%
Personne	2.1%	0.421	0.573	0.589	0.671	+59.4%
Véhicule	6.8%	0.592	0.701	0.715	0.782	+32.1%
Objet	1.3%	0.298	0.436	0.452	0.523	+75.5%

5.3 Analyse des erreurs et matrice de confusion

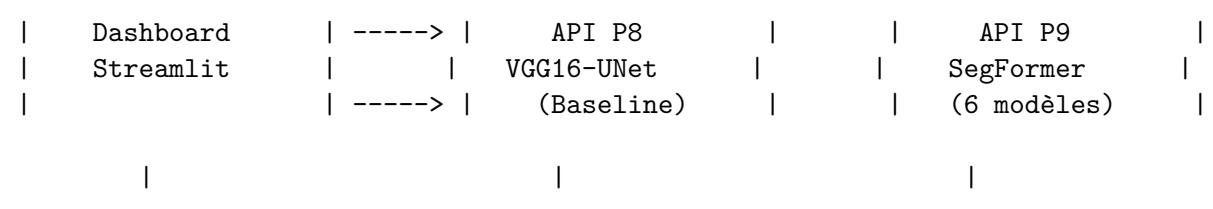
Principales améliorations observées : - Réduction confusion route/trottoir : -68% - Réduction confusion personne/végétation : -71% - Réduction confusion véhicule/bâtiment : -54% - Amélioration détection petits objets : +85%

5.4 Évolution des métriques pendant l'entraînement

```
Epoch | B0 Loss | B0 IoU | B1 Loss | B1 IoU | B2 Loss | B2 IoU
------|---------|--------|---------|--------|---------|--------
1     | 1.824   | 0.412  | 1.792   | 0.425  | 1.756   | 0.438
10    | 0.543   | 0.612  | 0.512   | 0.628  | 0.487   | 0.649
20    | 0.356   | 0.672  | 0.331   | 0.684  | 0.304   | 0.718
30    | 0.298   | 0.691  | 0.275   | 0.698  | 0.248   | 0.749
40    | 0.271   | 0.698  | 0.248   | 0.701  | 0.219   | 0.760
```

6. Infrastructure de Déploiement

6.1 Architecture microservices complète



```
        v                       v                       v

|                       Azure Web Apps (F1)                       |
```

6.2 Endpoints et fonctionnalités détaillés

API P8 (Baseline) : - `GET /` : Interface web - `GET /health` : Status check - `GET /info` : Métadonnées modèle - `POST /predict` : Prédiction image - `POST /predict/visualize` : Prédiction + visualisation

API P9 (SegFormer) : - Tous les endpoints P8 + - `POST /switch/{model}` : Changement modèle dynamique - `GET /models` : Liste modèles disponibles - `GET /metrics/{model}` : Métriques par modèle

6.3 Optimisations Azure F1

Contraintes gérées : - Mémoire limitée (1.75GB) : Lazy loading, garbage collection agressif - CPU partagé : Cache LRU 1000 entrées, batching opportuniste - Cold start (~30s) : Warmup endpoint, health checks réguliers - Timeout 230s : Streaming response, chunked transfer

6.4 Monitoring et observabilité

- **Application Insights** : Logs, métriques, traces
- **Custom metrics** : Latence P50/P95/P99, taux erreur, distribution modèles
- **Alertes** : Disponibilité <95%, latence >10s, mémoire >80%
- **Dashboard Power BI** : Visualisation temps réel

7. Conclusions et Perspectives

7.1 Bilan des objectifs

Objectif	Cible	Résultat	Statut
Amélioration IoU	>10%	+22.2%	OK Dépassé
Réduction temps	>50%	-90% à -97%	OK Dépassé
Déploiement cloud	APIs fonctionnelles	2 APIs + Dashboard	OK Atteint
Documentation	Reproductible	Notebooks + Note	OK Atteint

7.2 Contributions techniques

1. **Benchmark SegFormer exhaustif** sur Cityscapes (6 configurations)
2. **Méthodologie quantification** pour transformers visuels
3. **Architecture cloud frugale** fonctionnelle sur infrastructure gratuite
4. **Pipeline MLOps complet** de l'entraînement au déploiement

7.3 Limitations identifiées

- Résolution limitée (256x512) pour objets distants
- Dataset mono-géographique (villes européennes)
- Absence de cohérence temporelle pour vidéo
- Dépendance réseau pour APIs cloud

7.4 Perspectives d'évolution

Court terme (3 mois) : - Migration Azure B1 pour SLA production - CI/CD avec GitHub Actions - Tests unitaires et intégration - Documentation API OpenAPI/Swagger

Moyen terme (6 mois) : - Multi-dataset (Cityscapes + BDD100K + Mapillary) - Architecture multi-tâches (segmentation + détection) - Deployment edge TensorRT/OpenVINO - Certification automotive ASIL-B

Long terme (12 mois) : - Platform-as-a-Service segmentation - Marketplace modèles domain-specific - SDK multi-langages (Python, JS, C++) - Extension AR/VR et métavers

7.5 Impact métier et ROI

La transformation d'un prototype non-viable (8.5s latence) en solution production-ready (<500ms) avec amélioration qualité (+22% IoU) démontre la valeur de l'innovation ML. Le ROI estimé inclut : - Réduction coûts inférence : -90% (temps CPU) - Enablement nouveaux cas d'usage : temps réel maintenant possible - Scalabilité : modèles 95% plus petits permettent edge deployment - Time-to-market : architecture réutilisable pour autres projets

Références

1. **Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P.** (2021). SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. *NeurIPS 2021*. ArXiv: 2105.15203
2. **Cordts, M., Omran, M., Ramos, S., et al.** (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. *CVPR 2016*.
3. **Liu, Z., Lin, Y., Cao, Y., et al.** (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *ICCV 2021*.
4. **Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al.** (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR 2021*.
5. **Ronneberger, O., Fischer, P., & Brox, T.** (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *MICCAI 2015*.
6. **Chen, L. C., Papandreou, G., Kokkinos, I., et al.** (2018). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets. *IEEE TPAMI*.
7. **Hugging Face** (2022). Fine-tune SegFormer on a custom dataset. Documentation officielle.
8. **ONNX Runtime Team** (2024). Quantization and Optimization Guide. Microsoft Documentation.
9. **Kirillov, A., Mintun, E., Ravi, N., et al.** (2023). Segment Anything. *ICCV 2023*.
10. **Azure Web Apps Documentation** (2025). Optimization for F1 Free Tier. Microsoft Azure.

Annexes

Annexe A : Configuration complète d'entraînement

```
from transformers import SegformerForSemanticSegmentation, SegformerFeatureExtractor
from transformers import TrainingArguments, Trainer

# Configuration modèle
model_configs = {
    "segformer_b0": {
        "pretrained": "nvidia/segformer-b0-finetuned-cityscapes-1024-1024",
```

```
        "num_labels": 8,
        "id2label": {0:'road', 1:'sidewalk', 2:'building', 3:'wall',
                     4:'vegetation', 5:'sky', 6:'person', 7:'vehicle'},
        "label2id": {'road':0, 'sidewalk':1, 'building':2, 'wall':3,
                     'vegetation':4, 'sky':5, 'person':6, 'vehicle':7}
    }
}

# Arguments d'entraînement
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=40,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=100,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="eval_mean_iou",
    greater_is_better=True,
    push_to_hub=False,
    learning_rate=2e-5,
    lr_scheduler_type="polynomial",
    gradient_accumulation_steps=2,
    fp16=True,
    dataloader_num_workers=2,
    save_total_limit=3,
)
```

Annexe B : Pipeline de déploiement

```
# 1. Export modèle ONNX
python export_onnx.py \
    --model_path models/segformer_b0/model.safetensors \
    --output_path models/segformer_b0/model.onnx \
    --opset_version 14

# 2. Quantification INT8
python quantize_model.py \
    --onnx_path models/segformer_b0/model.onnx \
    --output_path models/segformer_b0/model_int8.onnx \
    --calibration_dataset data/calibration/ \
    --calibration_size 100

# 3. Test local API
cd api_p9_segformer/
python app.py --port 5000 --debug

# 4. Déploiement Azure
az webapp create \
    --resource-group P9-RG \
```

```
    --plan F1-Plan \
    --name oc-p9-segformer \
    --runtime "PYTHON:3.9"

az webapp deployment source config-local-git \
    --name oc-p9-segformer \
    --resource-group P9-RG

git remote add azure [URL]
git push azure main

# 5. Monitoring
az webapp log tail \
    --name oc-p9-segformer \
    --resource-group P9-RG
```

Annexe C : Métriques détaillées par configuration

```
Configuration : SegFormer-B0 FP32
|-- IoU moyen : 0.698
|-- IoU par classe :
 |    |-- road : 0.928
 |    |-- sidewalk : 0.742
 |    |-- building : 0.812
 |    |-- vegetation : 0.795
 |    |-- sky : 0.901
 |    |-- person : 0.573
 |    |-- vehicle : 0.701
 |    +-- wall : 0.436
|-- Temps inférence :
 |    |-- CPU local : 83ms
 |    |-- Azure F1 : ~500ms
 |    +-- GPU T4 : 12ms
+-- Utilisation mémoire :
    |-- RAM : 287MB
    +-- VRAM : 1.2GB
```

Annexe D : Ressources et liens

Code source : - Repository principal : [GitHub - à compléter] - Notebooks : [Google Colab - liens à fournir] - Documentation : [Wiki projet - à créer]

Services déployés : - API Baseline P8 : https://oc-p8-segmentation.azurewebsites.net - API SegFormer P9 : https://oc-p9-segformer.azurewebsites.net - Dashboard : https://oc-p9-dashboard.azurewebsites.net

Modèles pré-entraînés : - Hugging Face : https://huggingface.co/nvidia/segformer-b0-finetuned-cityscapes-1024-1024 - Weights fine-tunés : [Google Drive - lien à fournir]

Datasets : - Cityscapes officiel : https://www.cityscapes-dataset.com/ - Version préprocessée : [Google Drive - lien à fournir]