

Développez une preuve de concept



Sommaire

PARTIE 1 – CHOIX DU DATASET

PARTIE 2 – CHOIX DU MODÈLE

PARTIE 3 – DESCRIPTION DU MODÈLE

PARTIE 4 – RÉSULTAT DE LA MODÉLISATION

PARTIE 5 – API ET DASHBOARD

CONCLUSION

L'objet de l'étude

- Sélectionner un ancien projet et chercher à en **améliorer les performances**
- Recherche **thématique** :
 - Sélectionner une technique nouvelle
 - Monter en compétence
- Mettre en œuvre le nouvel algorithme afin de **le comparer** à un modèle plus classique
- Donner à voir les résultats grâce à un *dashboard* simple





PARTIE 1 – CHOIX DU DATASET

Quel projet ? Quel *dataset* ?

- Utilisation d'un ancien projet
- Tâche : Segmentation sémantique
- *Dataset* : CityScapes
- Modèle sélectionné à l'époque : *Unet* associé à *Resnet*

Figure 9 : Exemple connexion résiduelle

Figure 10 : Resnet34

5. Les outils utilisés

Pour développer le projet, nous utilisons :

- Keras : Keras est une bibliothèque de haut niveau pour la construction de modèles de deep learning.
- TensorFlow : TensorFlow est une bibliothèque de deep learning open source.
- OpenCV : OpenCV est une bibliothèque de vision par ordinateur.
- Flask : Flask est un framework web léger.
- Docker : Docker est une plateforme de conteneurisation.
- Azure : Azure est une plateforme de cloud computing.
- GitHub : GitHub est une plateforme de gestion de code.

Optimiser la performance du modèle en prenant en compte l'information spatiale

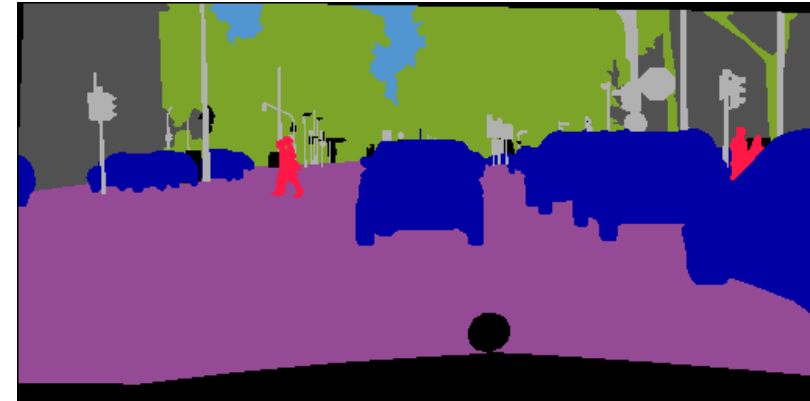
• Backbone : VGG16 et Resnet34

model	backbone	loss_function	val_iou	val_f_score	training_time	augmentation
baseline						
unet	resnet34	dice+focal	0.388	0.490	259.22 min	No
unet	vgg16	dice+focal	0.657	0.702	79.74 min	No
unet	resnet34	dice+focal	0.302	0.347	179.79 min	No
unet	resnet34	dice_CW+focal	0.664	0.732	147.38 min	Yes



Quel projet ? Quel *dataset* ?

- Images : P8_Cityscapes_leftImg8bit_trainvaltest
- Masks : P8_Cityscapes_gtFine_trainvaltest



- Prédécoupage :
 - Entraînement : 2975 images / masks
 - Validation : 500 images / masks
 - Test : 1525 images / masks

Segmentation : Fonctionnement classique

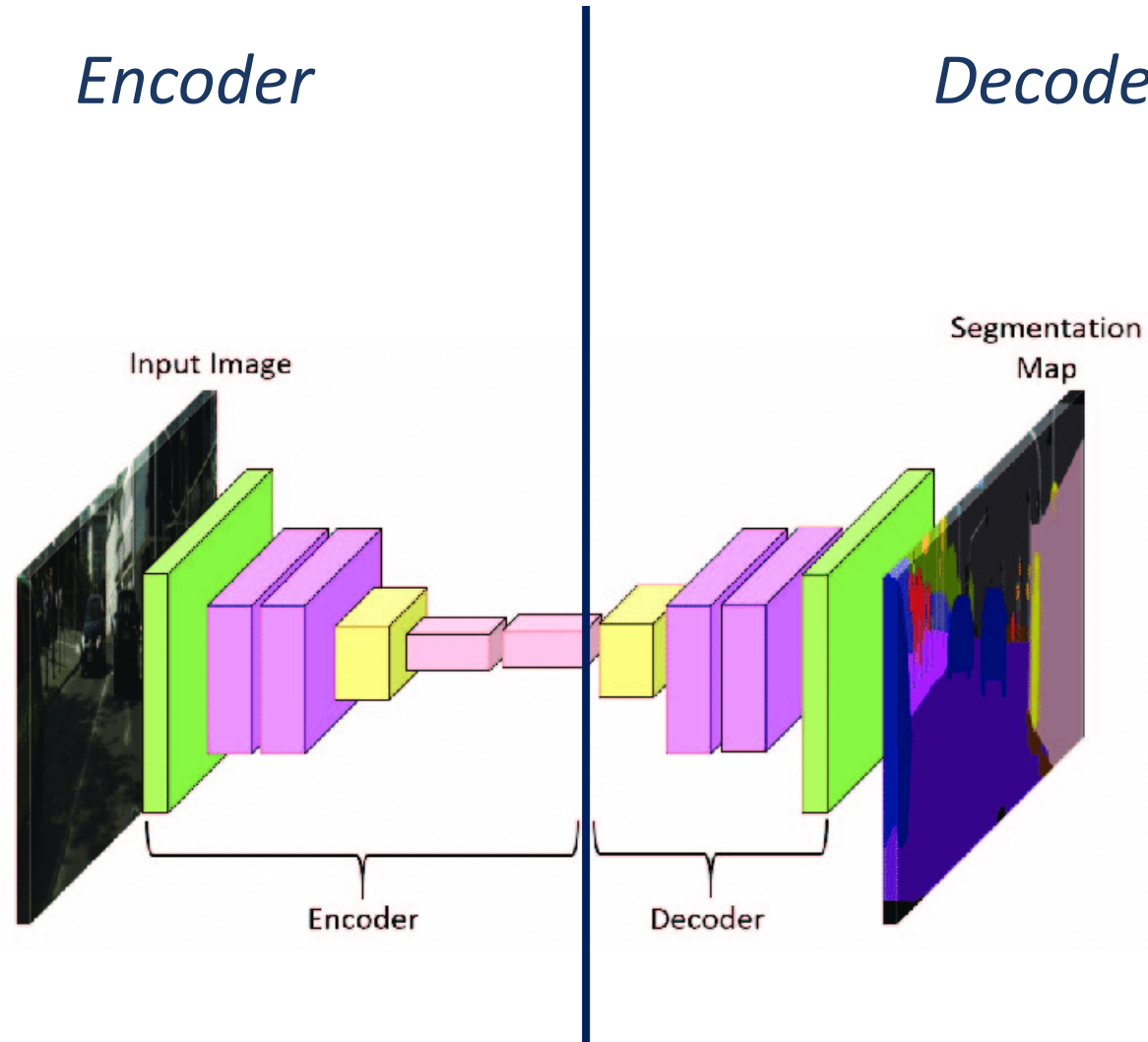
Architecture : *Encoder* *Decoder*

Extraire les features importantes

Basé sur des modèles connus sur la classification d'image (sans les couches de décision)

→ *Backbone*

Downsampling



Upsampling

Revenir aux dimensions initiales

Autant de canaux que de classes à détecter



PARTIE 2 – CHOIX DU MODÈLE

Un choix guidé par la performance

- [1] *Semantic Segmentation using Vision Transformers : A survey* par Hans Thisanke, Chamli Deshan, Kavindu Chamith, Sachith Seneviratne, Rajith Vidanaarachchi, Damayanthi Herath (2023)
 - Étude comparative
 - Différents algorithmes de segmentation
 - Tous inspirés du *Vision-Transformer (ViT)* :
 - Inspiré du succès des modèles NLP à Transformers
 - Considérer une image comme une séquence de *patches*, pour réduire la complexité
 - Détecter des relations très fines, et globales, grâce au mécanisme d'attention



Choix guidé par la performance

- *CityScapes* parmi les *datasets* utilisés pour comparer

Model	Variant	Backbone	#Params (M)	Datasets		
				ADE20K	Cityscapes	PASCAL-Context
SETR [5]	SETR- <i>Naïve</i> (16,160k) ^p	ViT-L [‡] [2]	305.67	48.06 / 48.80	-	-
	SETR- <i>PUP</i> (16,160k)	ViT-L [‡]	318.31	48.58 / 50.09	-	-
	SETR- <i>MLA</i> (16,160k)	ViT-L [‡]	310.57	48.64 / 50.28	-	-
	SETR- <i>PUP</i> (16,40k)	ViT-L [‡]	318.31	-	78.39 / 81.57	-
	SETR- <i>PUP</i> (16,80k)	ViT-L [‡]	318.31	-	79.34 / 82.15	-
	SETR- <i>Naïve</i> (16,80k)	ViT-L [‡]	305.67	-	-	52.89 / 53.61
	SETR- <i>PUP</i> (16,80k)	ViT-L [‡]	318.31	-	-	54.40 / 55.27
	SETR- <i>MLA</i> (16,80k)	ViT-L [‡]	310.57	-	-	54.87 / 55.83
Swin * [4]		Swin-T	60	46.1	-	-
		Swin-S	81	49.3	-	-
		Swin-B [‡]	121	51.6	-	-
		Swin-L [‡]	234	53.5	-	-
Segmenter § [11]	Seg-B	DeiT-B [‡] [81]	86	48.05	80.5	53.9
	Seg-B/Mask	DeiT-B [‡]	86	50.08	80.6	55.0
	Seg-L	ViT-L [‡]	307	52.25	80.7	56.5
	Seg-L/Mask	ViT-L [‡]	307	53.63	81.3	59.0
SegFormer [69]		MiT-B0 [‡]	3.4	37.4 / 38.0	76.2 / 78.1	-
		MiT-B1 [‡]	13.1	42.2 / 43.1	78.5 / 80.0	-
		MiT-B2 [‡]	24.2	46.5 / 47.5	81.0 / 82.2	-
		MiT-B3 [‡]	44.0	49.4 / 50.0	81.7 / 83.3	-
		MiT-B4 [‡]	60.8	50.3 / 51.1	82.3 / 83.9	-
		MiT-B5 [‡]	81.4	51.0 / 51.8	82.4 / 84.0	-
PVT * [71]	PVT v1 [71]	PVT-Tiny [‡]	17.0	35.7	-	-
		PVT-Small [‡]	28.2	39.8	-	-
		PVT-Medium [‡]	48.0	41.6	-	-
		PVT-Large [‡]	65.1	42.1	-	-
		PVT-Large [‡] *	65.1	44.8	-	-
	PVT v2 [72]	PVT v2-B0 [‡]	7.6	37.2	-	-
		PVT v2-B1 [‡]	17.8	42.5	-	-
		PVT v2-B2 [‡]	29.1	45.2	-	-
		PVT v2-B3 [‡]	49.0	47.3	-	-
		PVT v2-B4 [‡]	66.3	47.9	-	-
		PVT v2-B5 [‡]	85.7	48.7	-	-
	Twins-PCPVT	Twins-PCPVT-S [‡]	54.6	46.2 / 47.5	-	-
		Twins-PCPVT-B [‡]	74.3	47.1 / 48.4	-	-
		Twins-PCPVT-L [‡]	91.5	48.6 / 49.8	-	-
		Twins-SVT-S [‡]	54.4	46.2 / 47.1	-	-
		Twins-SVT-B [‡]	88.5	47.7 / 48.9	-	-
Twins [73]	Twins-SVT	Twins-SVT-L [‡]	133	48.8 / 50.2	-	-
DPT § [74]	DPT-Hybrid	ViT-Hybrid [‡]	123	49.02	-	60.46
	DPT-Large	ViT-L [‡]	343	47.63	-	-
HRFormer [75]	OCRNet(7,150k) ^p	HRFormer-S	13.5	44.0 / 45.1	-	-
	OCRNet(7,150k)	HRFormer-B	50.3	46.3 / 47.6	-	-
	OCRNet(7,80k)	HRFormer-S	13.5	-	80.0 / 81.0	-
	OCRNet(7,80k)	HRFormer-B	50.3	-	81.4 / 82.0	-
	OCRNet(15,80k)	HRFormer-B	50.3	-	81.9 / 82.6	57.6 / 58.5
	OCRNet(7,60k)	HRFormer-B	50.3	-	-	56.3 / 57.1
	OCRNet(7,60k)	HRFormer-S	13.5	-	-	53.8 / 54.6
Mask2Former [78]		Swin-T	-	47.7 / 49.6	-	-
		Swin-L [‡]	216	56.1 / 57.3	-	-
		Swin-L-FaPN [‡]	-	56.4 / 57.7	-	-
		Swin-L [‡]	216	-	83.3 / 84.3	-
		Swin-B [‡]	-	-	83.3 / 84.5	-



Un choix pragmatique

- Disponible sur la plateforme *HuggingFace*
- *Dispose d'une version « compatible » avec le framework TensorFlow*
- *Classé 2nd en termes de performances*

 **SegFormer**



Model	Variant	Backbone	#Params (M)	Datasets		
				ADE20K	Cityscapes	PASCAL-Context
SETR [5]	SETR- <i>Naïve</i> (16,160k) ^p	ViT-L [‡] [2]	305.67	48.06 / 48.80	-	-
	SETR- <i>PUP</i> (16,160k)	ViT-L [‡]	318.31	48.58 / 50.09	-	-
	SETR- <i>MLA</i> (16,160k)	ViT-L [‡]	310.57	48.64 / 50.28	-	-
	SETR- <i>PUP</i> (16,40k)	ViT-L [‡]	318.31	-	78.39 / 81.57	-
	SETR- <i>PUP</i> (16,80k)	ViT-L [‡]	318.31	-	79.34 / 82.15	-
	SETR- <i>Naïve</i> (16,80k)	ViT-L [‡]	305.67	-	-	52.89 / 53.61
	SETR- <i>PUP</i> (16,80k)	ViT-L [‡]	318.31	-	-	54.40 / 55.27
	SETR- <i>MLA</i> (16,80k)	ViT-L [‡]	310.57	-	-	54.87 / 55.83
Swin * [4]		Swin-T	60	46.1	-	-
		Swin-S	81	49.3	-	-
		Swin-B [‡]	121	51.6	-	-
		Swin-L [‡]	234	53.5	-	-
Segmenter § [11]	Seg-B	DeiT-B [‡] [81]	86	48.05	80.5	53.9
	Seg-B/Mask	DeiT-B [‡]	86	50.08	80.6	55.0
	Seg-L	ViT-L [‡]	307	52.25	80.7	56.5
	Seg-L/Mask	ViT-L [‡]	307	53.63	81.3	59.0
SegFormer [69]		MiT-B0 [‡]	3.4	37.4 / 38.0	76.2 / 78.1	-
		MiT-B1 [‡]	13.1	42.2 / 43.1	78.5 / 80.0	-
		MiT-B2 [‡]	24.2	46.5 / 47.5	81.0 / 82.2	-
		MiT-B3 [‡]	44.0	49.4 / 50.0	81.7 / 83.3	-
		MiT-B4 [‡]	60.8	50.3 / 51.1	82.3 / 83.9	-
		MiT-B5 [‡]	81.4	51.0 / 51.8	82.4 / 84.0	-
PVT * [71]	PVT v1 [71]	PVT-Tiny [‡]	17.0	35.7	-	-
		PVT-Small [‡]	28.2	39.8	-	-
		PVT-Medium [‡]	48.0	41.6	-	-
		PVT-Large [‡]	65.1	42.1	-	-
		PVT-Large [‡] *	65.1	44.8	-	-
	PVT v2 [72]	PVT v2-B0 [‡]	7.6	37.2	-	-
		PVT v2-B1 [‡]	17.8	42.5	-	-
		PVT v2-B2 [‡]	29.1	45.2	-	-
		PVT v2-B3 [‡]	49.0	47.3	-	-
		PVT v2-B4 [‡]	66.3	47.9	-	-
		PVT v2-B5 [‡]	85.7	48.7	-	-
	Twins-PCPVT	Twins-PCPVT-S [‡]	54.6	46.2 / 47.5	-	-
		Twins-PCPVT-B [‡]	74.3	47.1 / 48.4	-	-
		Twins-PCPVT-L [‡]	91.5	48.6 / 49.8	-	-
Twins [73]	Twins-SVT	Twins-SVT-S [‡]	54.4	46.2 / 47.1	-	-
		Twins-SVT-B [‡]	88.5	47.7 / 48.9	-	-
		Twins-SVT-L [‡]	133	48.8 / 50.2	-	-
DPT § [74]	DPT-Hybrid	ViT-Hybrid [‡]	123	49.02	-	60.46
	DPT-Large	ViT-L [‡]	343	47.63	-	-
HRFormer [75]	OCRNet(7,150k) ^p	HRFormer-S	13.5	44.0 / 45.1	-	-
	OCRNet(7,150k)	HRFormer-B	50.3	46.3 / 47.6	-	-
	OCRNet(7,80k)	HRFormer-S	13.5	-	80.0 / 81.0	-
	OCRNet(7,80k)	HRFormer-B	50.3	-	81.4 / 82.0	-
	OCRNet(15,80k)	HRFormer-B	50.3	-	81.9 / 82.6	57.6 / 58.5
	OCRNet(7,60k)	HRFormer-B	50.3	-	-	56.3 / 57.1
	OCRNet(7,60k)	HRFormer-S	13.5	-	-	53.8 / 54.6
Mask2Former [78]		Swin-T	-	47.7 / 49.6	-	-
		Swin-L [‡]	216	56.1 / 57.3	-	-
		Swin-L-FaPN [‡]	-	56.4 / 57.7	-	-
		Swin-L [‡]	216	-	83.3 / 84.3	-
		Swin-B [‡]	-	-	83.3 / 84.5	-



Autres ressources

- [2] *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers*
par Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, Ping Luo (2021)
 - Article ayant introduit l'algorithme
 - Description de sa structure



Autres ressources

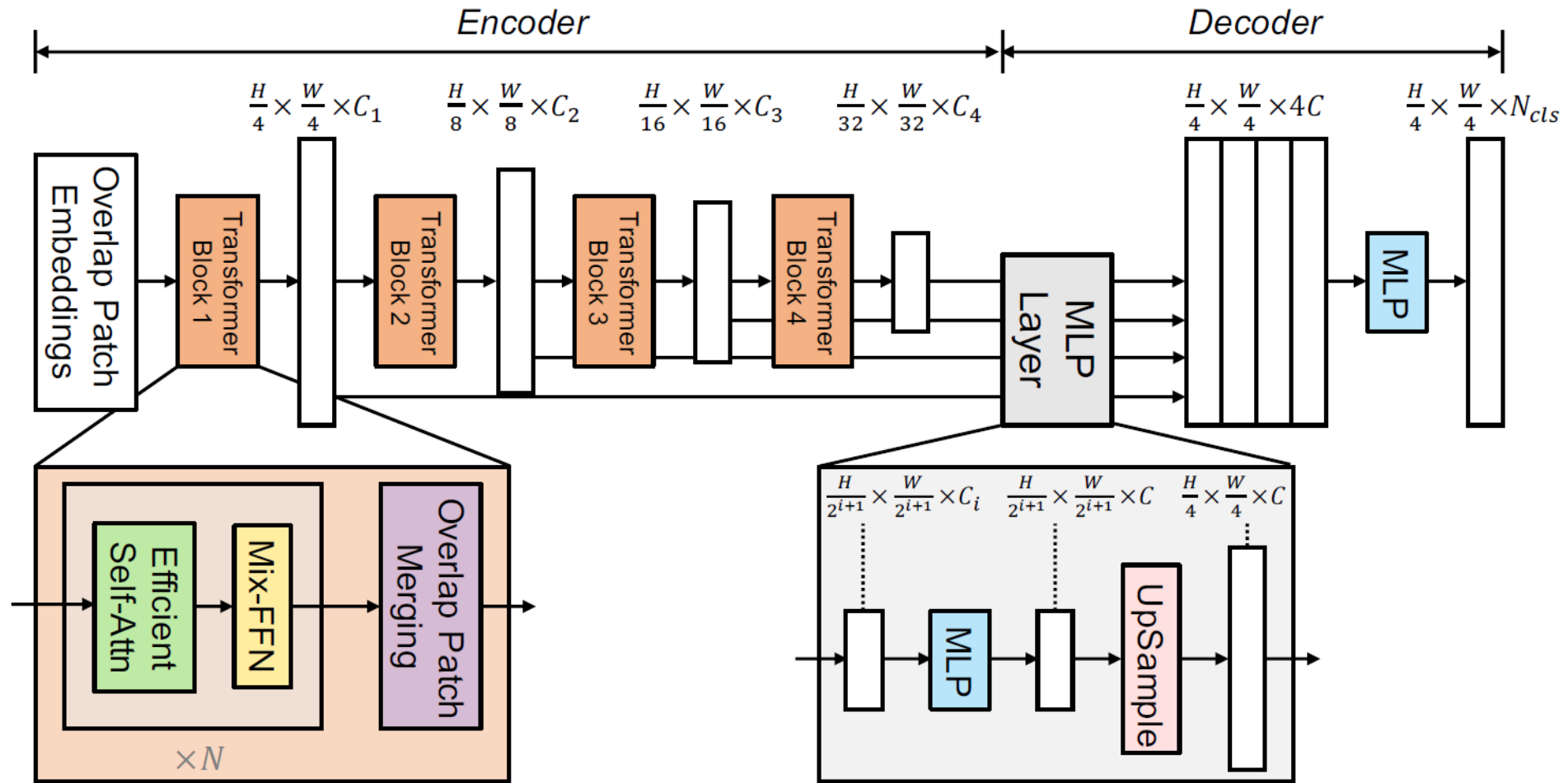
- [3] *MedAI #32: Simple & Efficient Design for Semantic Segmentation with Transformers*
<https://www.youtube.com/watch?v=Yf9fNn1fWy8&t=380s>
par Enze Xie
- [4] *Semantic segmentation with SegFormer and Hugging Face Transformers*
<https://keras.io/examples/vision/segformer>
par Sayak Paul (2023)
- [5] *Semantic segmentation for Attribution methods tutorial*
<https://colab.research.google.com/drive/1AHg7KO1fCOX5nZLGZfxkZ2-DLPPdSfbX#scrollTo=0c7d9d33-2de5-4f03-a793-a76dd2b1e238>
par Antonin Poché (2023)



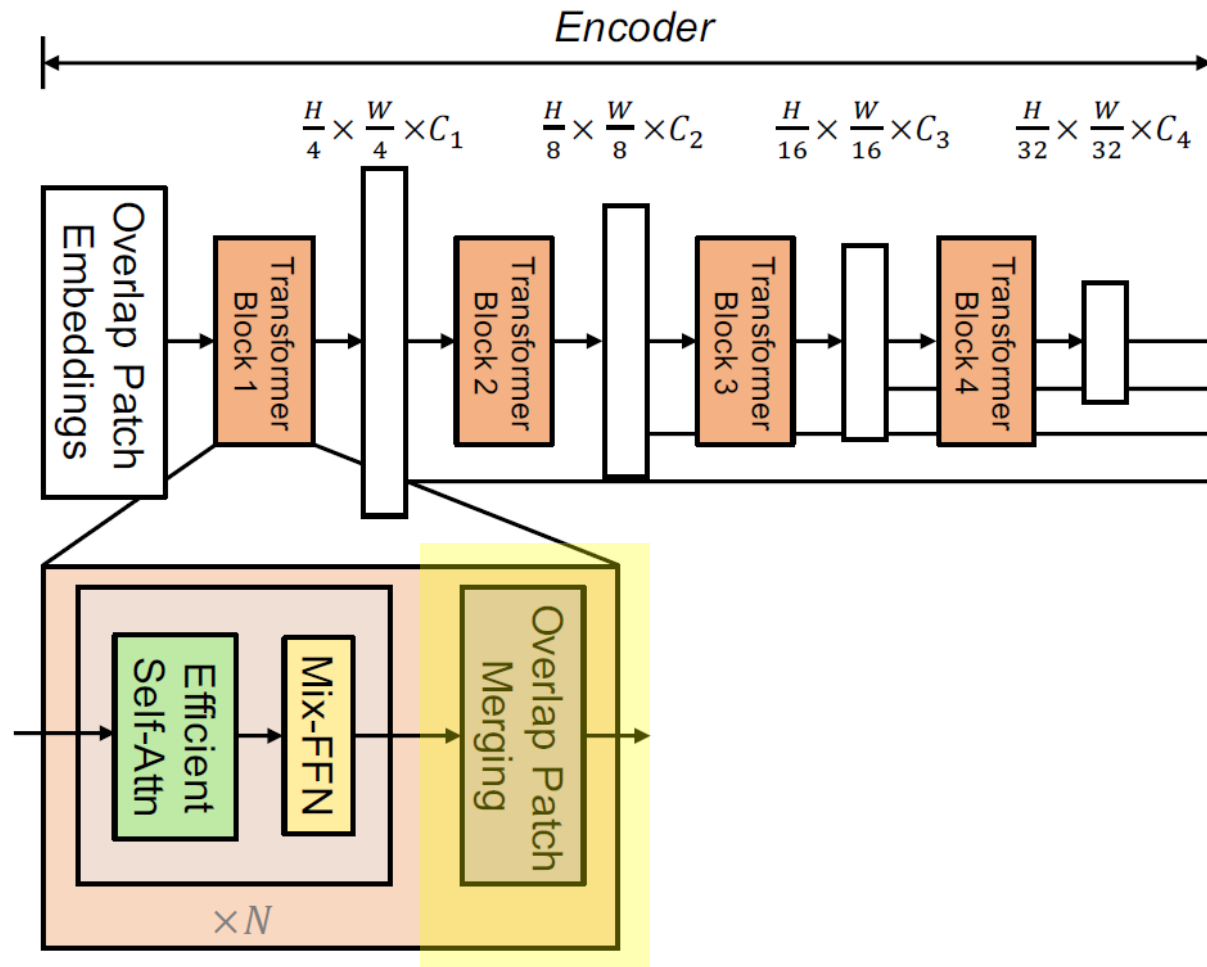


PARTIE 3 – DESCRIPTION DU MODÈLE

Architecture classique *Encoder / Decoder*



Hierarchical Transformer Encoder



Overlapped Patch Merging

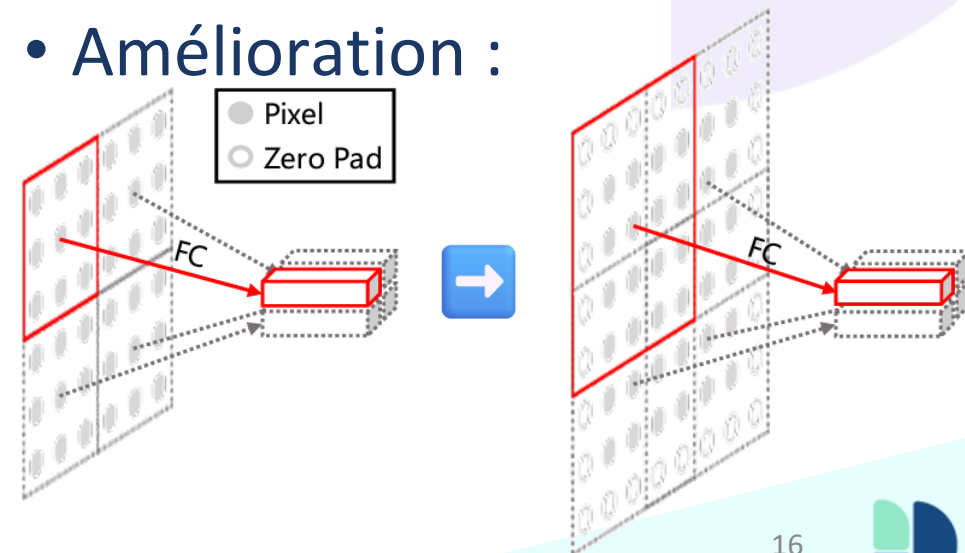
- Principe initial :

Patch $N \times N \times 3$

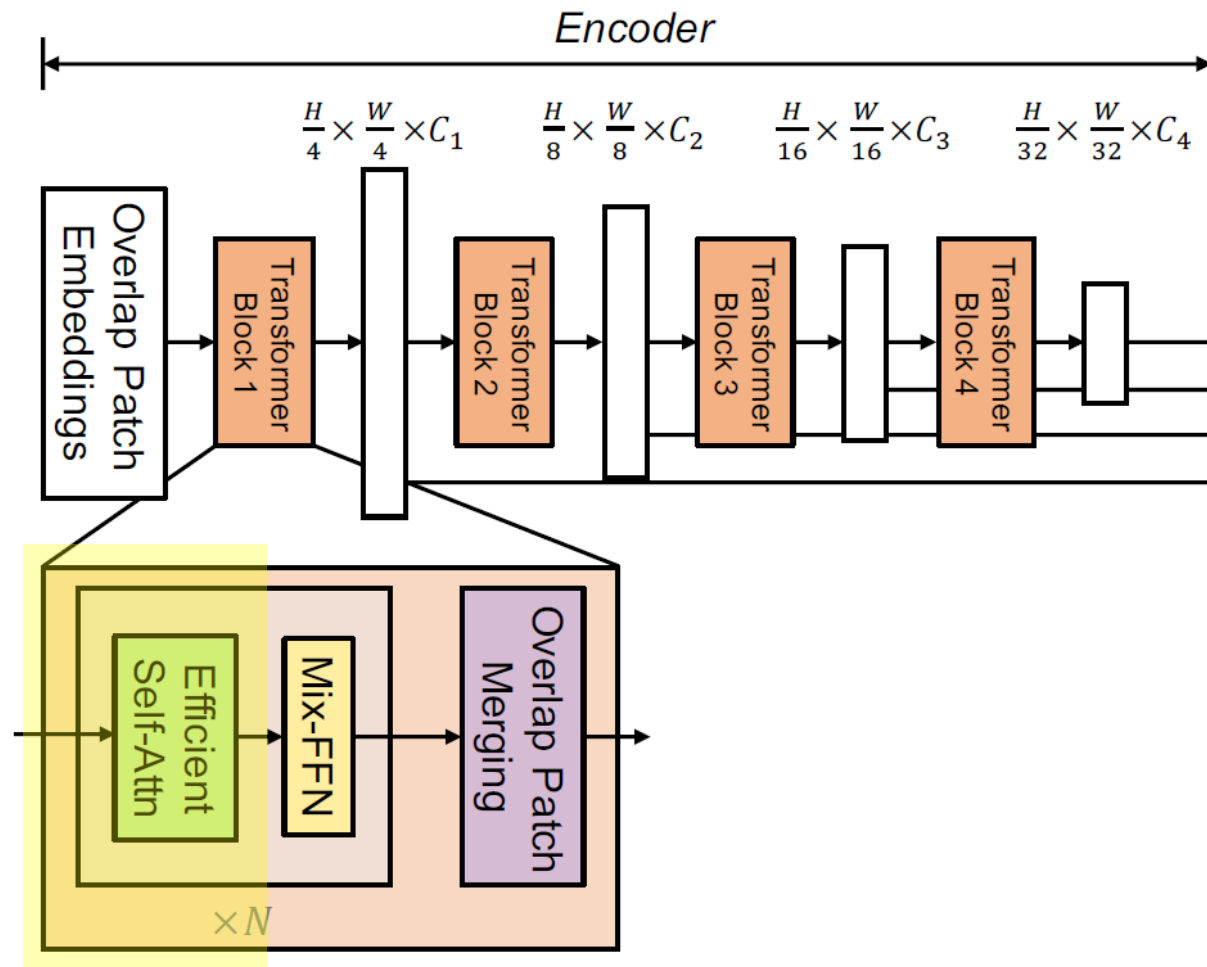


Vecteur $1 \times 1 \times C_i$

- Amélioration :



Hierarchical Transformer Encoder



Efficient Self-Attention

- Complexité initiale : $O(N^2)$ avec $N = H_i \times W_i$



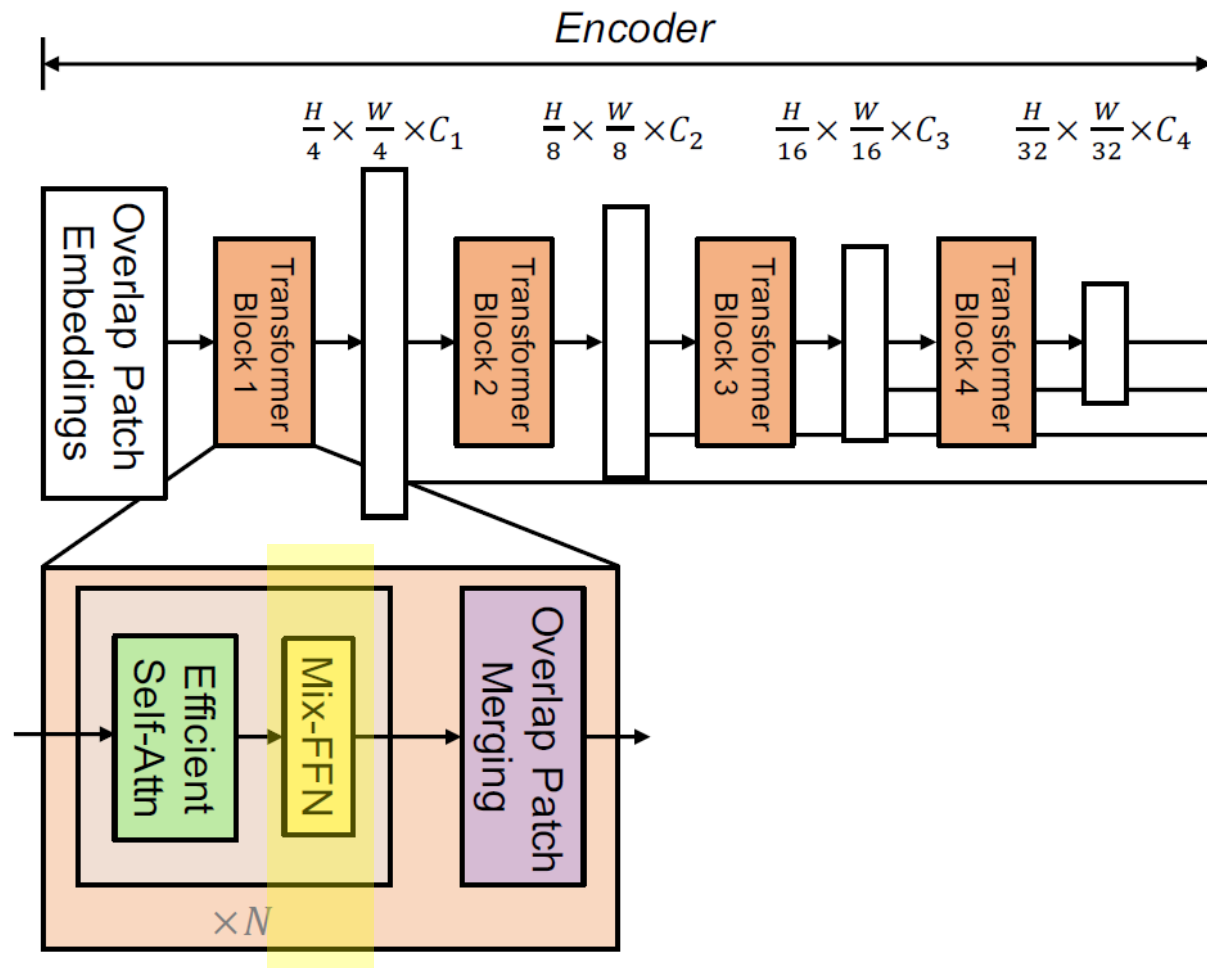
- Réduction : $O(N^2/R)$

Méthode :

- convolution avec $Kernel = Stride = \sqrt{R}$
- couche linéaire

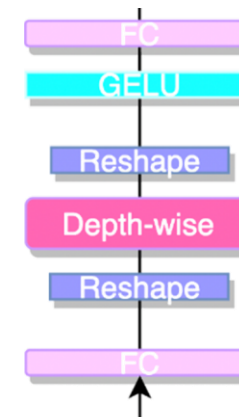


Hierarchical Transformer Encoder



Mix-FFN

- Solution initiale : *positional encoding* pour apprendre de la position des patches
- Problème : performance diminue à cause des changements de résolution
- Solution :



$$\mathbf{x}_{out} = \text{MLP}(\text{GELU}(\text{Conv}_{3 \times 3}(\text{MLP}(\mathbf{x}_{in})))) + \mathbf{x}_{in}$$

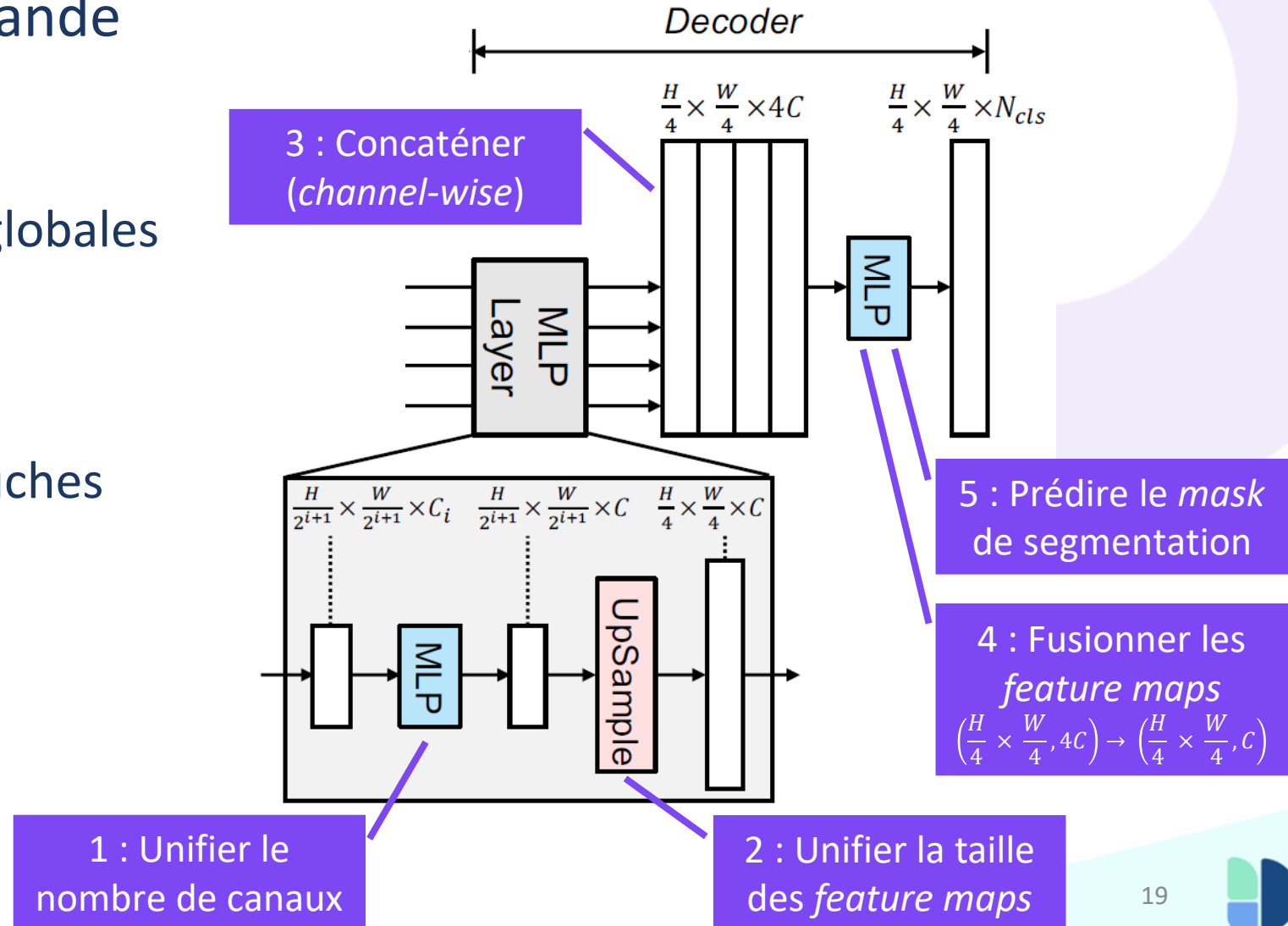


Lightweight All-MLP Decoder

- Des *features maps* de grande qualité :
 - Différentes échelles
 - Informations locales ET globales



- Un *decoder* très simple :
 - Sans les nombreuses couches convolutives habituelles
 - Basé sur des blocs *MLP*





PARTIE 4 – RÉSULTAT DE LA MODÉLISATION

Métrique d'évaluation

- Mean IoU score (indice de Jaccard) :

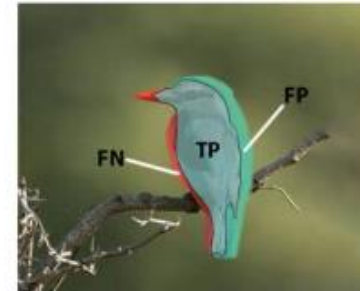
$$IoU = \frac{TP}{(TP + FP + FN)}$$



Ground Truth Mask



Predicted Mask



- Training time

Quel *SegFormer* ? Quelle *Baseline* ?

- Modèle sélectionné sur le projet précédent : *Unet-Resnet*
- Plusieurs versions disponibles pour chacun :
 - *SegFormer* : 5 variantes *B1* à *B5*
 - *Resnet* : 5 variantes *Resnet18* à *Resnet152*
- Choix du *B1* pour le *SegFormer* (contraintes du matériel)
- Sélectionner des modèles de tailles équivalentes :

SegFormer B1

```
=====
Total params: 13679816 (52.18 MB)
Trainable params: 13679304 (52.18 MB)
Non-trainable params: 512 (2.00 KB)
```



Unet-Resnet18

```
=====
Total params: 14341585 (54.71 MB)
Trainable params: 14331659 (54.67 MB)
Non-trainable params: 9926 (38.77 KB)
```

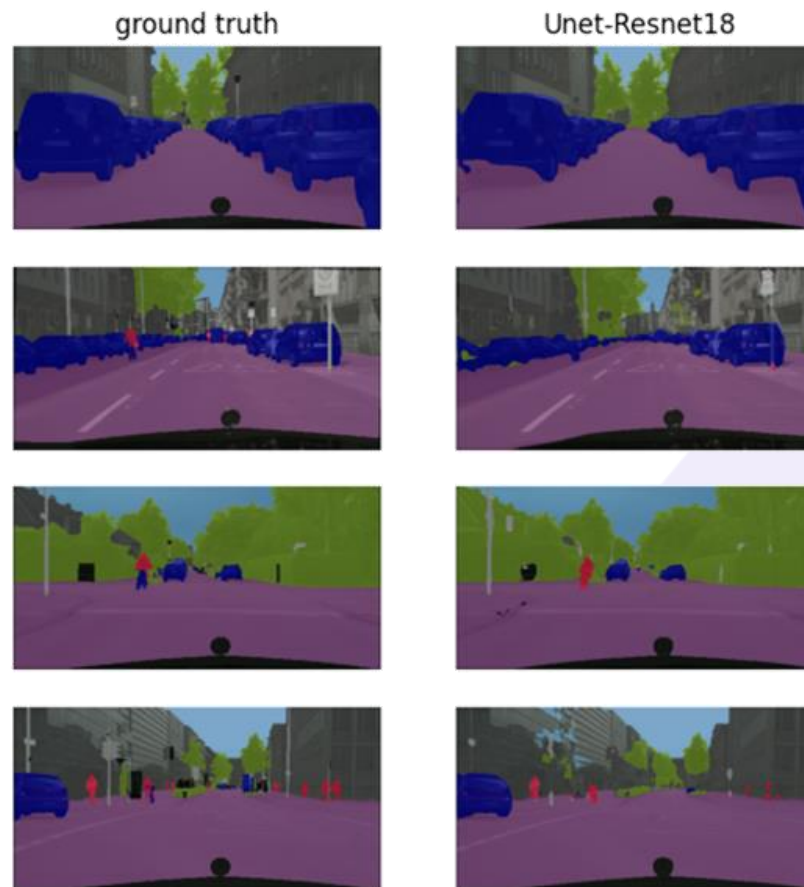
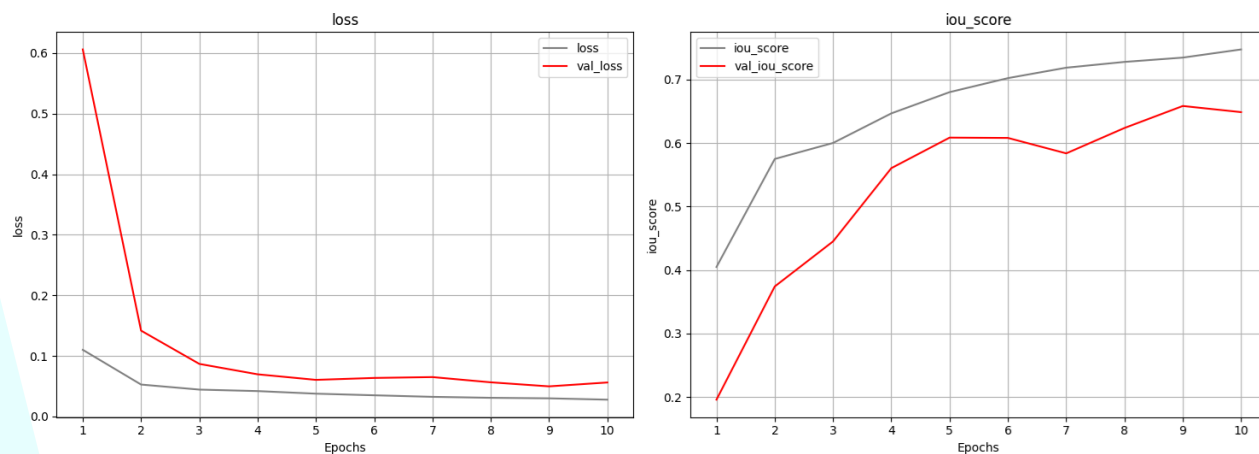
Preprocessing

- Utilisation d'un Générateur de données
- Limiter taille images :
 - *Unet-Resnet* : 256×512
 - *Segformer* : 384×384
- Augmentation de données
- Preprocess :
 - *Unet-Resnet* : Zero-centering et $RGB \rightarrow BGR$
 - *SegFormer* : Normalization
- Simplification des catégories : $32 \rightarrow 8$
- Mise en *batches*
- *Shuffle* (si besoin)

```
# define augmentations for training
list_of_transforms = [
    A.OneOf(
        [
            A.FancyPCA(p=1, alpha=1),
            A.HueSaturationValue(p=1, hue_shift_limit=20, sat_shift_limit=30, val_shift_limit=20),
            A.ColorJitter(p=1, brightness=0.2, contrast=0.2, saturation=0.2),
        ],
        p=0.5
    ),
    A.RandomShadow(
        p=0.5,
        shadow_roi=(
            0, 0.4,
            1, 1
        ),
        num_shadows_lower=1,
        num_shadows_upper=4,
        shadow_dimension=5
    ),
    A.CoarseDropout(
        p=0.5,
        min_holes=2,
        max_holes=8,
        min_height=0.05,
        max_height=0.1,
        min_width=0.025,
        max_width=0.05
    )
]
```

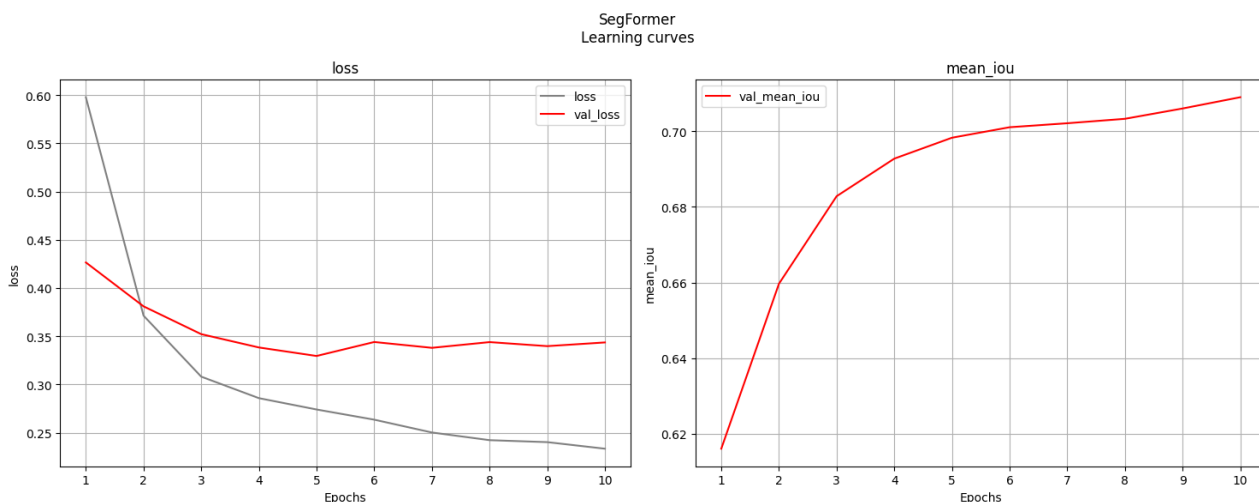
Baseline

Unet-Resnet18
Learning curves

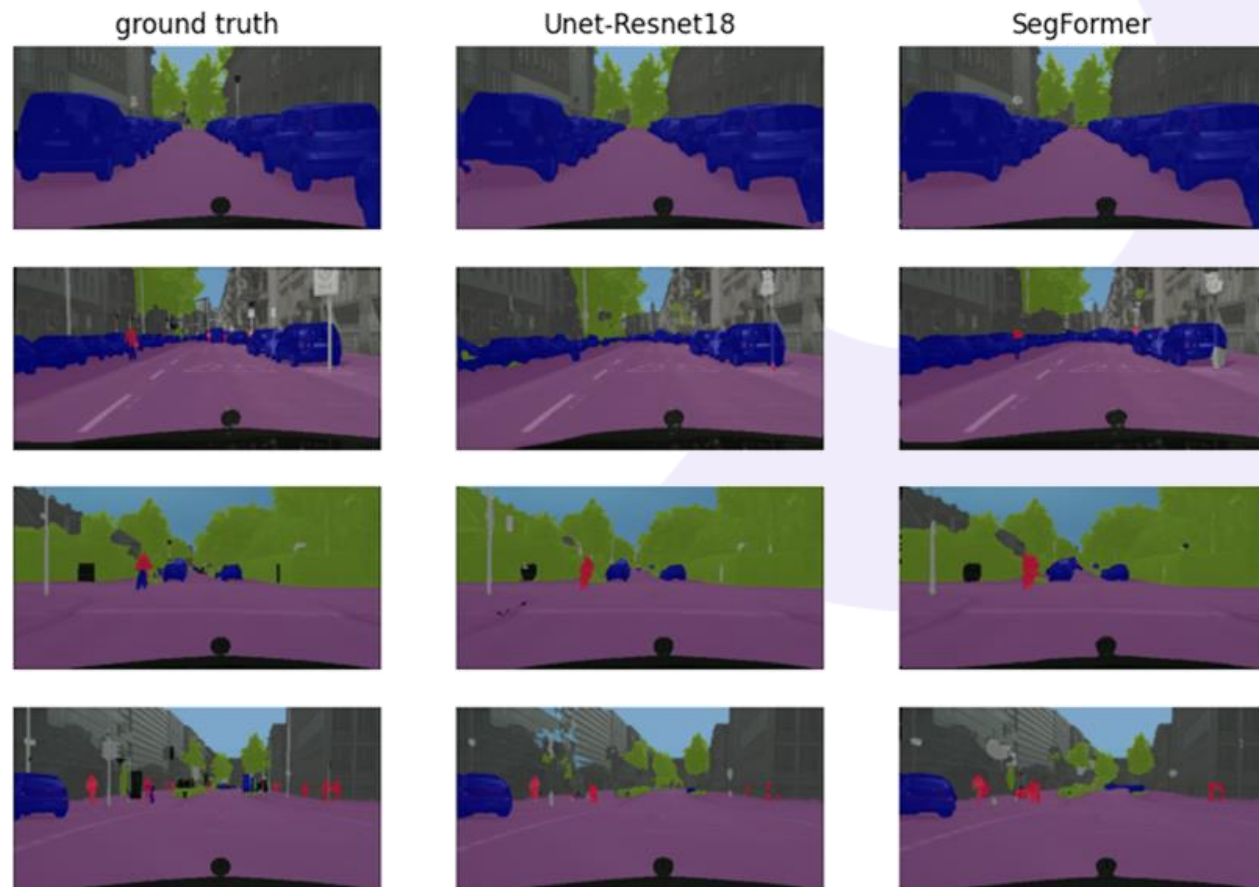


model	backbone	val iou	training time
Unet	Resnet18	0.65	3 h

SegFormer

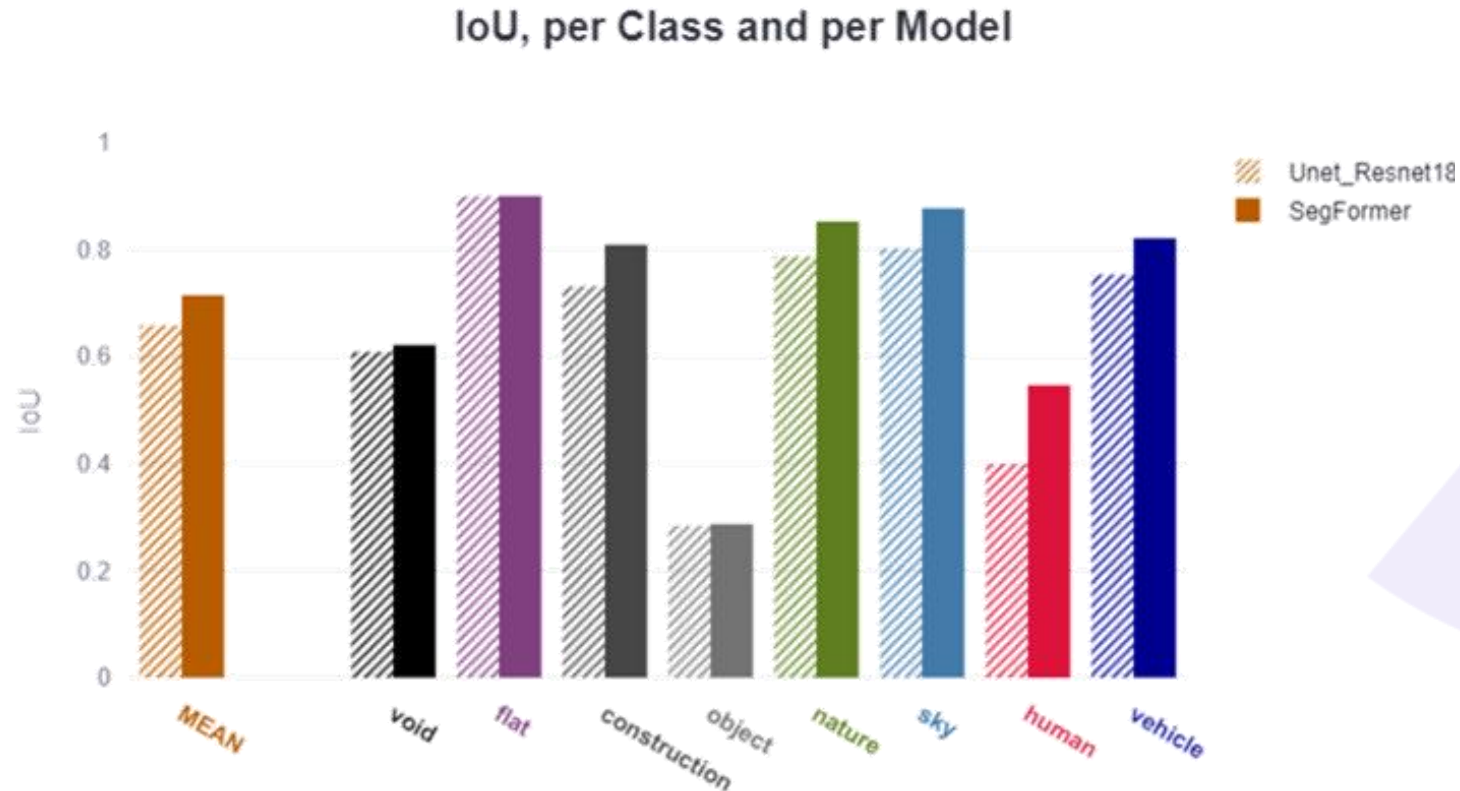


- De meilleurs résultats !
- Au prix d'un temps d'entraînement bien plus conséquent



model	backbone	val iou	training time
Unet	Resnet18	0.65	3 h
SegFormer	MIT-B1	0.71	15 h

Et par classe ?



- Une prédominance sur toutes les catégories
- Très marquée sur les humains

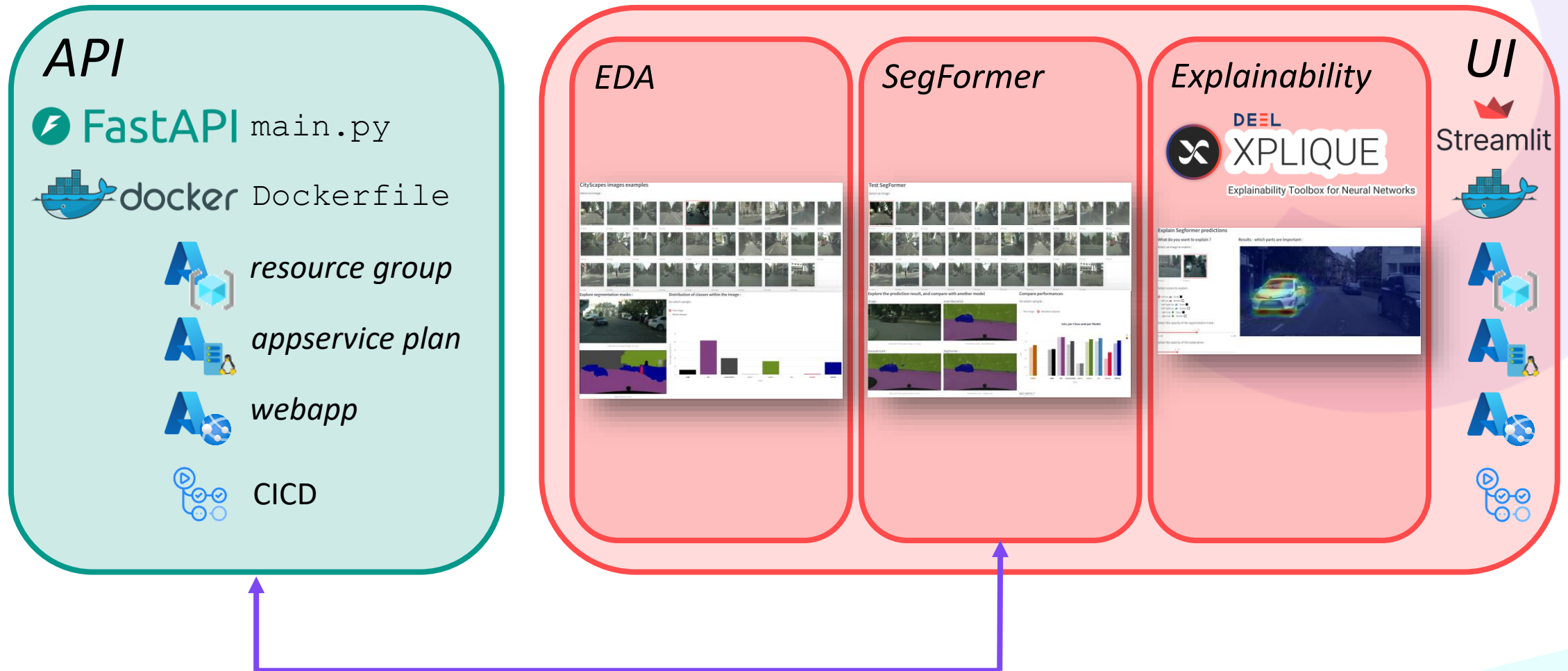
model	backbone	val iou	training time
Unet	Resnet18	0.65	3 h
SegFormer	MIT-B1	0.71	15 h



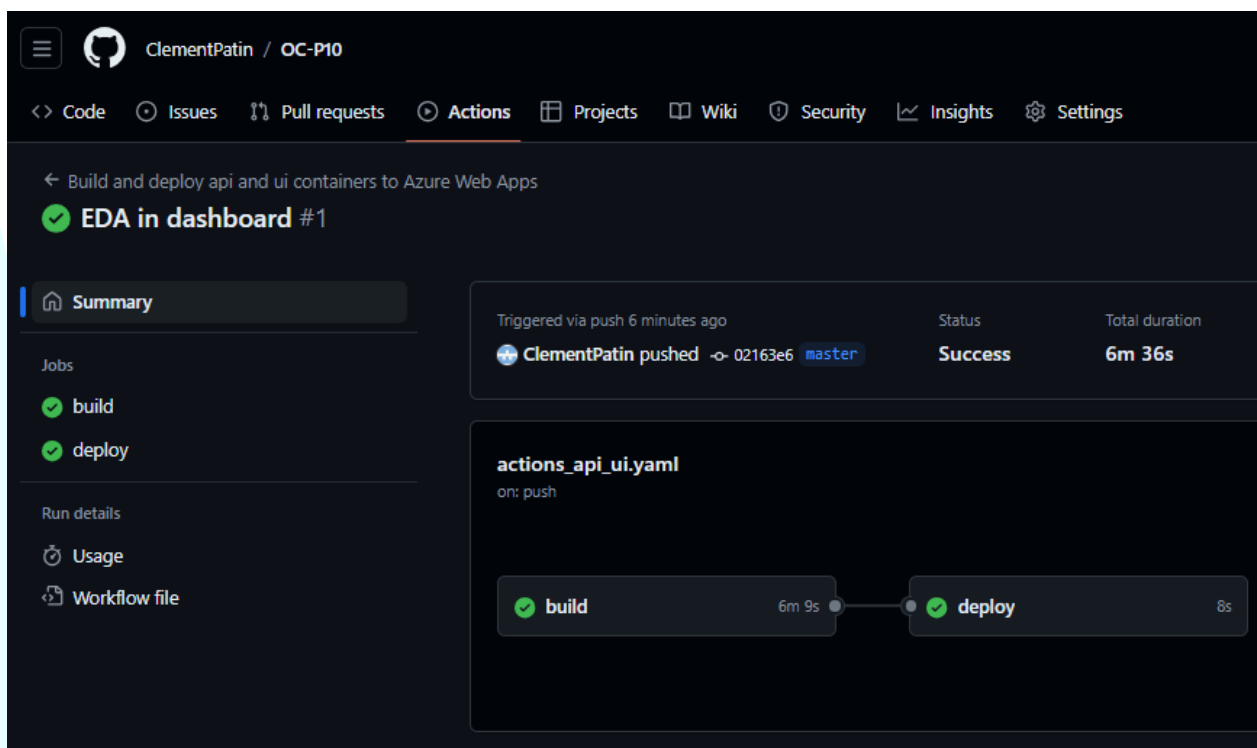


PARTIE 5 – API ET DASHBOARD

Architecture



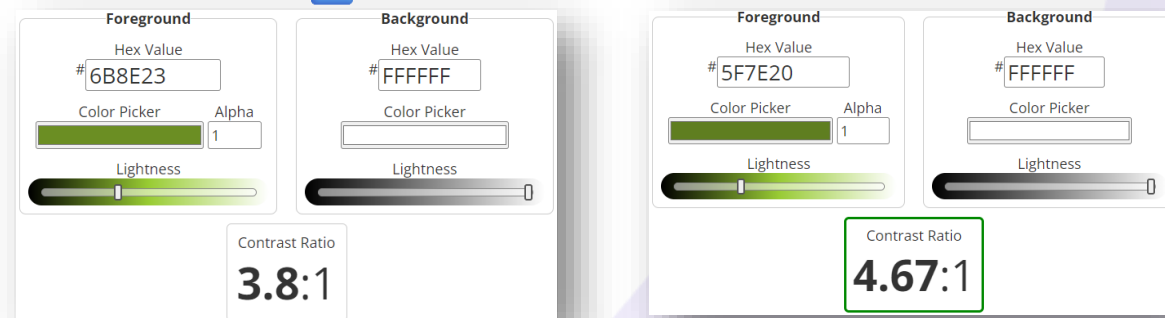
Déploiement continu



The screenshot shows a GitHub repository named 'ClementPatin / OC-P10'. The 'Actions' tab is selected, displaying a workflow named 'EDA in dashboard #1'. The workflow summary indicates it was triggered by a push to the master branch and completed successfully. The workflow steps are 'build' (6m 9s) and 'deploy' (8s). The workflow file is named 'actions_api_ui.yaml' and is triggered on push.

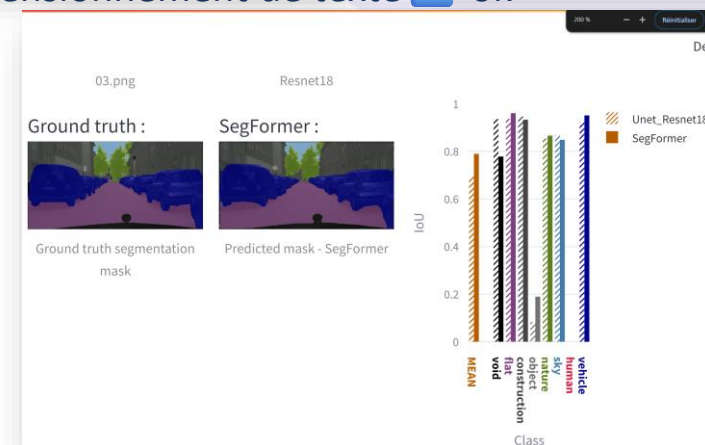
Accessibilité - WCAG Streamlit

- 1.1.1 Contenu non textuel → captions, labels
- 1.4.1 Utilisation de la couleur → graphique avec patterns
- 1.4.3 Contraste → *Contrast Checker WebAIM*



The image shows the Contrast Checker WebAIM interface. It displays two sets of color contrast settings. The first set shows a foreground color of #6B8E23 and a background color of #FFFFFF, resulting in a contrast ratio of 3.8:1. The second set shows a foreground color of #5F7E20 and a background color of #FFFFFF, resulting in a contrast ratio of 4.67:1. Both sets include color pickers, alpha value inputs, and lightness sliders.

- 1.4.4 Redimensionnement de texte → ok



- 2.4.2 Titre de page →
`st.set_page_config(page_title="SegFormer - CityScapes")`



CONCLUSION

The background image shows the interior of a modern car. The steering wheel is visible on the left, and the dashboard features a large digital display showing a navigation map and various driving metrics. The car is positioned on a road with other vehicles in the distance. The word 'CONCLUSION' is prominently displayed in the center of the image in a large, blue, sans-serif font.