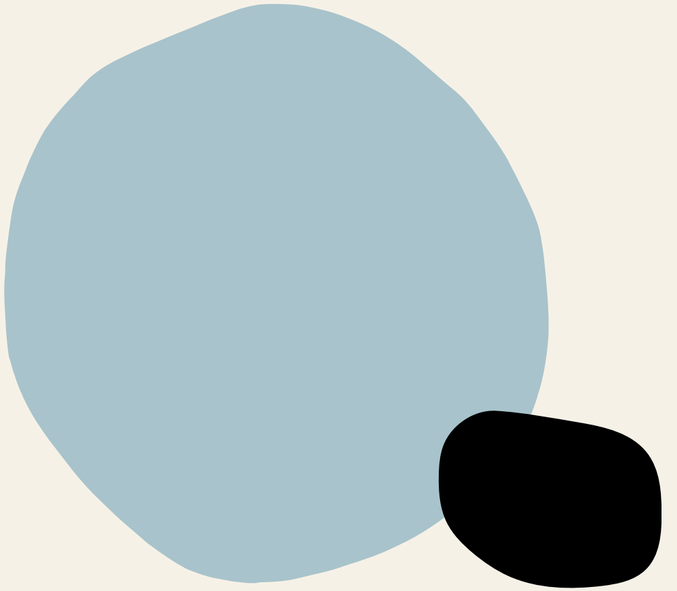
Deep learning for image segmentation

Lorenzo Azerine

Louis Carron



Friday, 17th, January

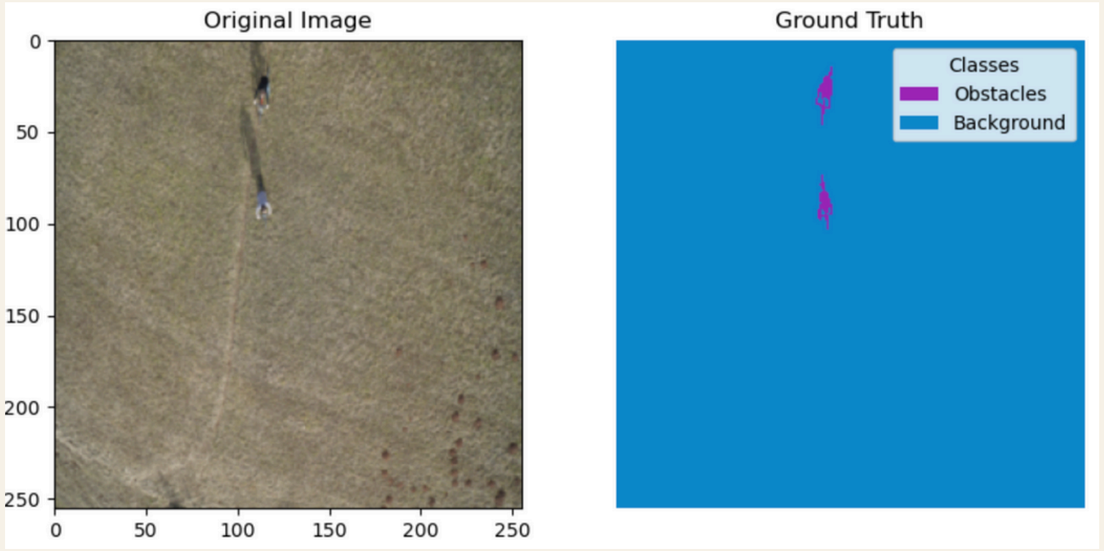
CONTENT

I - INTRODUCTION

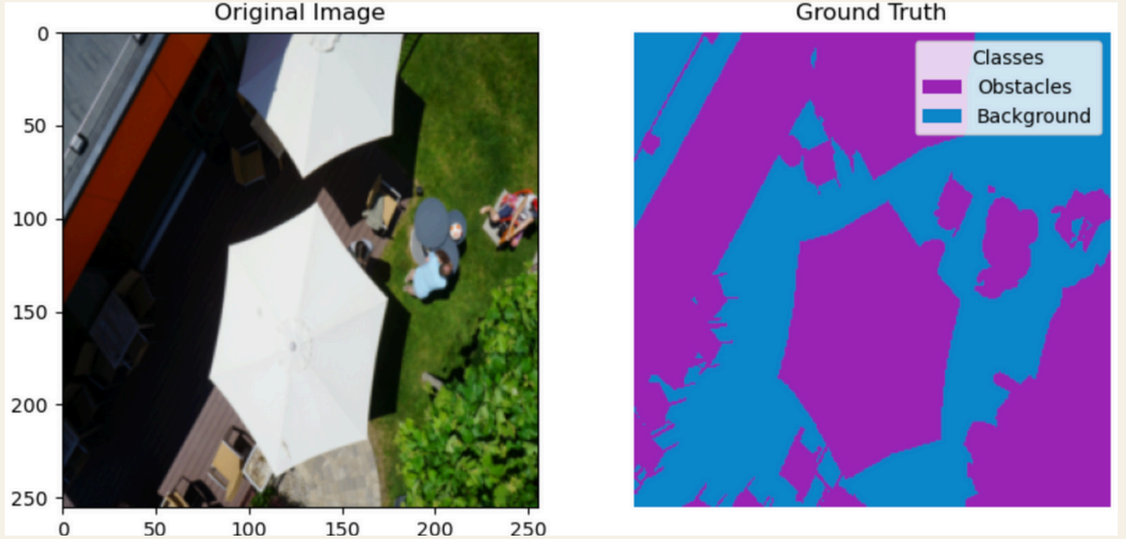
Problem Challenges, Protocol, Augmentation, Metrics

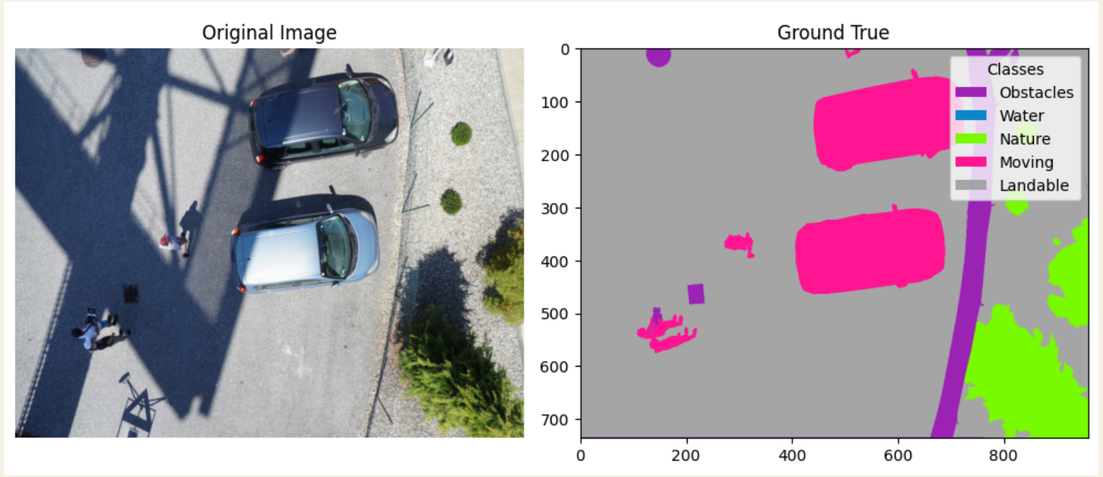
INTRODUCTION

Problem challenges

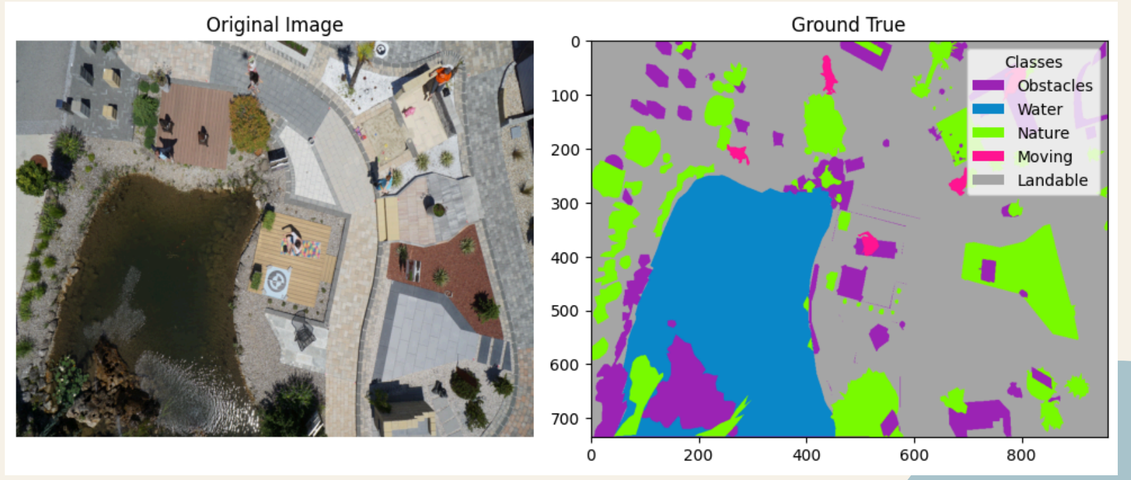


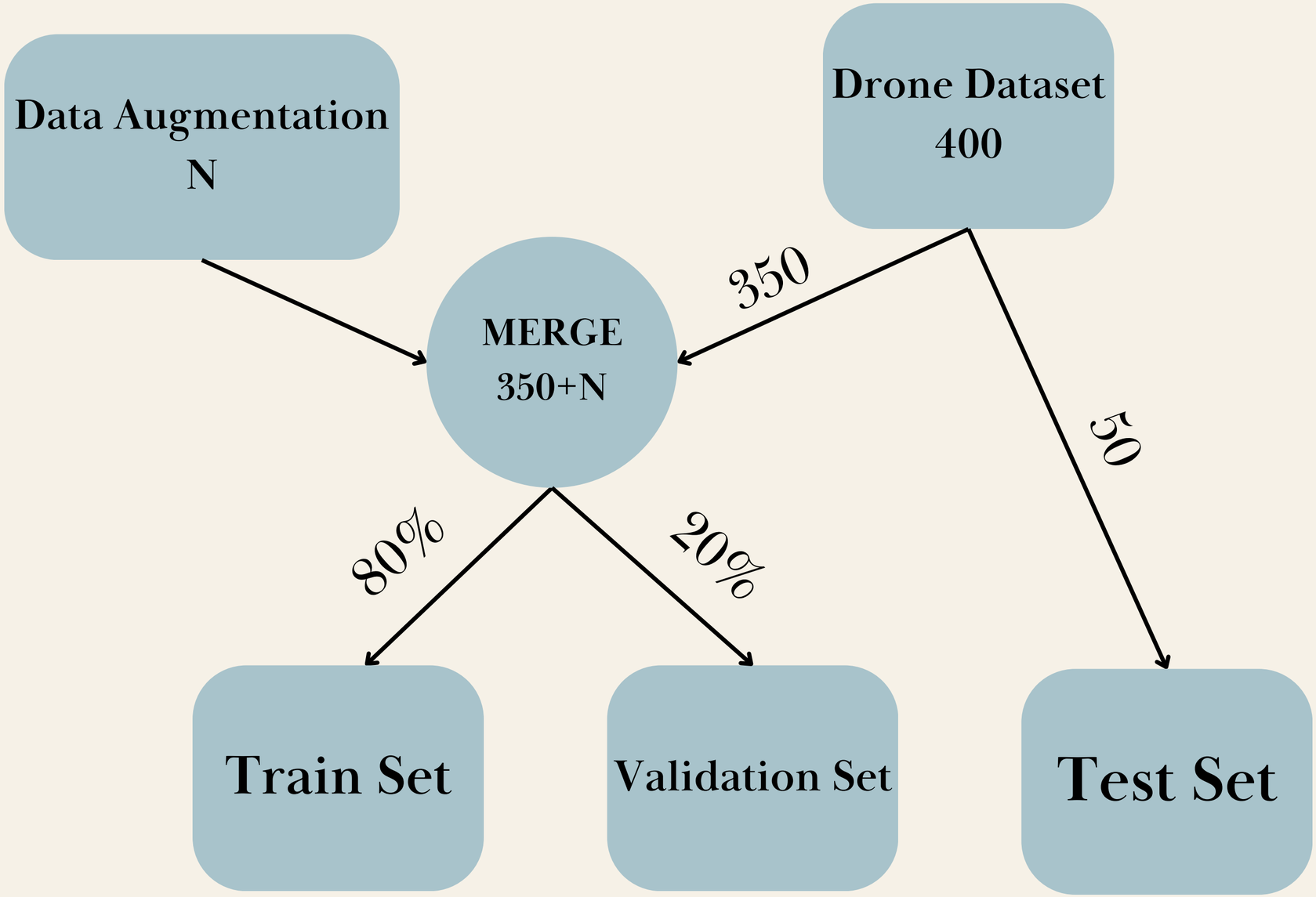
Binary





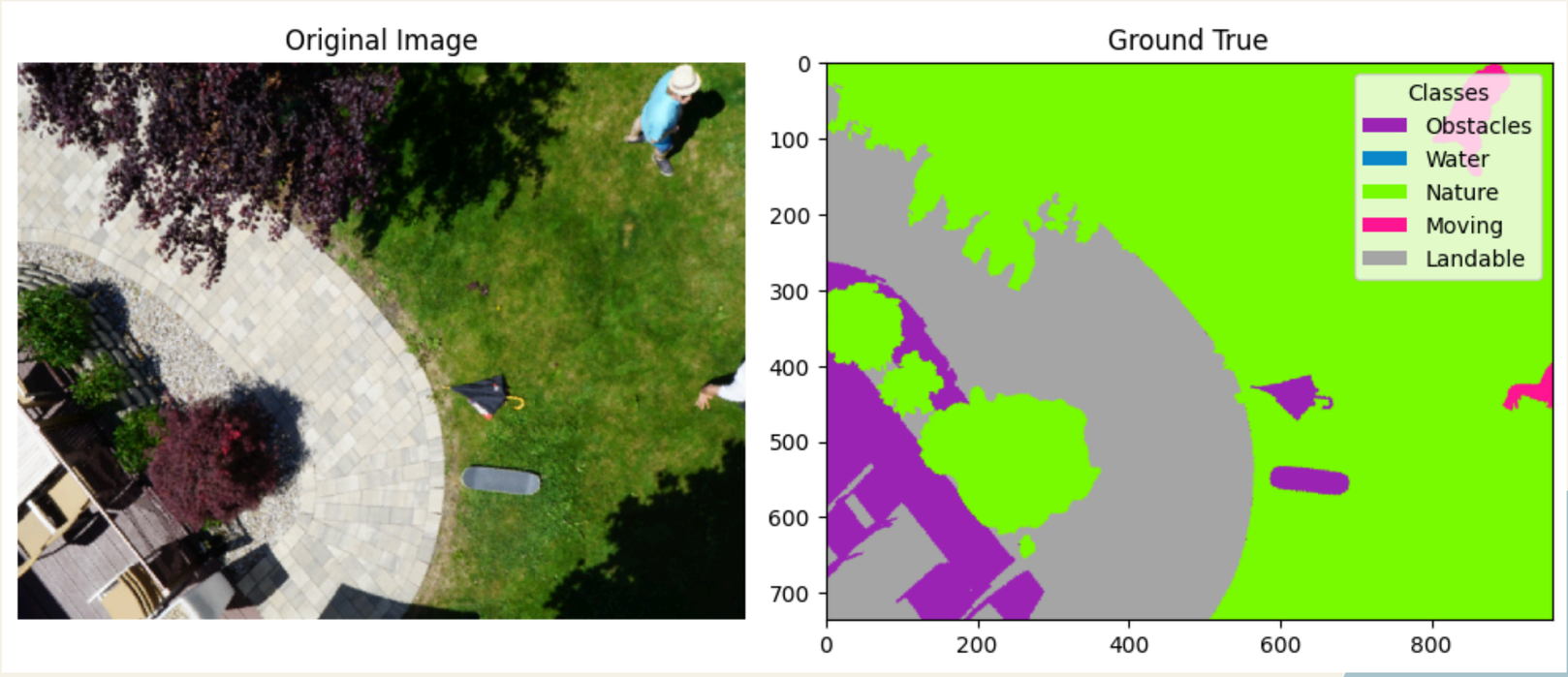
Multi




Data Augmentation
N
Drone Dataset
400
MERGE
350+N
350
50
80%
20%
Train Set
Validation Set
Test Set

Augmentation

- *Random **Horizontal** and **Vertical** Flip (p=0.5)*
- *Random **Rotation**, uniform in [-30, 30]°*
- *Centered **Crop***
- ***Nearest** Resize*

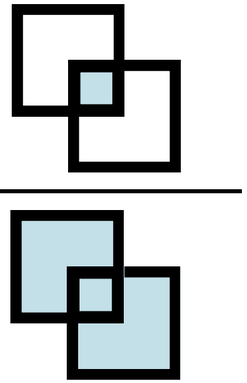


Mean Pixel Accuracy :

$$\mathrm{mPA} = \frac{1}{C}\sum_{c=1}^{C}\frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c + \mathrm{FP}_c}$$

Mean Intersection over Union :

$$\mathbf{mIoU} = \frac{1}{C}\sum_{c=1}^{C}$$



IoU put the emphasis on the quality of the outputs

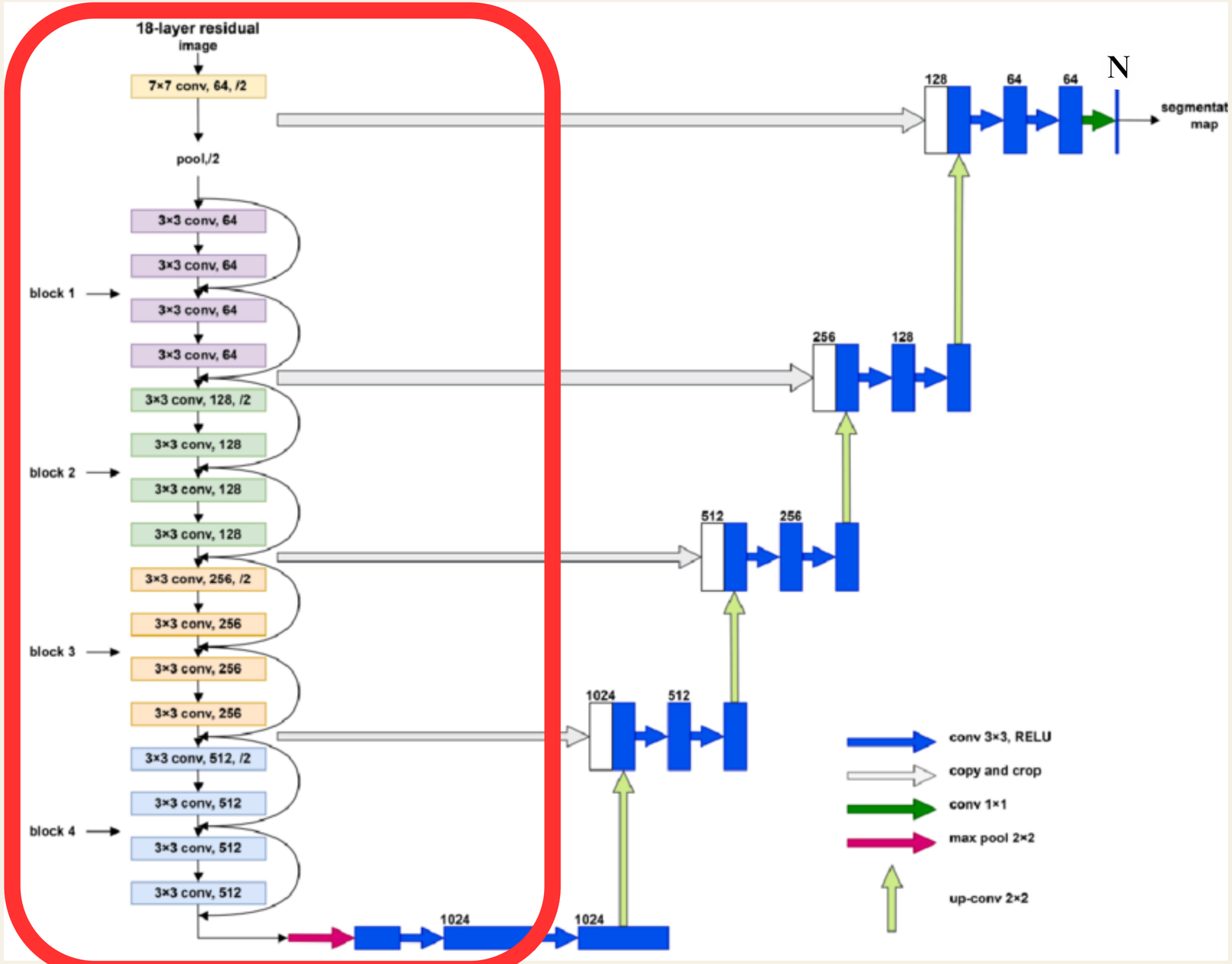
Both guaranty equity between classes, reliable metrics when dealing with unbalance data

II - U-NET

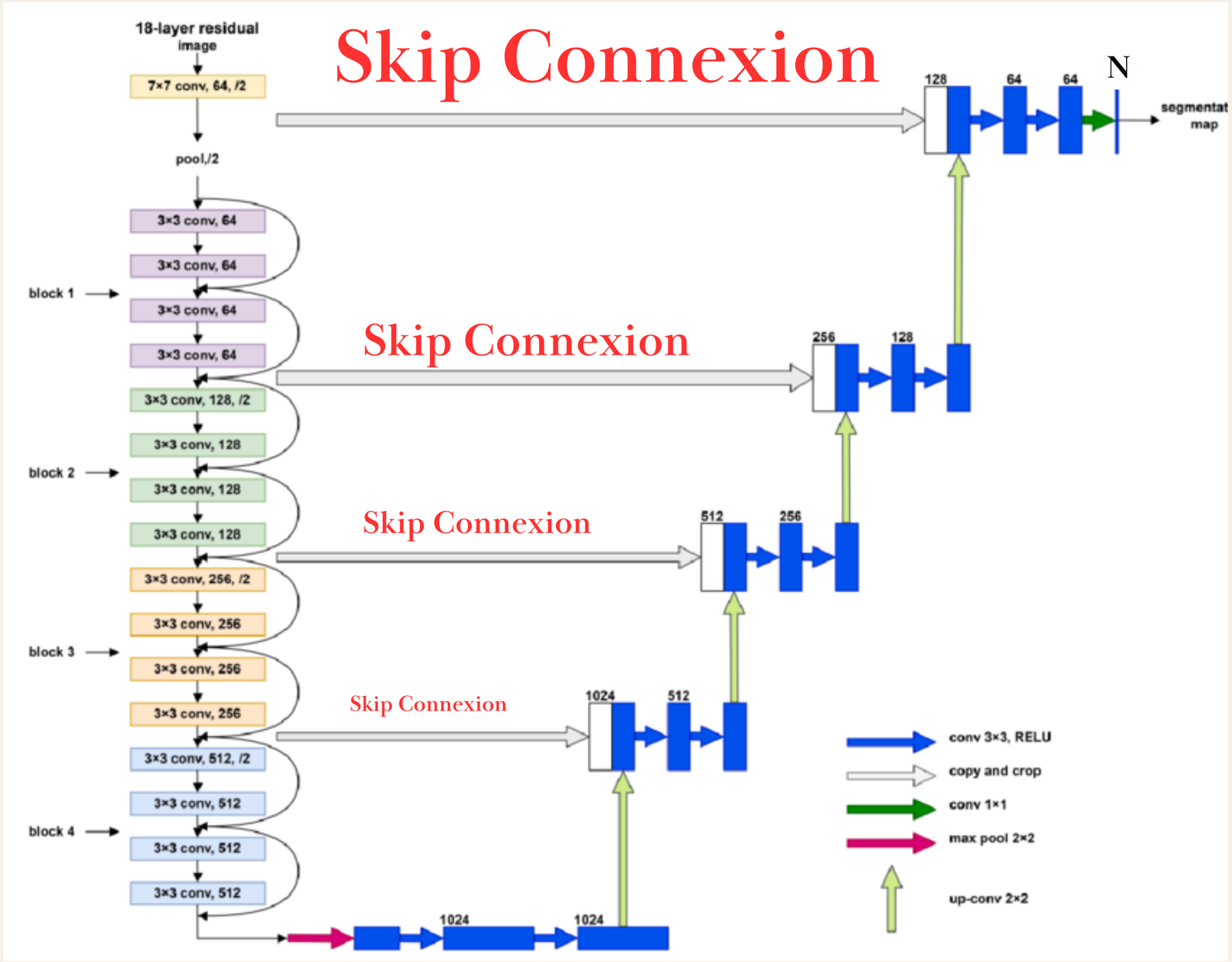
Overview, Training, Outputs

Overview

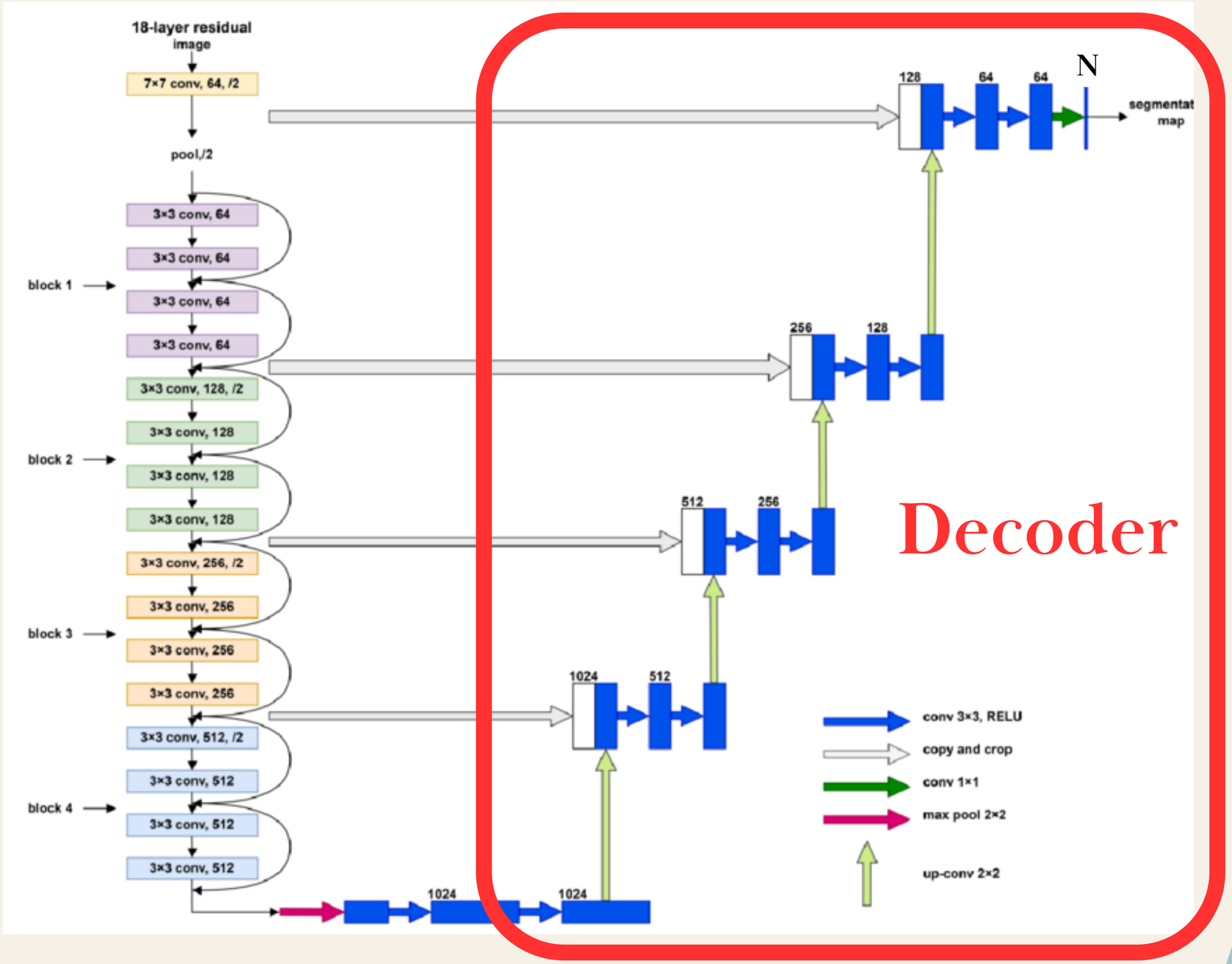
ResNet18[1]



Overview


18-layer residual image
7×7 conv, 64, /2
pool,/2
3×3 conv, 64
3×3 conv, 64
block 1
3×3 conv, 64
3×3 conv, 64
3×3 conv, 128, /2
3×3 conv, 128
block 2
3×3 conv, 128
3×3 conv, 128
3×3 conv, 256, /2
3×3 conv, 256
block 3
3×3 conv, 256
3×3 conv, 256
3×3 conv, 512, /2
3×3 conv, 512
block 4
3×3 conv, 512
3×3 conv, 512
1024
1024
Skip Connexion
Skip Connexion
Skip Connexion
Skip Connexion
128
64
64
N
segmentat map
256
128
512
256
1024
512
conv 3×3, RELU
copy and crop
conv 1×1
max pool 2×2
up-conv 2×2

Overview


18-layer residual
image
7×7 conv, 64, /2
pool,/2
3×3 conv, 64
3×3 conv, 64
block 1
3×3 conv, 64
3×3 conv, 64
3×3 conv, 128, /2
3×3 conv, 128
block 2
3×3 conv, 128
3×3 conv, 128
3×3 conv, 256, /2
3×3 conv, 256
block 3
3×3 conv, 256
3×3 conv, 256
3×3 conv, 512, /2
3×3 conv, 512
block 4
3×3 conv, 512
3×3 conv, 512
1024
1024
1024
512
512
256
256
128
128
64
64
N
segmentat
map
Decoder
conv 3×3, RELU
copy and crop
conv 1×1
max pool 2×2
up-conv 2×2

Components :

- Loss function : **Tversky Loss designed to avoid miss detection (FN).**

$$\text{Tversky Index} = \frac{\text{TP}}{\text{TP} + \alpha \cdot \text{FP} + \beta \cdot \text{FN}}$$

$$\text{Tversky Loss} = 1 - \text{Tversky Index}$$

- Optimizer : **Adam**
- Scheduler : **Reduce On Plateau** with **factor 0.1**
- Batch size : 32

Encoder weights :

- **Imagenet** DataBase

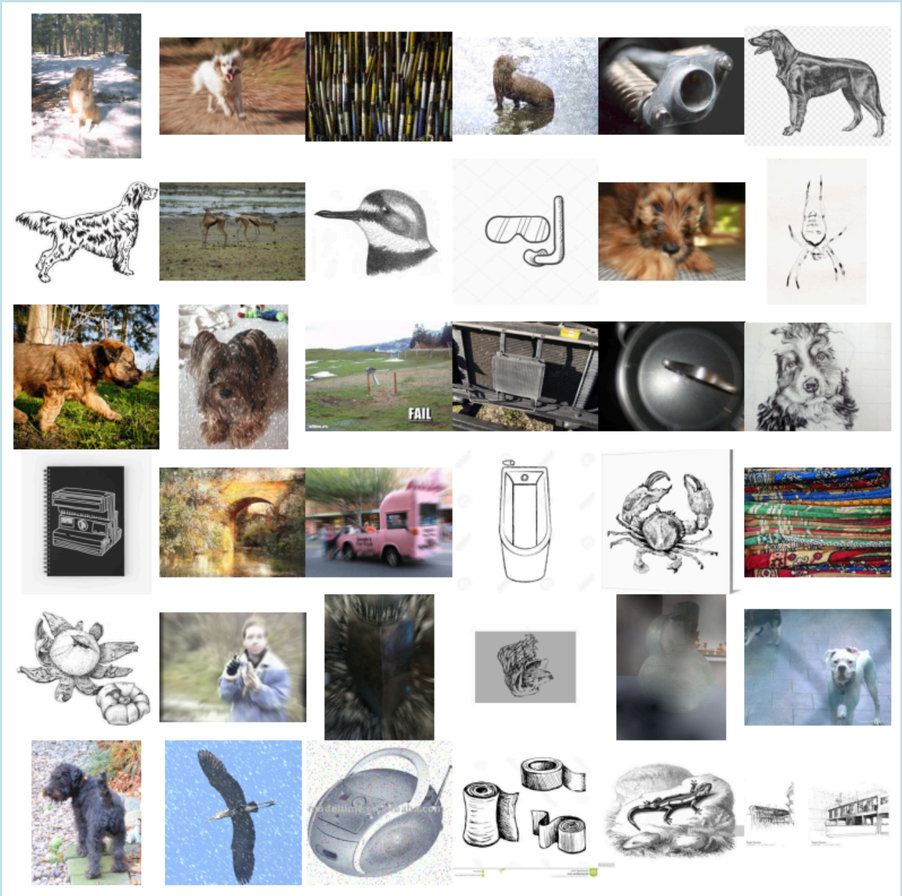
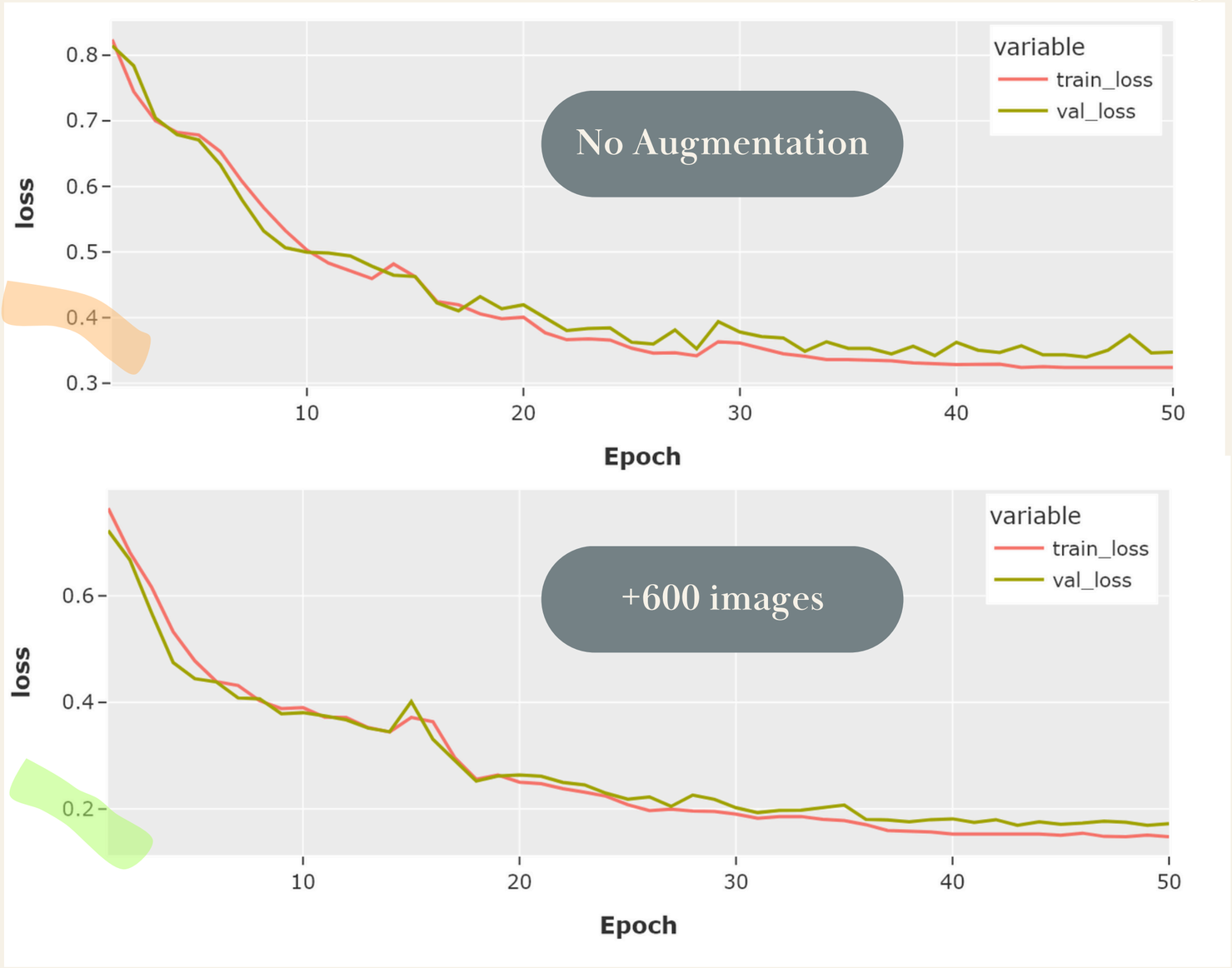


IMAGE-NET [2]

Impact of the Data Augmentation on the *Goodness of fit* :

- -0.2 loss
- 25 mn -> 55 mn

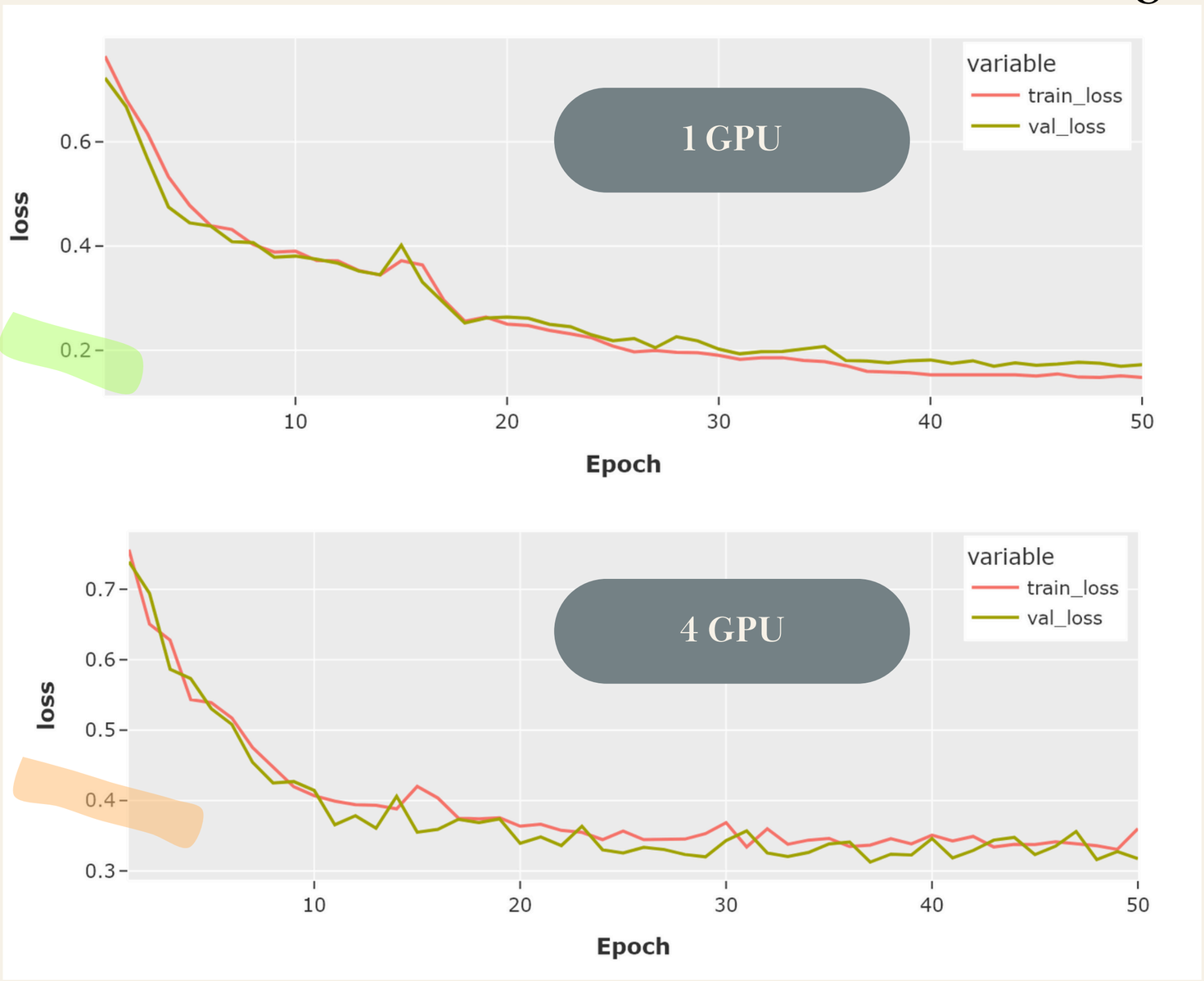
Ressources : **1 GPU**

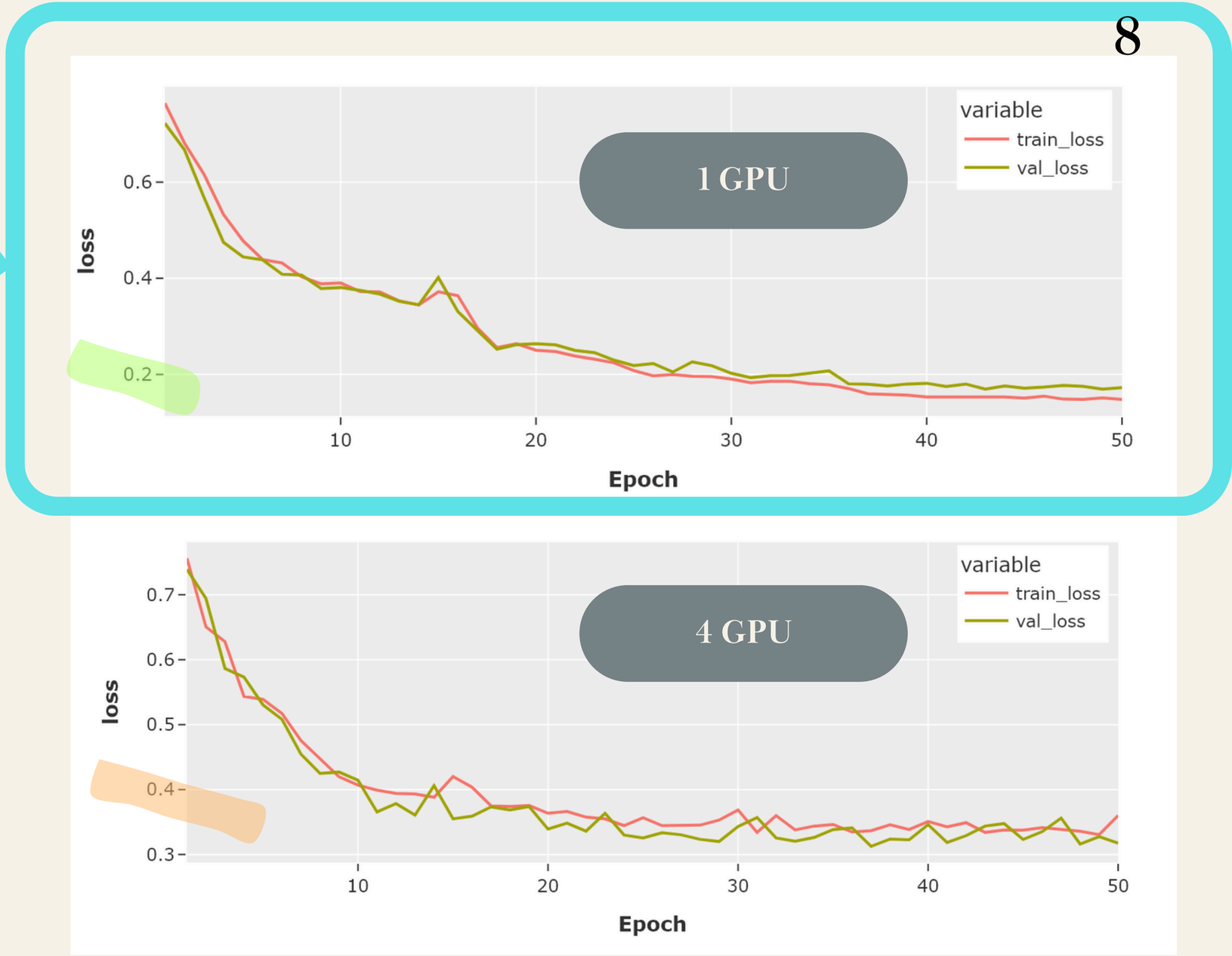


Impact of the Multi-GPU Training on the *Goodness of fit* :

- 55 mn -> 15 mn
- Instability
- +0.2 loss

Augmentation : +600 imgs



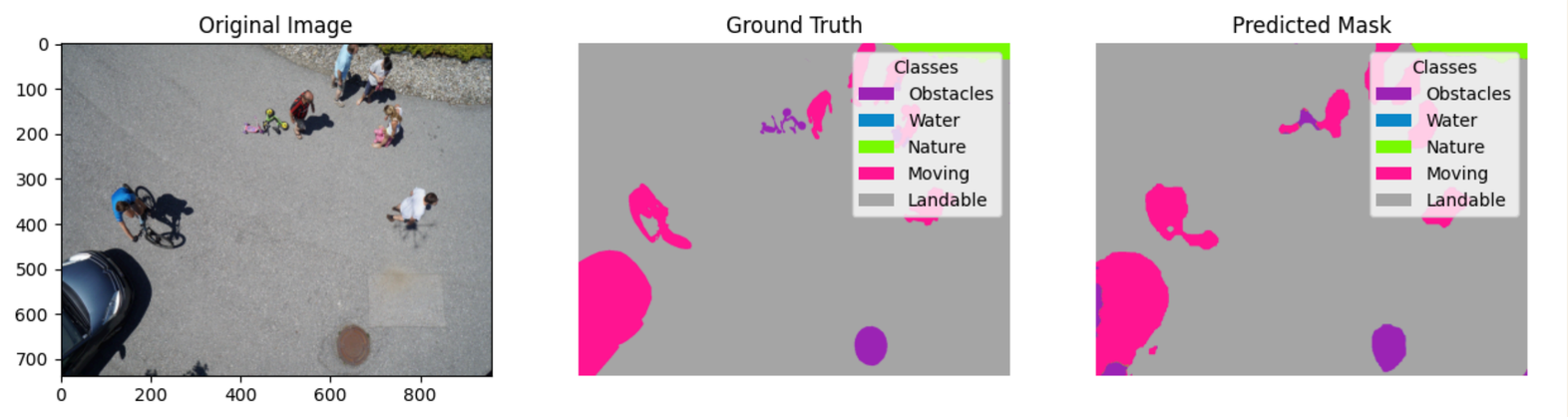


Keep this one,
the best,
for evaluation

Augmentation : +600 imgs

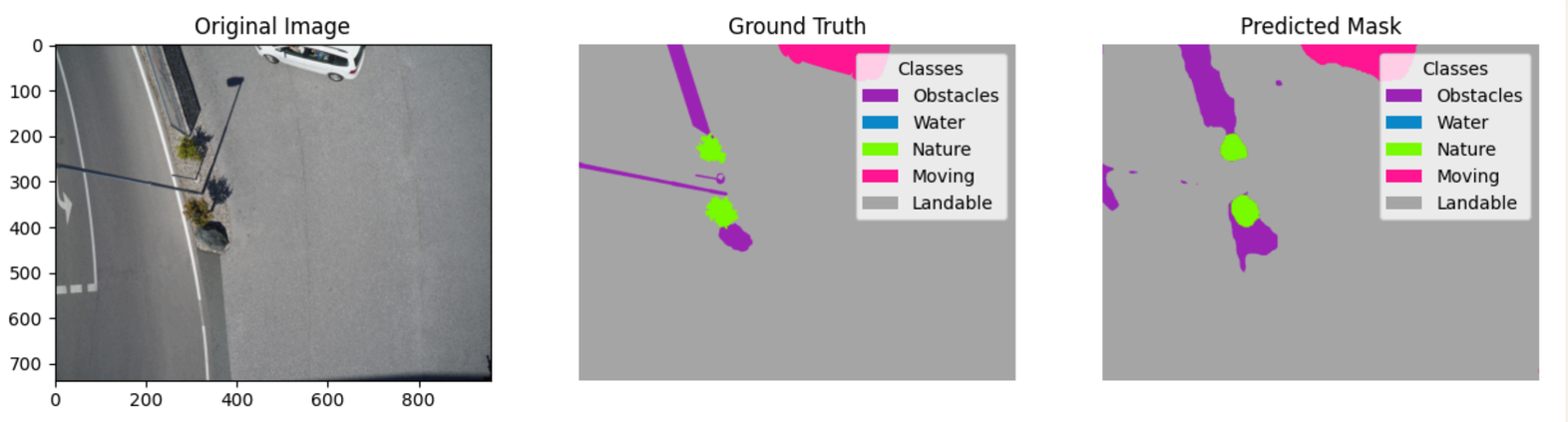
TOP 5 BEST OUTPUTS !

5th



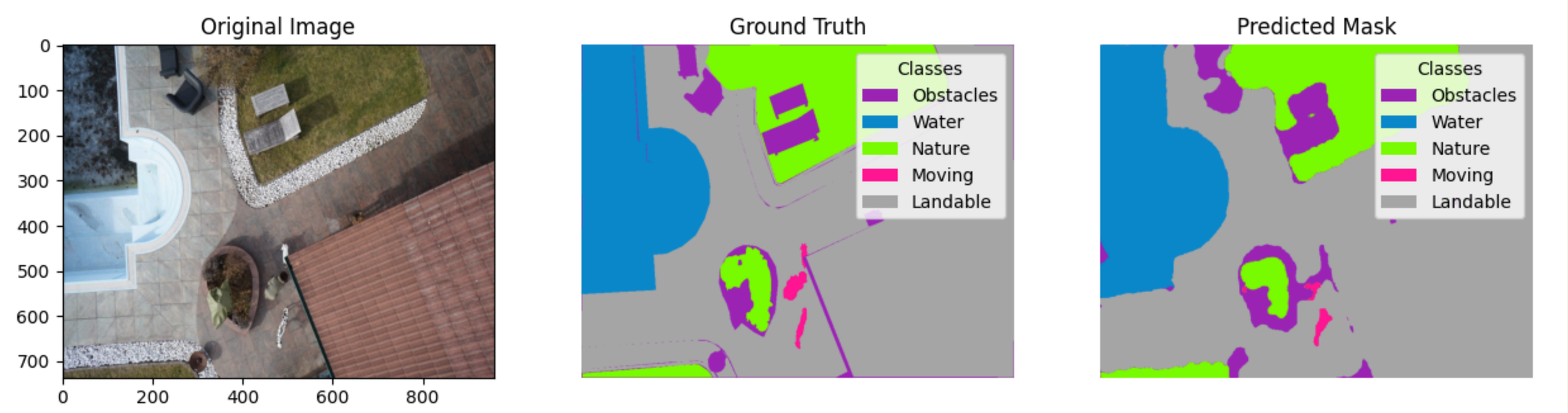
mIoU = 0.62, mPA = 0.89

4th



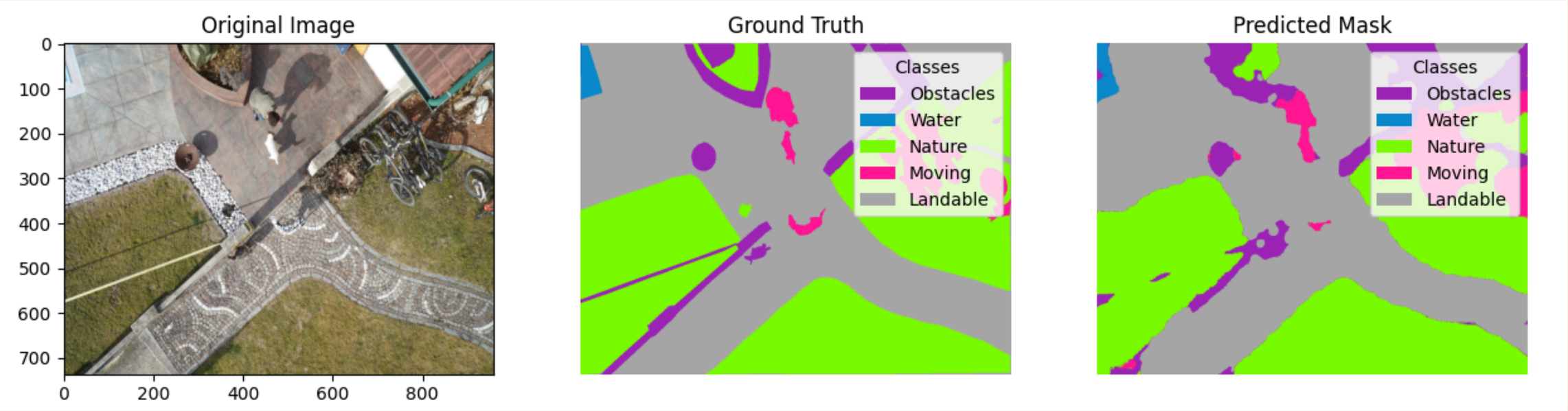
mIoU = 0.62, mPA = 0.90

3rd



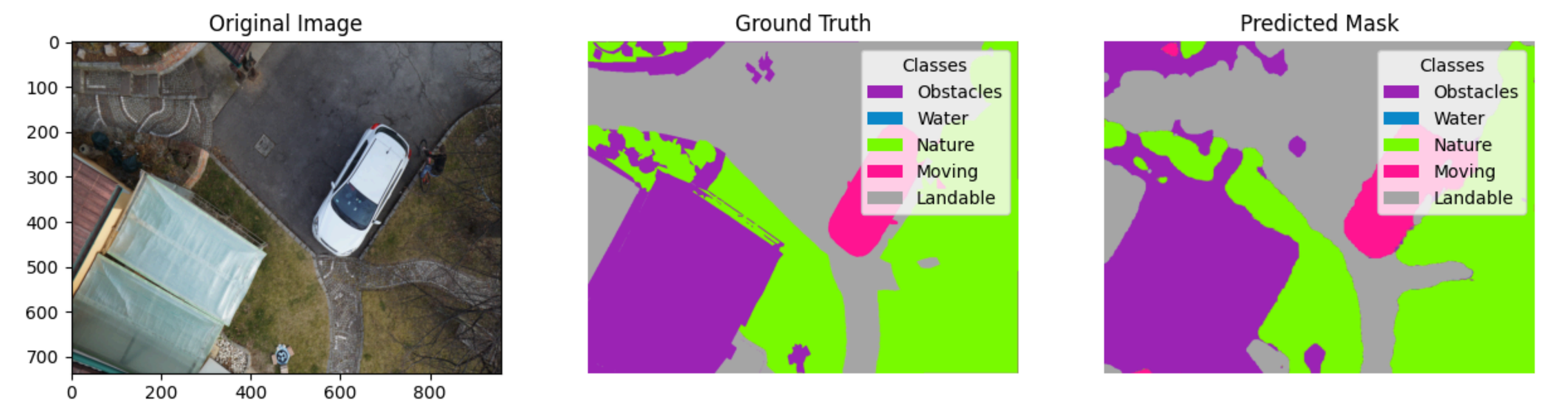
mIoU = 0.74, mPA = 0.81

2nd



mIoU = 0.69, mPA = 0.87

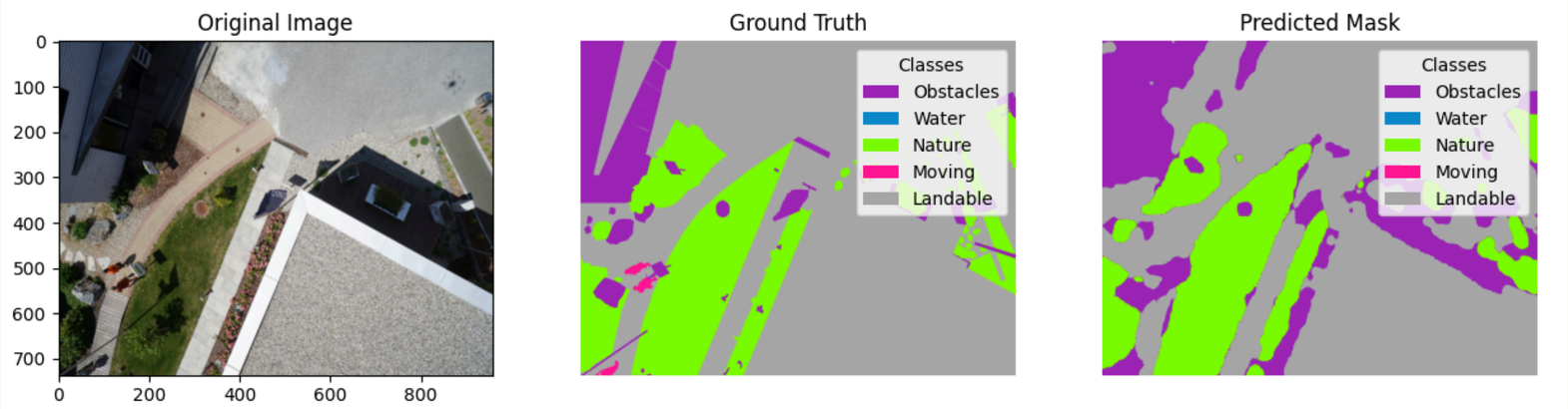
1st



mIoU = 0.65, mPA = 0.93

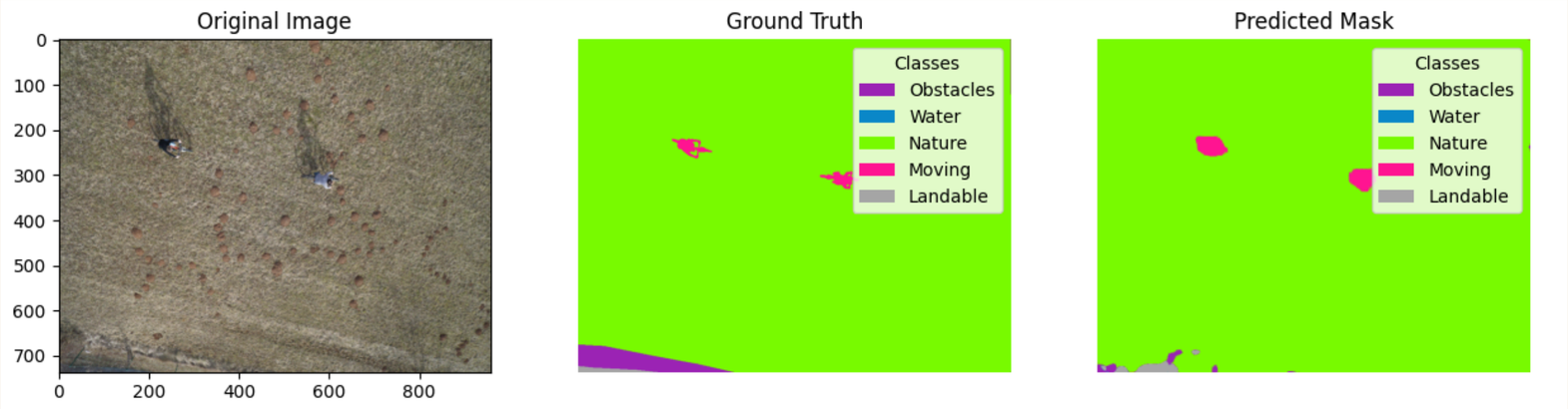
TOP 5 WORST OUTPUTS !

5th



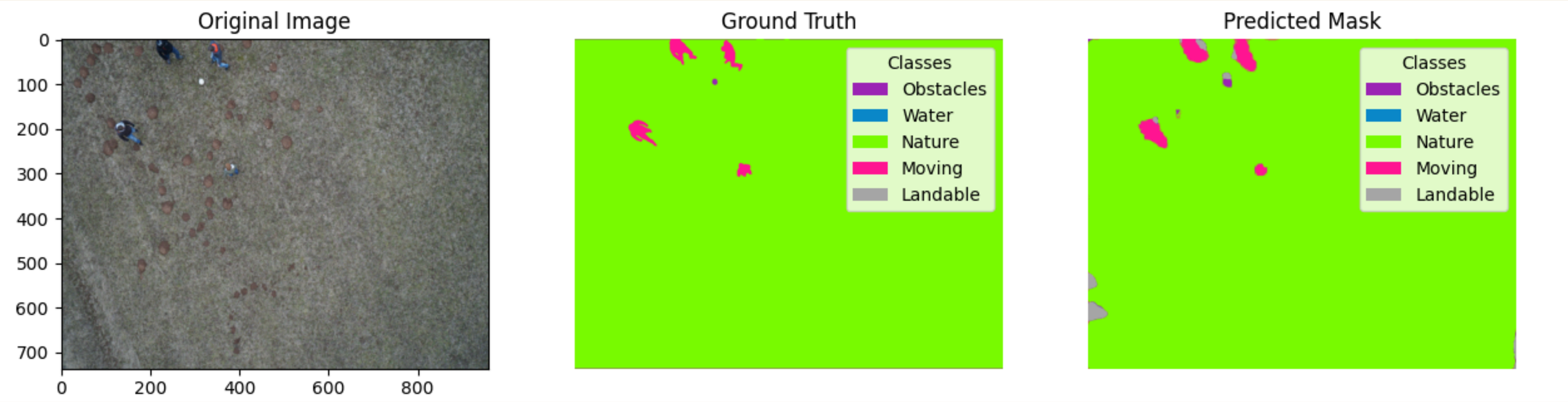
mIoU = 0.39, mPA = 0.62

4th



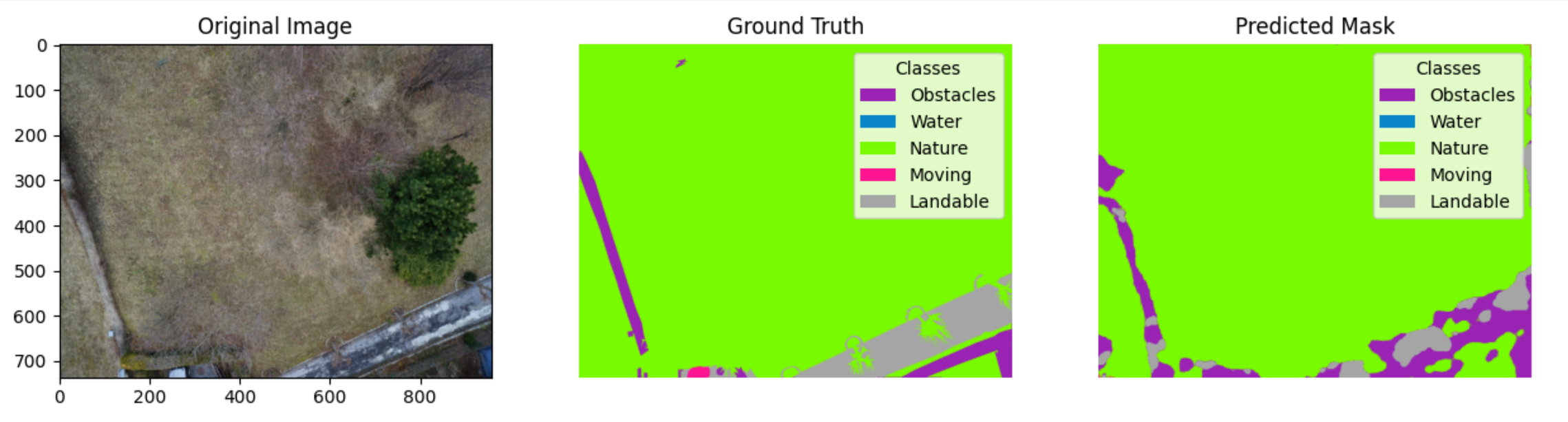
mIoU = 0.37, mPA = 0.62

3rd



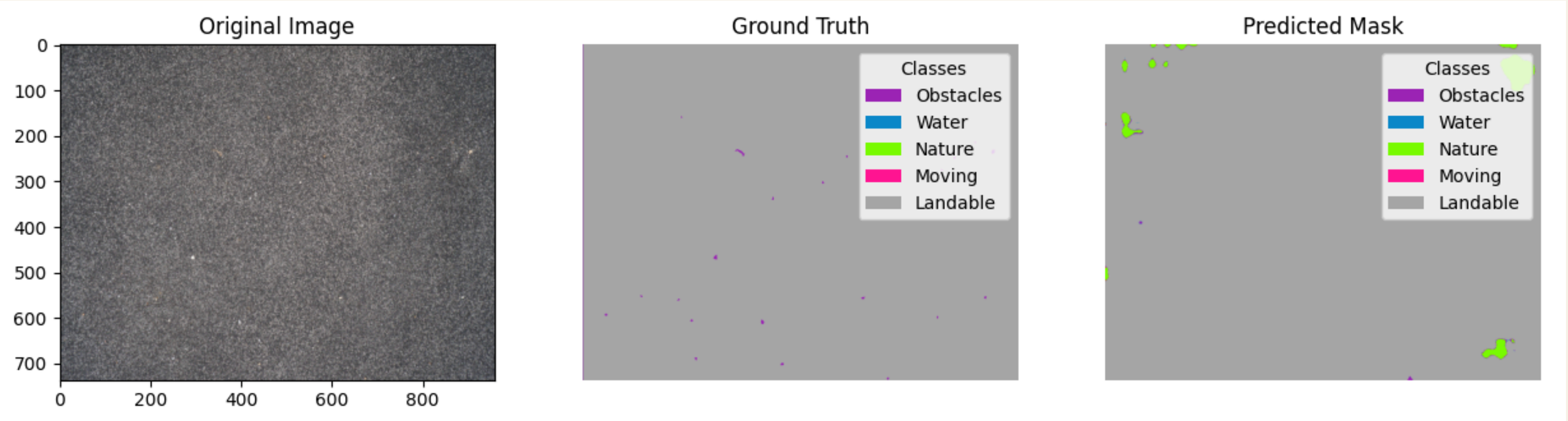
mIoU = 0.32, mPA = 0.64

2nd



mIoU = 0.29, mPA = 0.48

1st



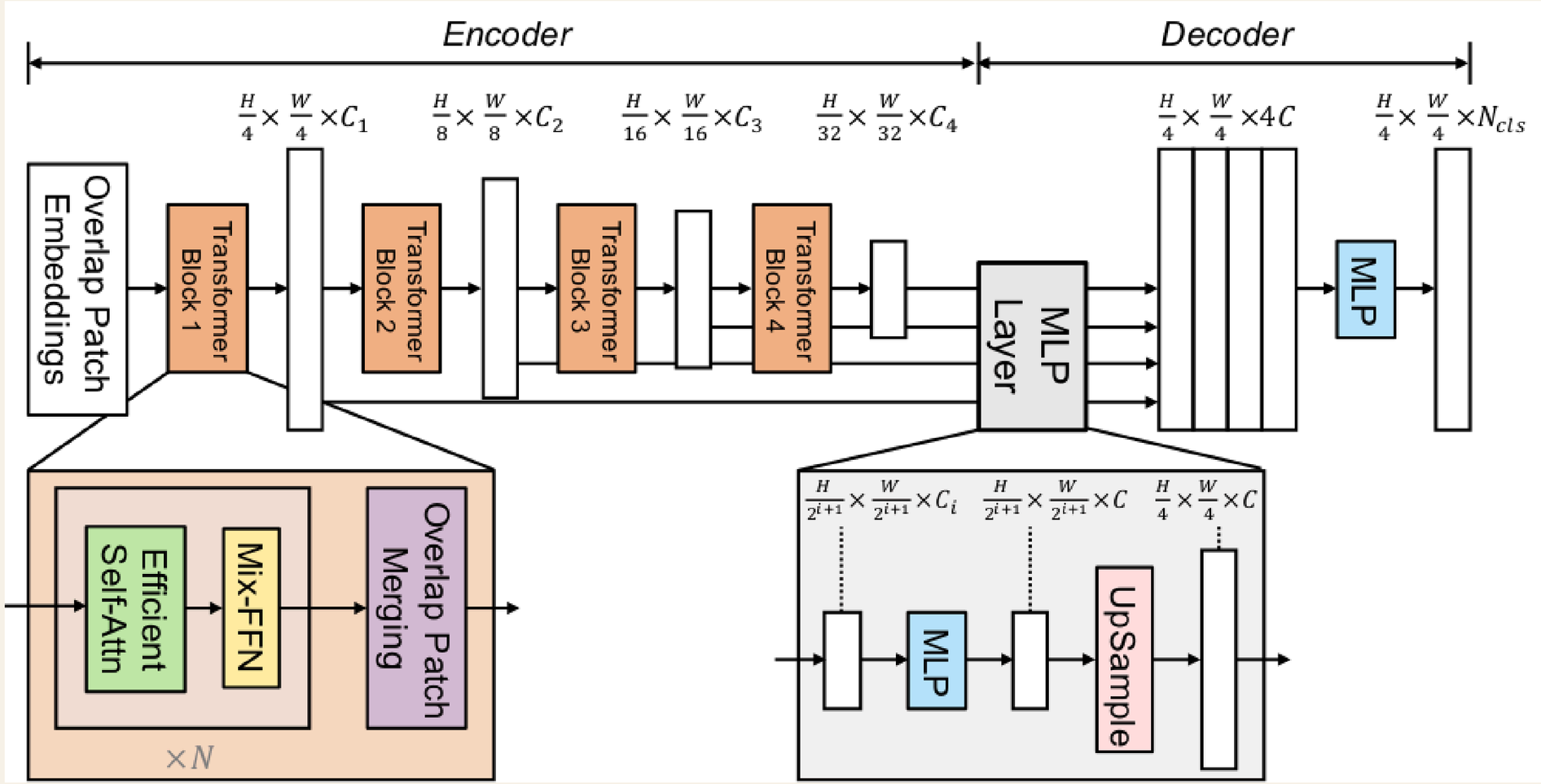
mIoU = 0.19, mPA = 0.50

III - SEGFORMER

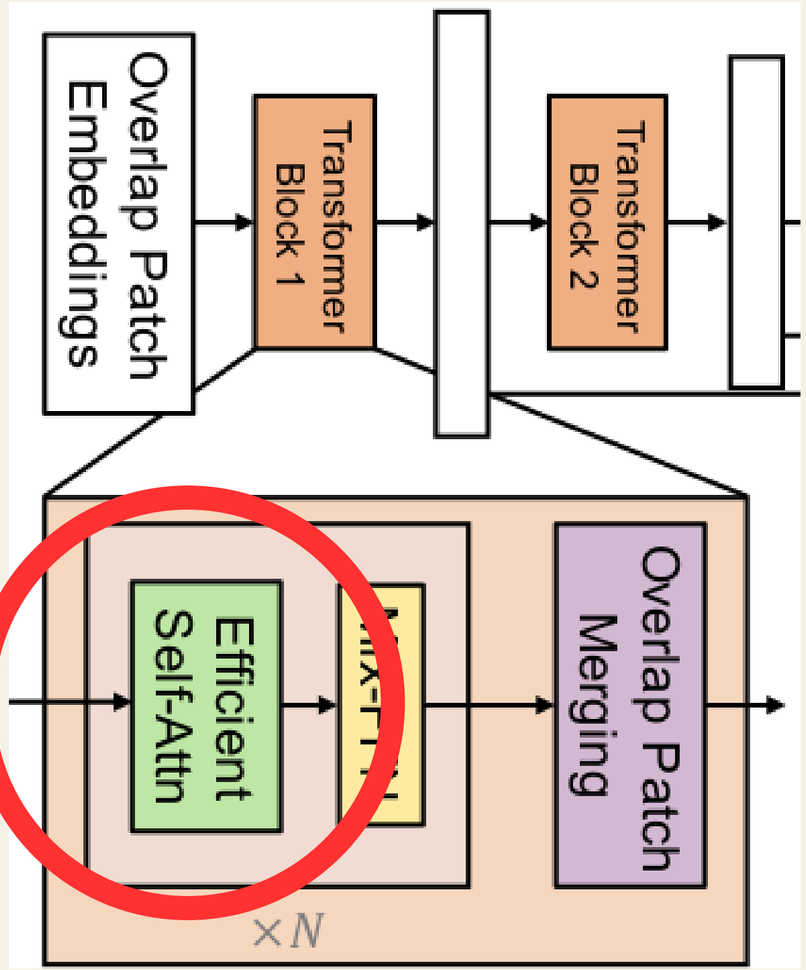
Overview, Training, Outputs

SegFormer [3] from HuggingFace

SegFormer[3] from HuggingFace :



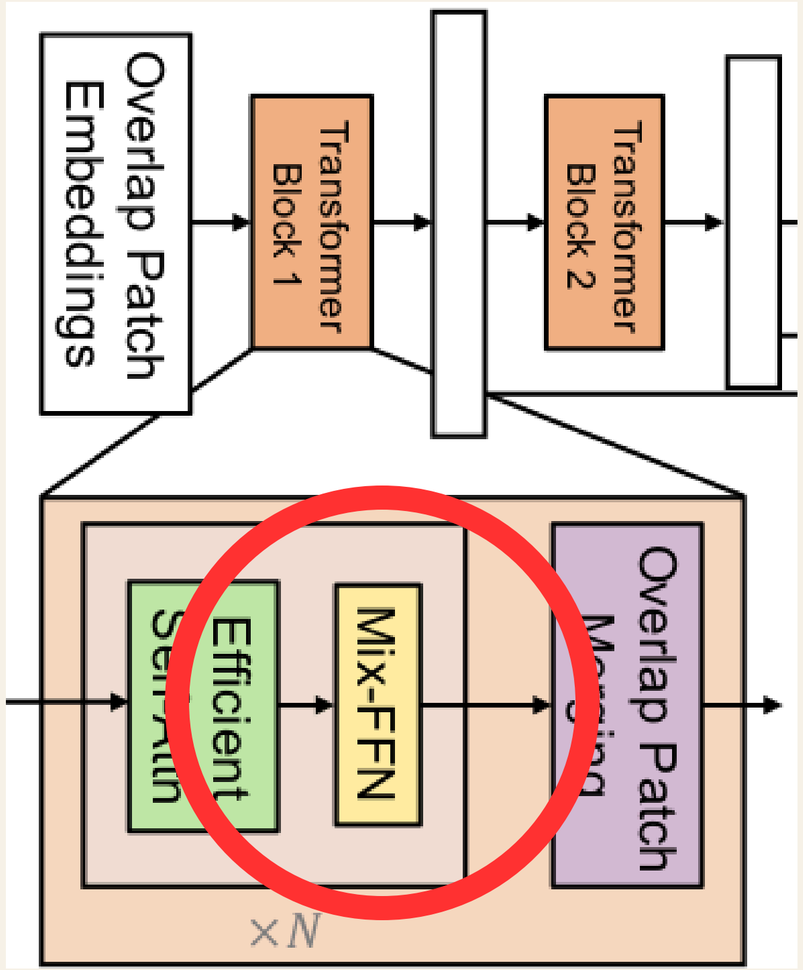
A classical Encoder-Decoder Architecture



ENCODER : ‘mit-b0'

Efficient Self-Attention :

Attention Matrix optimized computation using batches



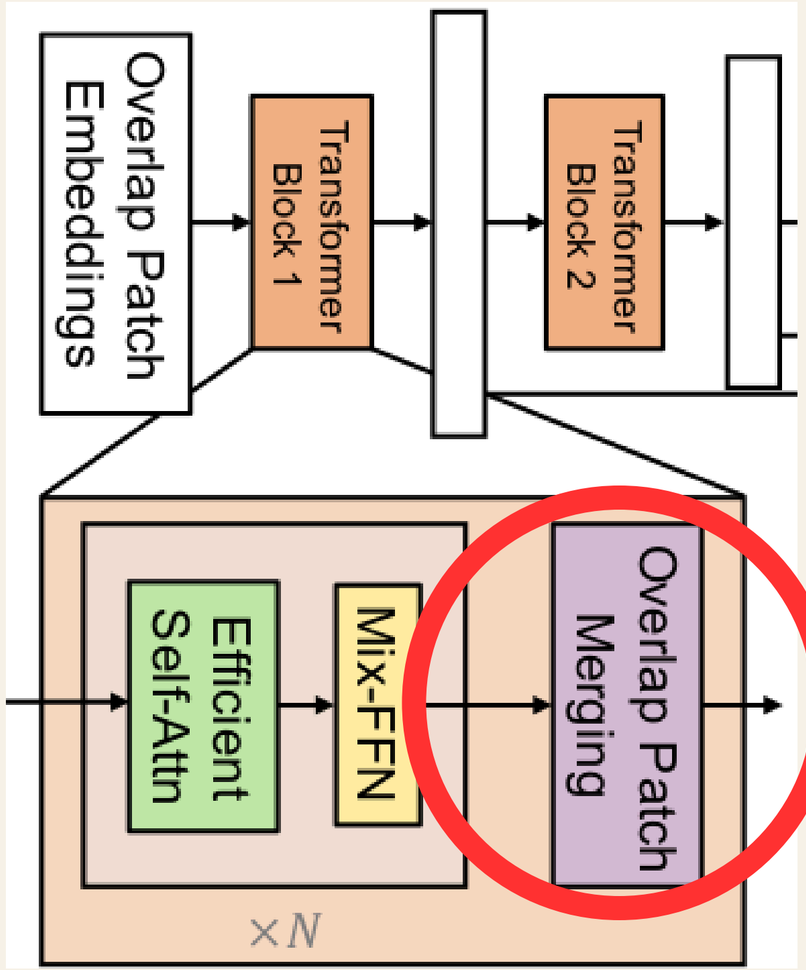
ENCODER : ‘mit-b0'

Efficient Self-Attention :

Attention Matrix optimized computation using batches

Mix-FFN :

Convolution & Feed-Forward Network layer for both local and global informations



ENCODER : ‘mit-b0'

Efficient Self-Attention :

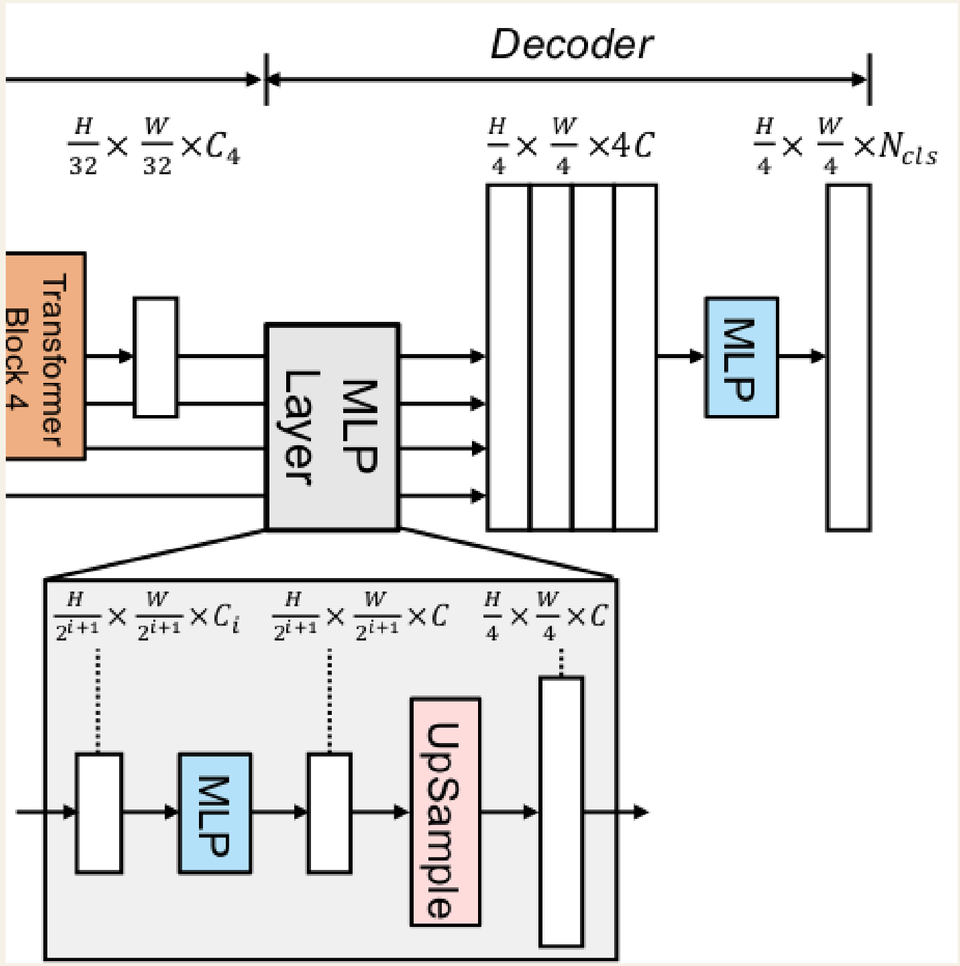
Attention Matrix optimized computation using batches

Mix-FFN :

Convolution & Feed-Forward Network layer for both local and global informations

Overlap Patch Merging :

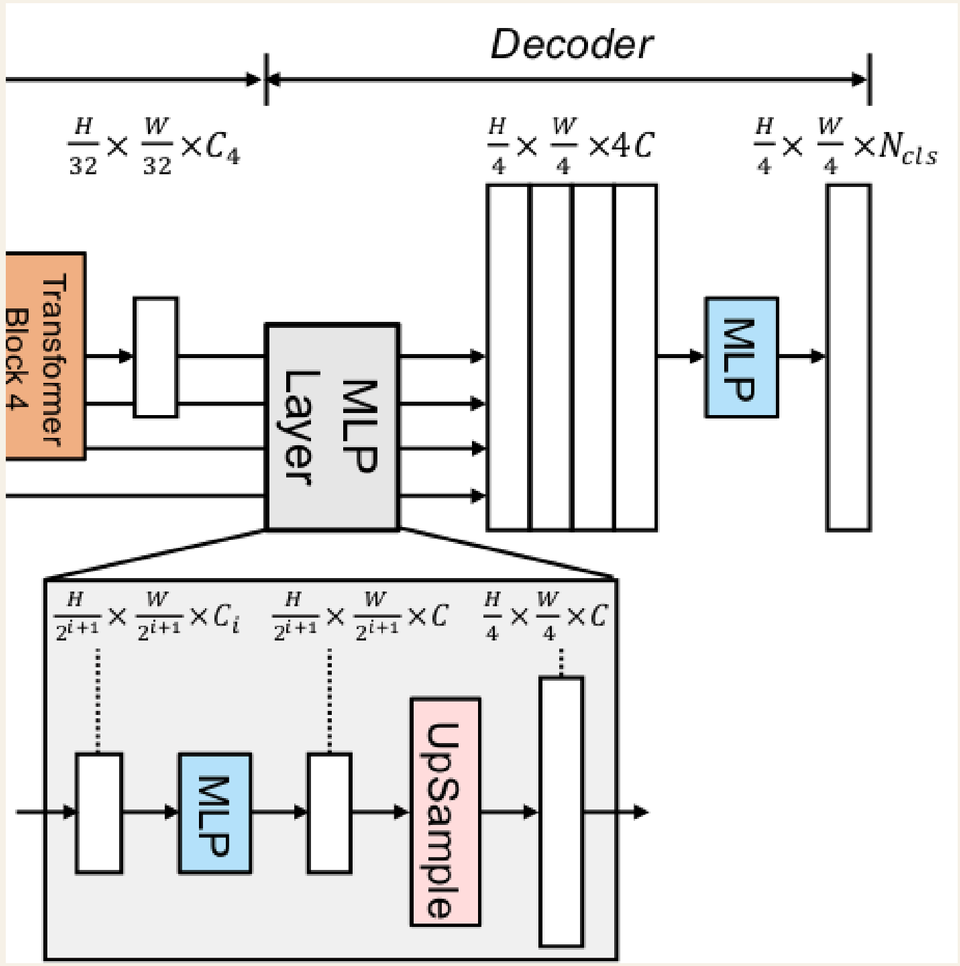
Convolution layer with stride to downsample the image.



HAND-MADE DECODER :

Same Architecture :

Very simple using MLP and Upsample to capture local and global information.



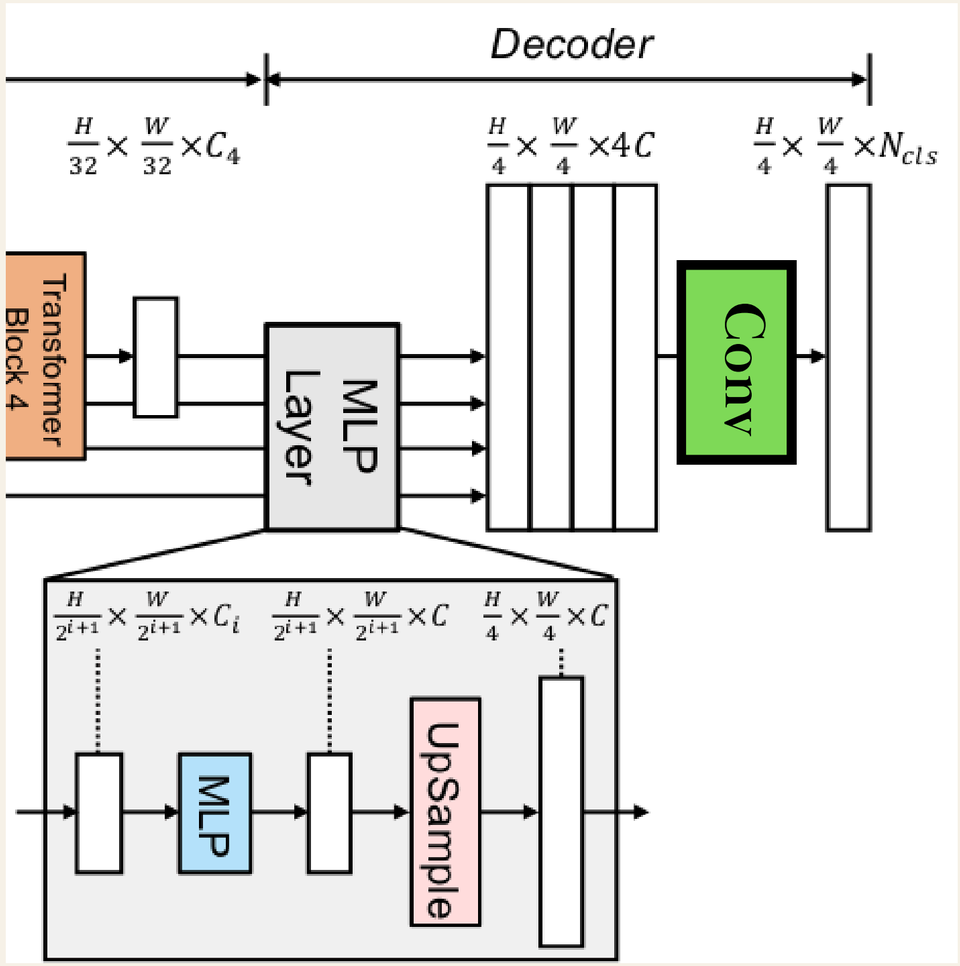
HAND-MADE DECODER :

Same Architecture :

Very simple using MLP and Upsample to capture local and global information.

Different Activation Function :

Use ReLu Activation function instead of **GeLu, less flexibility to avoid over-fitting**



HAND-MADE DECODER :

Same Architecture :

Very simple using MLP and learnable Upsampling to capture local and global information.

Different Activation Function :

Use ReLu Activation function instead of **GeLu, less flexibility to avoid over-fitting**

Final Convolution :

Reconstruct local information instead of global using MLP layer
Reduce the number of parameters

Components :

- The same as for the U-Net

Encoder weights :

- Imagenet DataBase

Augmentation :

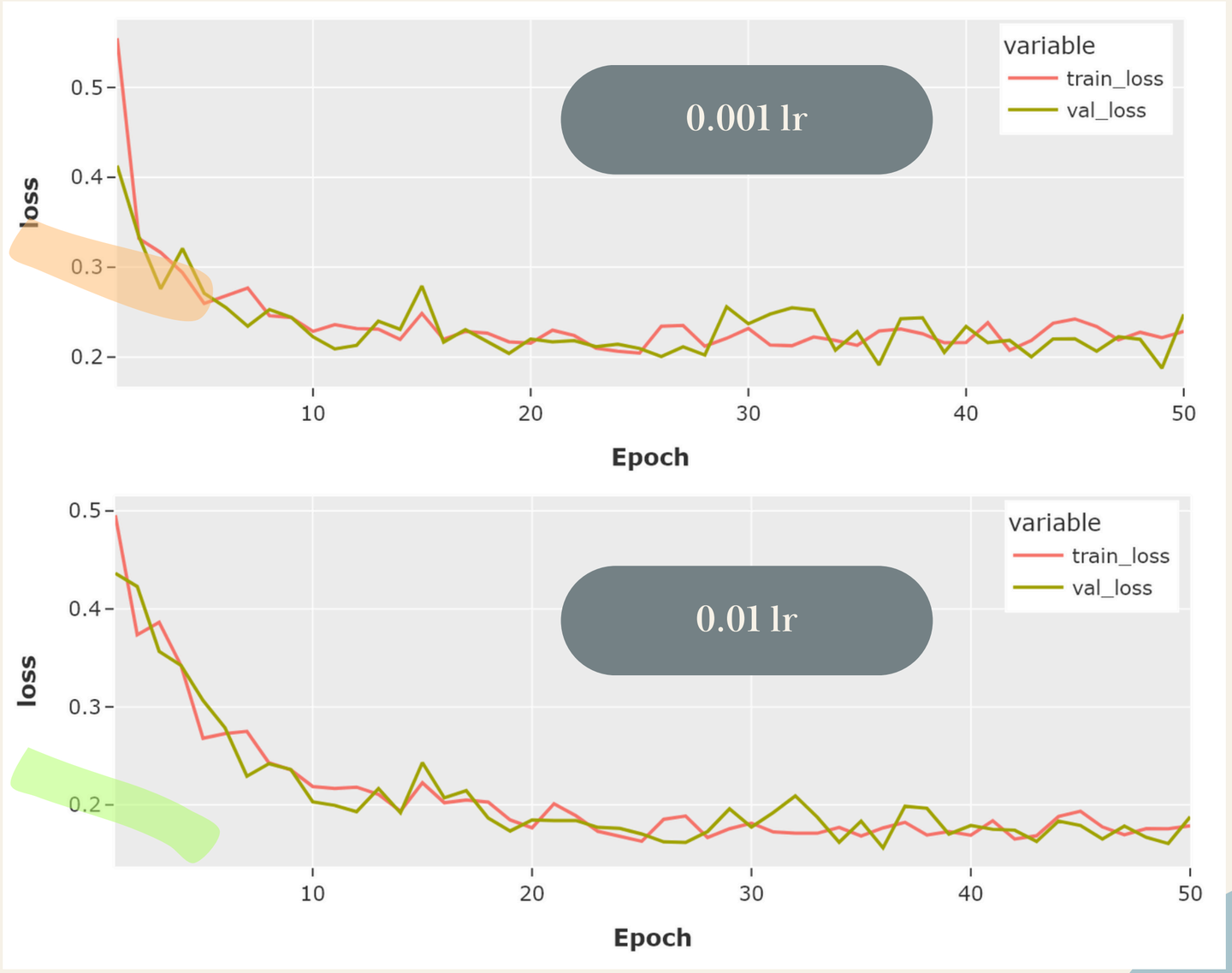
- +600 samples

Ressources :

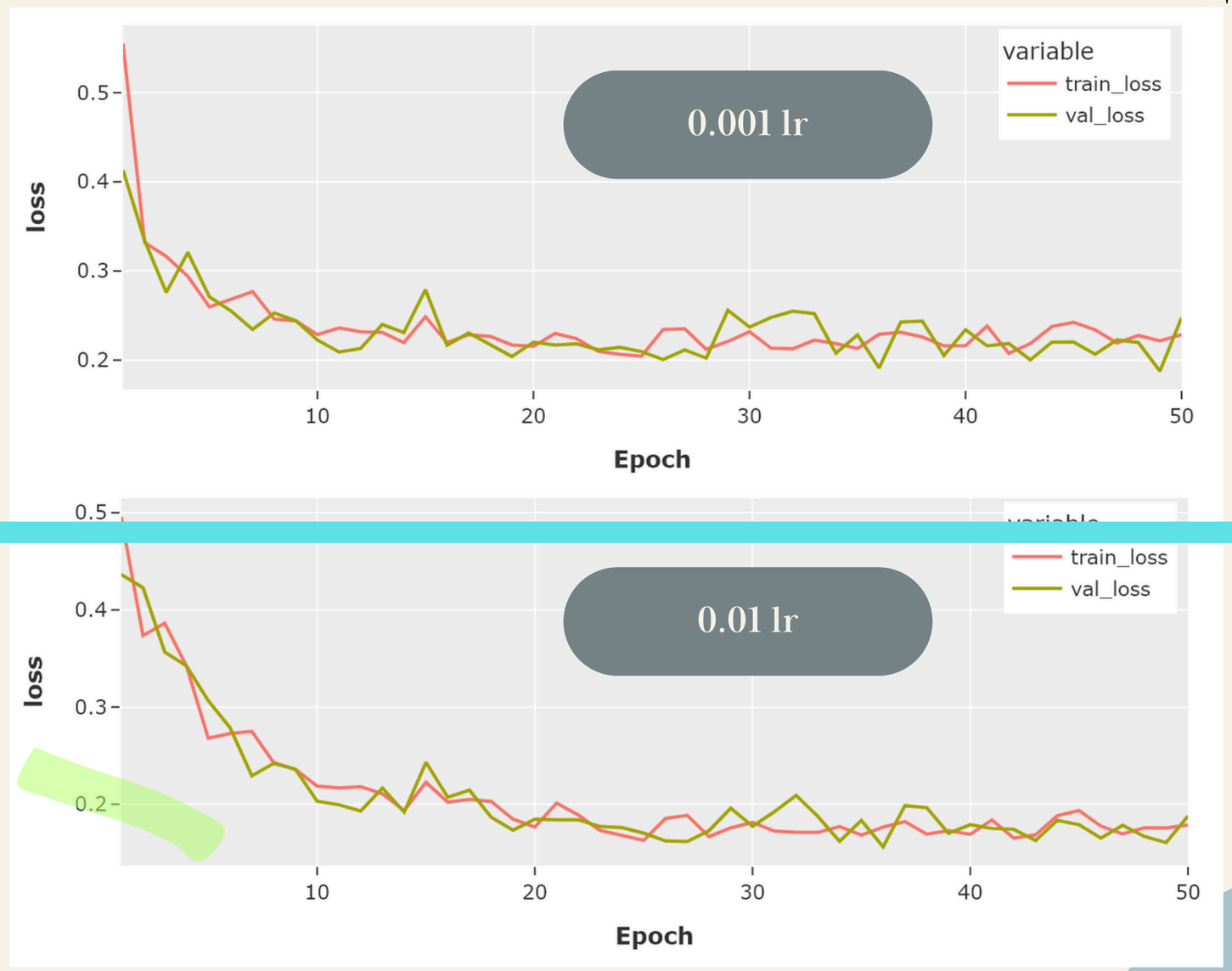
- Only 4 GPUs possible !

Deal with loss issues :

- Smaller batch on DDP, increase the learning rate !



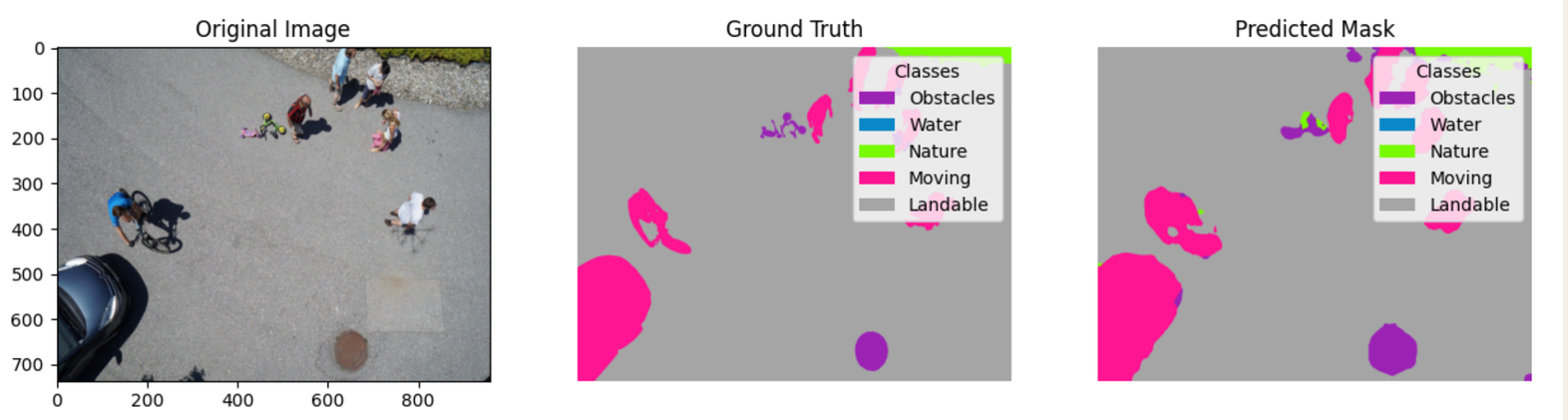
Keep this one,
for evaluation



TOP 2 BEST OUTPUTS !

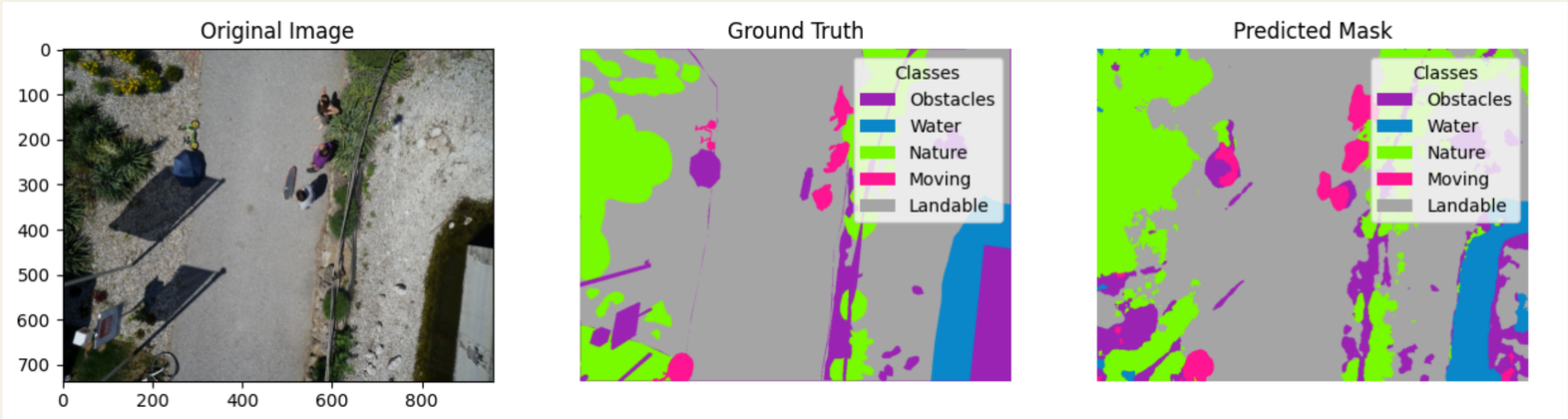
Outputs

1st



mIoU = 0.67, mPA = 0.92

2nd

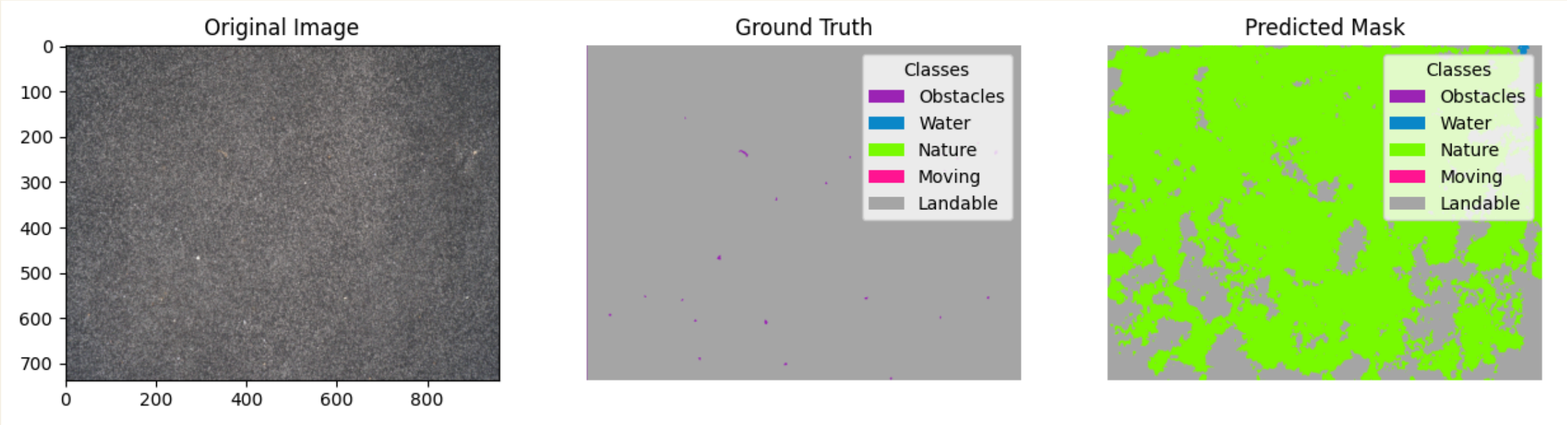


mIoU = 0.62, mPA = 0.92

TOP 2 WORST OUTPUTS !

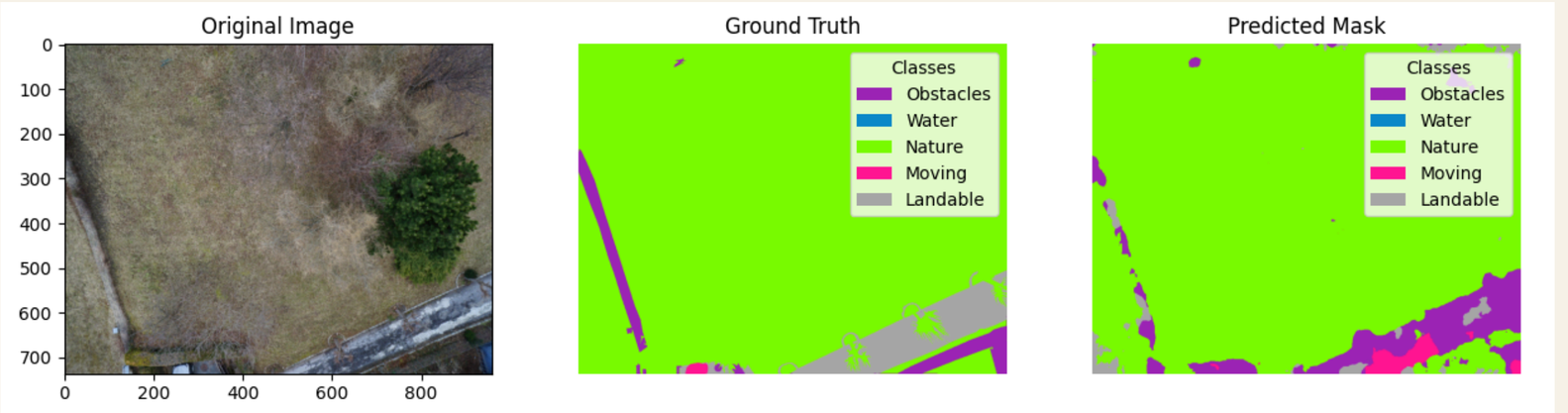
Outputs

1st



mIoU = 0.06, mPA = 0.12

2nd



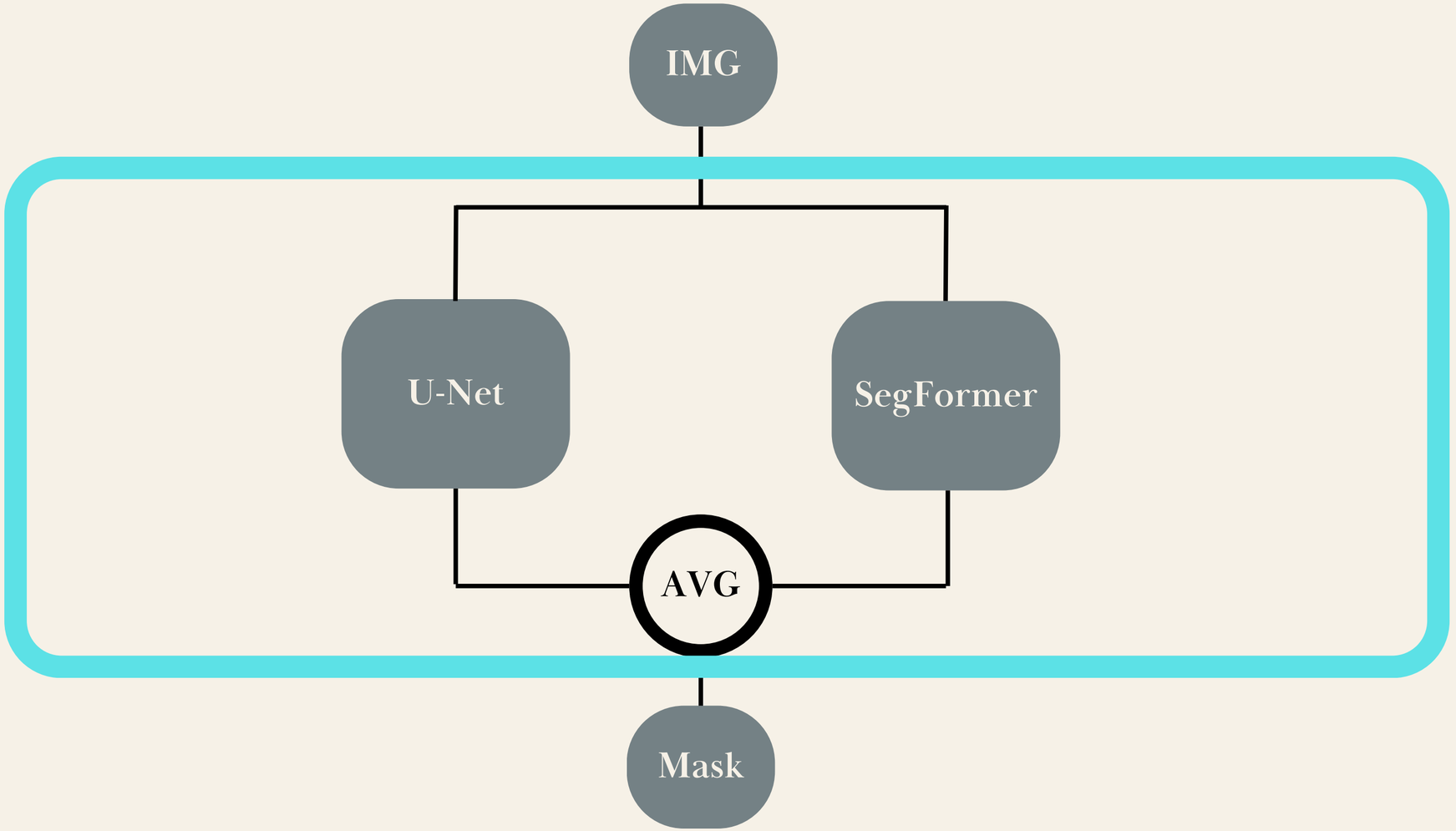
mIoU = 0.28, mPA = 0.38

IV- U-FORMER

Overview, Outputs

UFormer : Retreive the information from both trained models

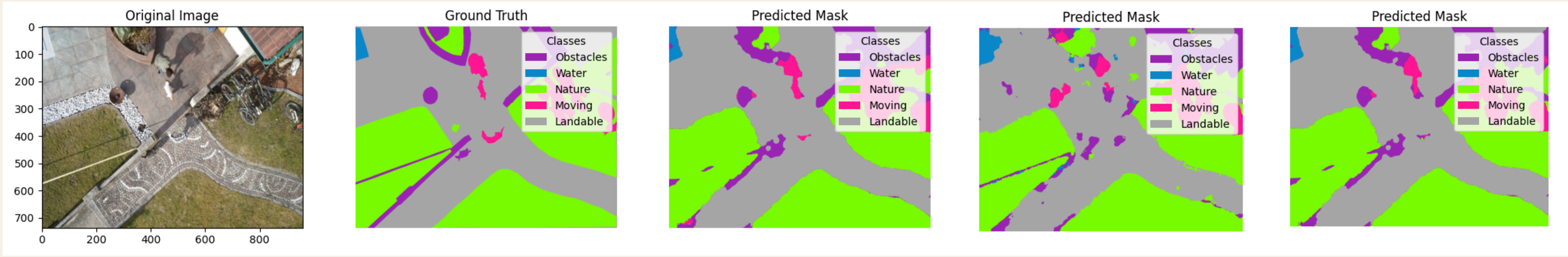
UFormer : Retreive the information from both trained models



BEST OUTPUT !

Outputs

U-NET SegFo UFo

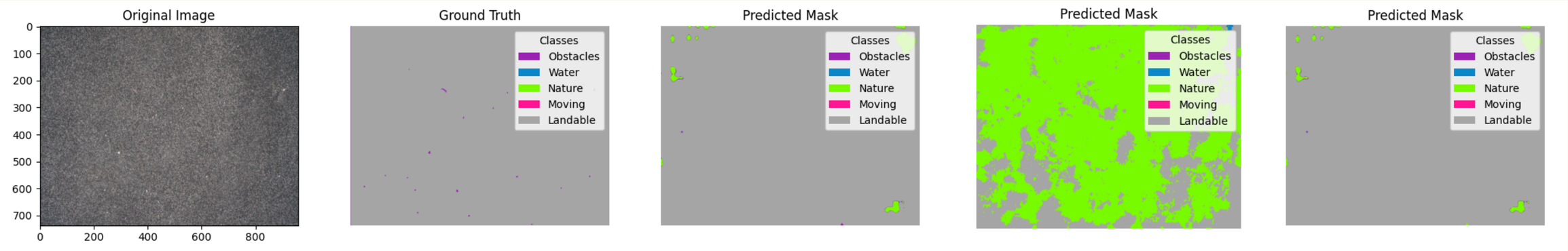


mIoU = 0.73, mPA = 0.88

WORST OUTPUT !

Outputs

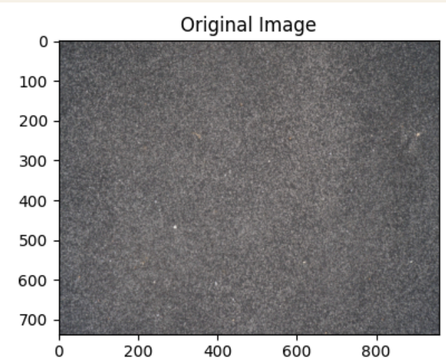
U-NET SegFo UFo

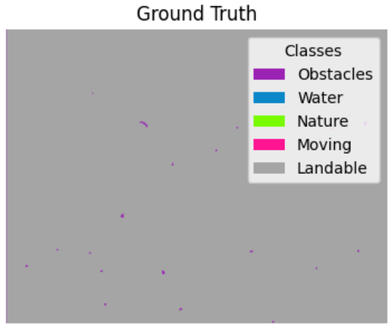


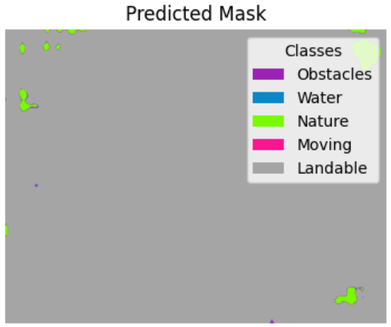
mIoU = 0.24, mPA = 0.49

Outputs

U-NET

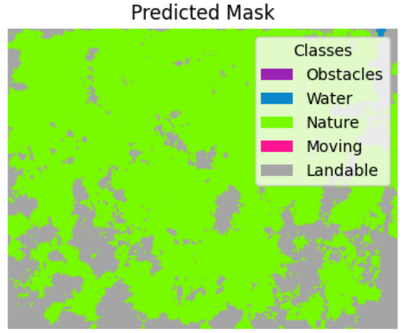




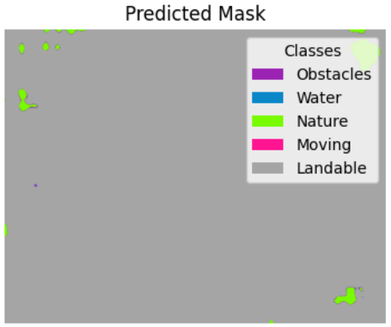


IoU = 0.24, mPA = 0.49

SegFo



UFo



Better than the U-Net !

V- TEST RESULTS

Overview, Outputs

Performance of each model on the Test Set :

Model	Test Loss	Test mIoU	Test mPA	GPU(s)	Time
SegFormer	0.216	0.519	0.816	4	15 min
U-Net	0.172	0.533	0.812	1	55 min
UFormer	0.156	0.536	0.811	X	X

Table 1: Results for multiclass segmentation tasks.

Performance of each model on the Test Set :

Model	Test Loss	Test mIoU	Test mPA	GPU(s)	Time
SegFormer	0.216	0.519	0.816	4	15 min
U-Net	0.172	0.533	0.812	1	55 min
UFormer	0.156	0.536	0.811	X	X

Table 1: Results for multiclass segmentation tasks.

- Best **Mean Pixel Accuracy (MPA)** for the **SegFormer**
- Gathering information with **UFormer** leads to the best **IoU**

VI - CONCLUSION

Limit, discussion

Optimized hyper-paremeters :

- UFormer's weighted average coefficients
- Tversky Loss coefficients alpha and beta

Try a Modeling distribution across GPUs instead of a Data parallelisation.

- May be really efficient for the SegFormer to deal with loss issues.

Upscale the data resolution.

- Try the efficiency of our distributed computation with 6K images

Skills acquired :

- Organize and modularize a Python’s Data Science Project.
- Coding with PyTorch
- Use of GriCad and OAR system
- Implementation of data distributed computations

THANK YOU ALL !

Sources

[1] : CHEN, Y. (2023). APPLICATION OF RESNET18-UNET IN SEPARATING TUMORS FROM BRAIN MRI IMAGES. RETRIEVED FROM RESEARCHGATE UNDER CREATIVE COMMONS ATTRIBUTION 3.0 UNPORTED LICENSE.

[2] : PAPERS WITH CODE. (2023). IMAGENET-HARD: THE HARDEST IMAGES REMAINING FROM A STUDY OF THE POWER OF ZOOM AND SPATIAL BIASES IN IMAGE CLASSIFICATION. RETRIEVED FROM PAPERS WITH CODE UNDER THE MIT LICENSE.

[3] : XIE, E., WANG, W., YU, Z., ANANDKUMAR, A., ALVAREZ, J. M., & LUO, P. (2021). SEGFORMER: SIMPLE AND EFFICIENT DESIGN FOR SEMANTIC SEGMENTATION WITH TRANSFORMERS. ARXIV. HTTPS://DOI.ORG/10.48550/ARXIV.2105.15203