



# CHALLENGE DATA ENS

## LAND COVER PREDICTIVE MODELING FROM SATELLITE IMAGES

Raphaël Basler, Charlotte Arnould et Alexis Ego

# PRESENTATION CHALLENGE

## *Preligens*



Fournit des **informations** sur des missions critiques  
afin de permettre la prise de décision

Technologie basée sur l'analyse **automatisée** des **flux**  
du renseignement géospatial

# PRESENTATION CHALLENGE

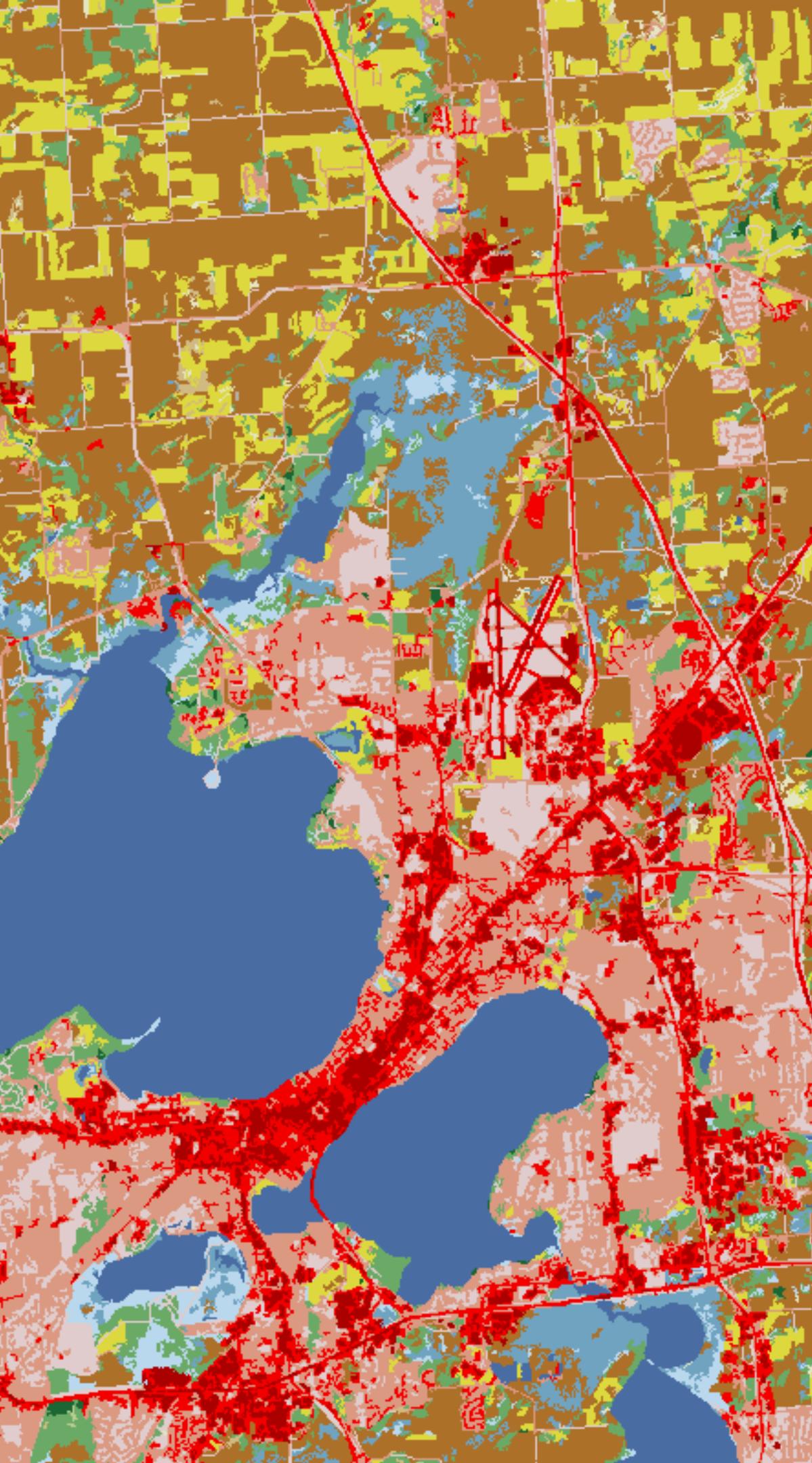
## *Cartographie de la couverture terrestre*

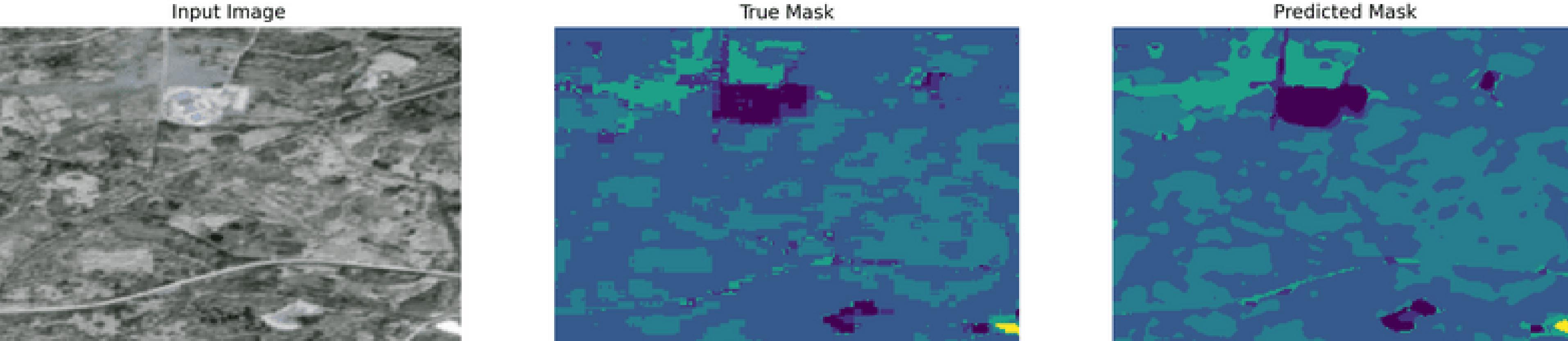
\*

Consiste à cartographier les matériaux physiques à la surface de la Terre (eau, neige, forêt...)

\*

Dans le cadre de Preligens :  
Cela permet d'apporter de l'information autour d'un site d'intérêt





## PRESENTATION CHALLENGE

### *Objectif du challenge*

\* Prédire la proportion de chaque classe sur une image satellite

\* Apprentissage d'un masque et extraction des proportions des classes

#### Classes à identifier :

- 0 Pas de données
- 1 Nuages
- 2 Zones artificielles et construction
- 3 Zones cultivées
- 4 Forêts de feuillus
- 5 Forêts de conifères
- 6 Végétation herbacée
- 7 Matériaux naturels (ex : roches)
- 8 Neiges permanentes
- 9 Corps d'eau



# PRESENTATION CHALLENGE

## *Évaluation du challenge*

La métrique utilisée pour l'évaluation est la divergence de Kullback-Leibler avec un terme supplémentaire pour lisser la divergence autour de 0.

$$\mathbf{KL}(y, \hat{y}) = \sum_{i=1}^C (y_i + \epsilon) \log \left( \frac{y_i + \epsilon}{\hat{y}_i + \epsilon} \right)$$

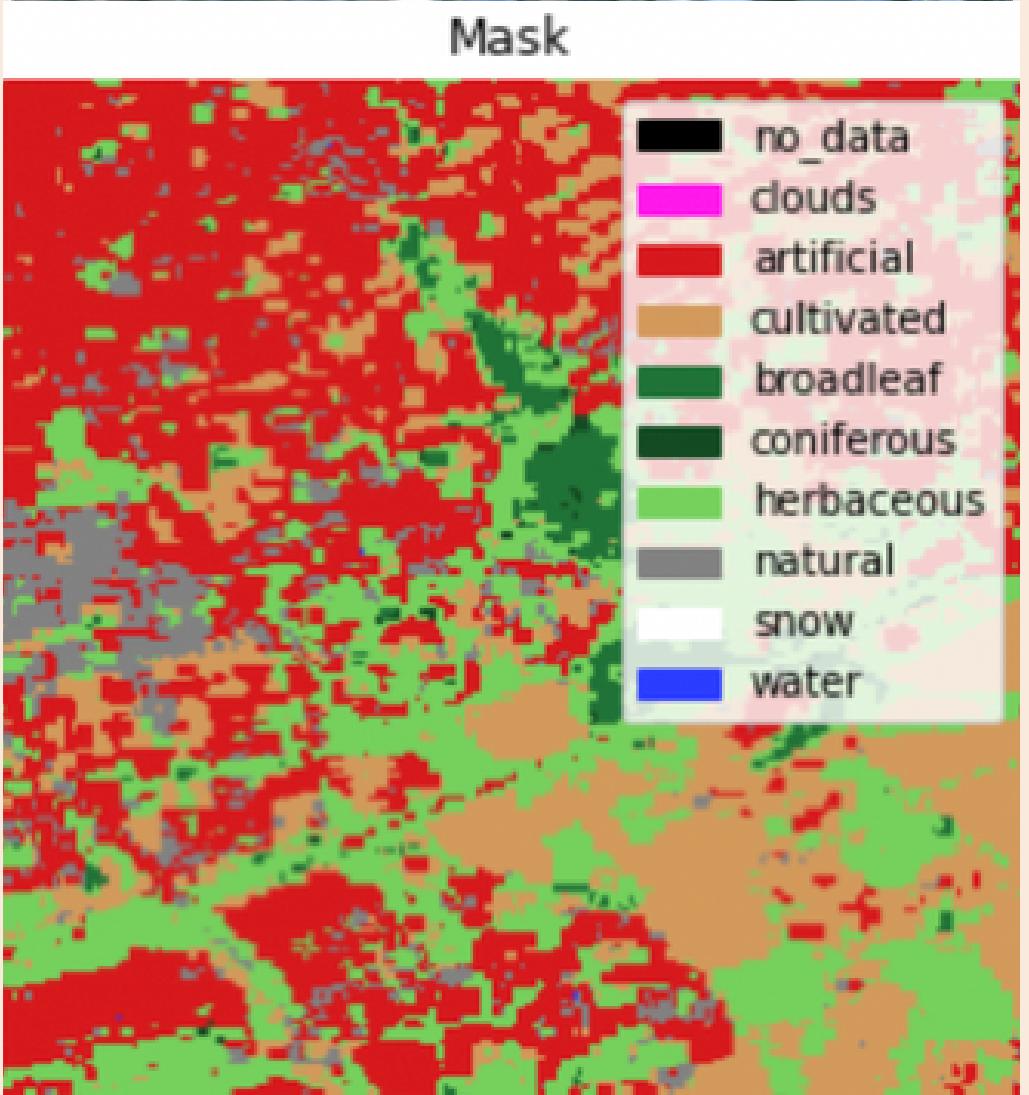
C'est une mesure de la différence entre deux distributions  
l'objectif est donc de la minimiser.

# PRESENTATION CHALLENGE

## *Origine du dataset*

- \* Les images proviennent d'un set de données issus du projet européen **S2GLC**
- \* Le satellite qui a pris les images est **Sentinel-2**
- \* La résolution est telle qu'un pixel couvre 10 m<sup>2</sup>
- \* Objectif du projet : réaliser une **cartographie** complète de l'Europe





# PRESENTATION CHALLENGE

## *Contenu du dataset*

- \* 18491 images pour le training set  
5043 dont seulement la moitié est disponible pour le test set
- \* Un masque de segmentation et la proportion de chaque classe pour toutes les images du training set.
- \* Une image est constituée de 256\*256 pixels, chaque pixel étant codé sur 4 channels (CMYK).  
Un masque est une image de 256\*256 pixels avec pour chaque pixel un entier correspondant au numéro de la classe.

# PRESENTATION CHALLENGE

## *Répartition des classes*



La répartition des classes est très **déséquilibrée** dans le training set.

Ce déséquilibre est également présent dans le test set.



Comment prendre en compte ce déséquilibre lors de l'entraînement du modèle ?



# CONNAISSANCE METIER

## *Gestions des classes*

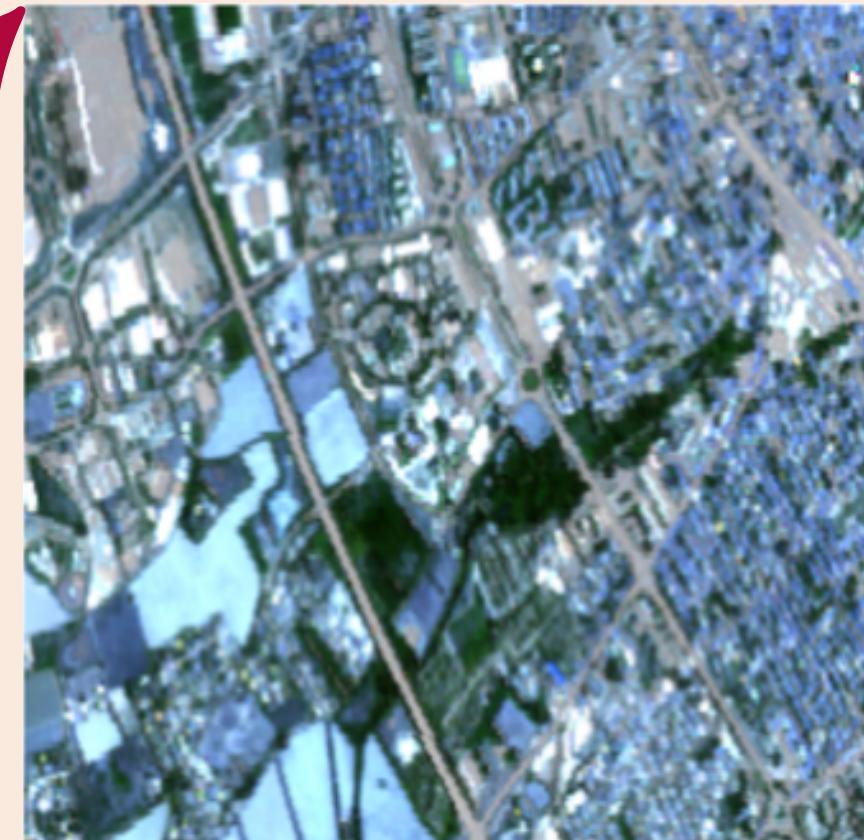
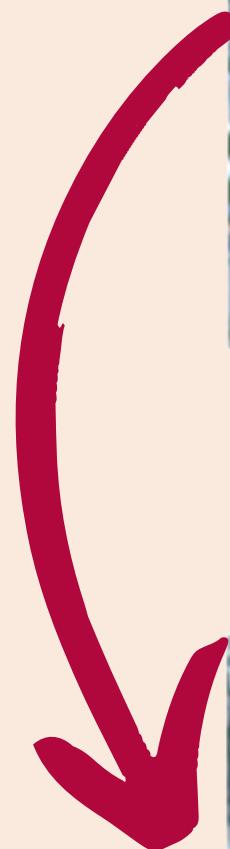
- \* La classe "**pas de données**" n'est pas présente dans le dataset d'entraînement mais existe toujours par héritage du projet S2GLC
- \* Les **nuages** sont dépendant du temps et n'ont pas de lien avec la couverture du sol on ne cherche donc pas à les prédire
- \* Un vecteur de poids est associé à chaque classe, il est fixé à zéro pour les deux classes ci-dessus pour les autres classes il est calculé de la manière suivante :

$$p_i = \frac{\sum_{j=2}^{nb\ classe} nb\ pixel \in C_j}{nb\ pixel \in C_i * (nb\ classe - 2)}$$

# CONNAISSANCE METIER

## *Data augmentation*

- \* Avant d'être présentées au modèle les images subissent des transformations aléatoires :
  - Rotation de 90° ou 270°
  - Symétrie axiale verticale ou horizontale
- \* Ces transformations permettent de rendre le modèle plus robuste.



# CONNAISSANCE METIER

## *Que prédire ?*

La proportion des classes :

**Réseau convolutionnel classique** avec une/des couches linéaires en sortie.

\* Signal de supervision faible lors de l'entraînement. Risque de conduire à des difficultés d'apprentissage

\* Optimisation directement liée à l'évaluation du challenge.

Un masque de segmentation :

**Réseau convolutionnel atypique** sans couches linéaires.

\* Signal de supervision plus fort donc apprentissage plus facile.

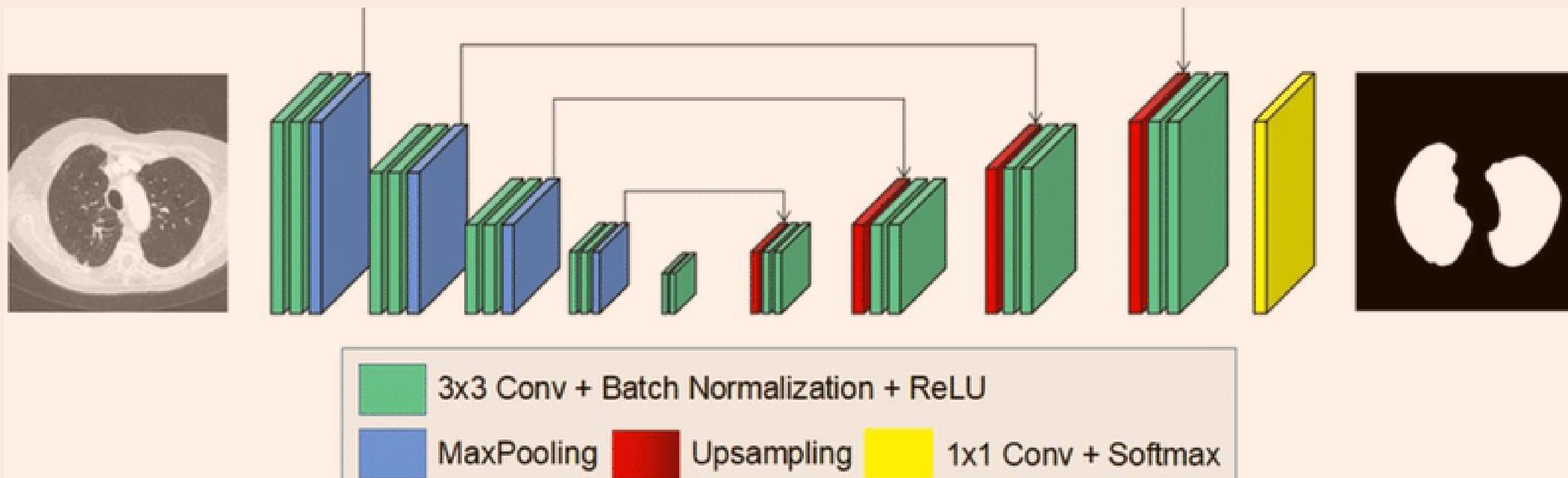
\* Optimisation indirectement liée à l'évaluation du challenge.

# ETAT DE L'ART

## *Présentation du benchmark*



Basé sur architecture **U-net** : réseau entièrement convolutionnel



# DESCRIPTION DE LA MÉTHODE

## *Analyse du benchmark*

Initialement, le benchmark utilise **tensorflow**

Plusieurs options s'offrent à nous :

- 1 Reprendre à zéro en pytorch
- 2 Garder tensorflow



On choisit de garder tensorflow afin de conserver les fonctionnalités déjà implémentées

# DESCRIPTION DE LA MÉTHODE

## *Mise en place du benchmark*



**AVANT**

Correction des erreurs causées par iPython

**APRÈS**

Adaptation du code pour fonctionner sur les GPUs

# DESCRIPTION DE LA MÉTHODE

## *Correction du benchmark*



Vecteur de poids associé aux classes erroné  
[ 0, 0, x1, x2, x3, x4, x5, x6 ]



Seulement 8 valeurs pour 10 classes en incluant les deux 0  
des classes que l'on ne veut pas prédire.  
[0.0, 0.0, 2.5, 0.37, 0.55, 1.6, 0.45, 15, 1.4e+5, 5.2]

# DESCRIPTION DE LA MÉTHODE

## *Difficultés rencontrées (1)*

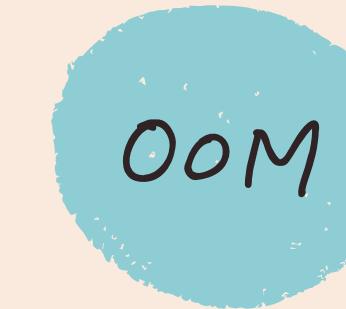
Modèle beaucoup trop grand ~ 1M paramètres



Entrainement dure 7 jours  
sur CPU



Entrainement dure 10  
heures sur les GPUs de  
l'école MAIS pas toujours  
disponibles



Erreur Out of Memory lorsque  
la taille du batch est  
supérieure à 16  
Un hyperparamètre en  
moins



```
tensorflow.python.framework.errors_impl.ResourceExhaustedError: OOM when allocating tensor with shape[32,96,256,256] and type float on /  
job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU_0_bfc [Op:Conv2D]
```

# DESCRIPTION DE LA MÉTHODE

## *Difficultés rencontrées (2)*

**PosixPath** incompatible avec python



Une **erreur** est retournée : BaseDir/"new\_folder"

Utilisation de **os** pour gérer les paths et créer les nouveaux dossiers



Peut-être que c'était compatible avec iPython

# DESCRIPTION DE LA MÉTHODE

## *Difficultés rencontrées (3)*

Dataset en tensorflow sans documentation



Type Batch/Prefetch/... incompatibles avec les méthodes de gridsearch pour les  
**hyper-paramètres**

Voir annexe page 26 pour plus de détails

# DESCRIPTION DE LA MÉTHODE

## *Cross Validation*



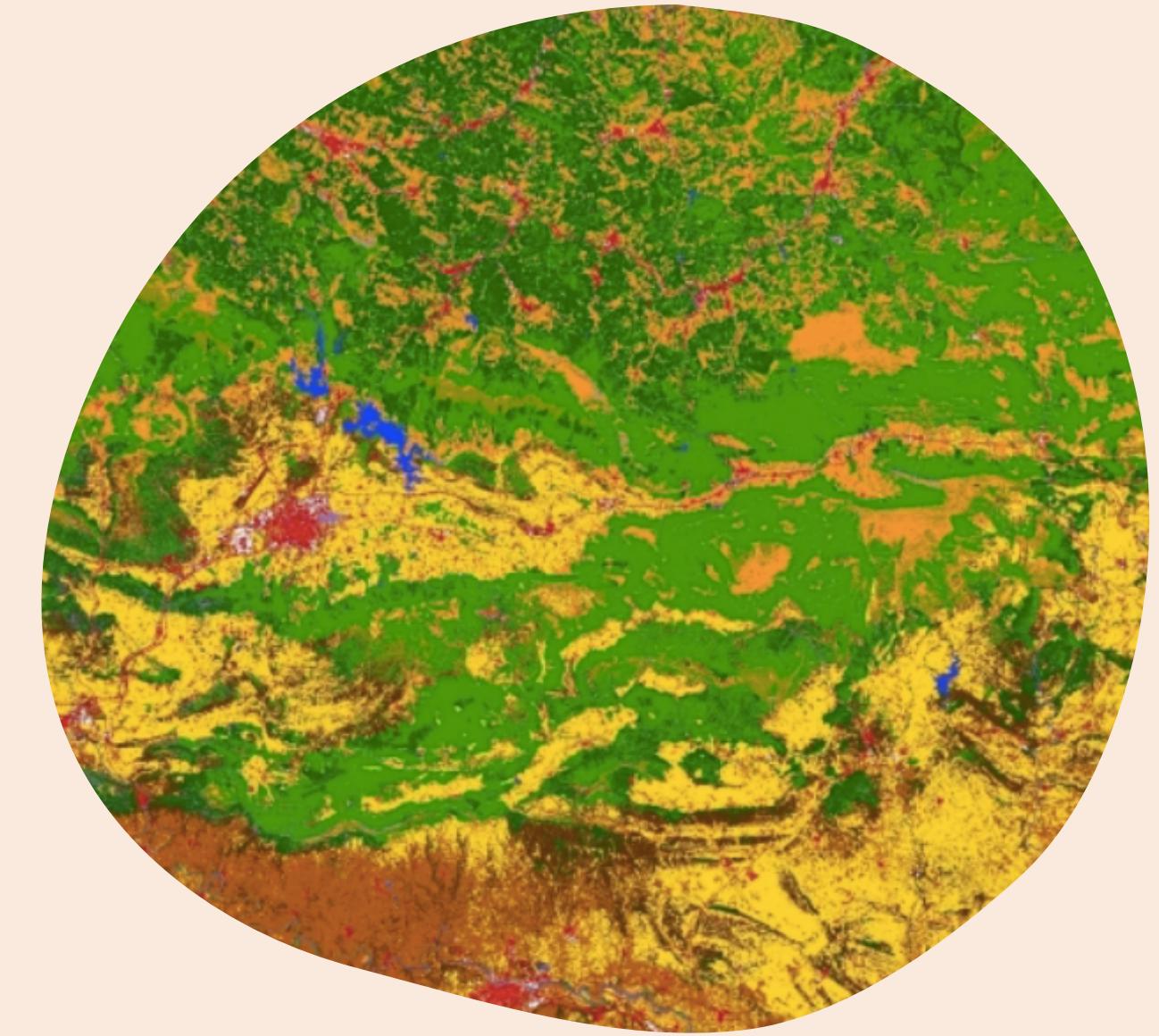
Implémentation du grid search a la main via une cross validation



Séparation du dataset en 5 folds puis entraînement sur 4 folds et validation sur le dernier



Cross validation beaucoup trop longue ~75h pour évaluer une combinaison d'hyper-paramètre



# DESCRIPTION DE LA MÉTHODE

## *Hyper-paramètres*



Essayer d'optimiser la descente de gradient :

Adam

RMSprop

Adagrad



Batch size : impossible de tester en raison de la grande quantité de RAM demandée par le modèle.



Essayer d'optimiser la learning rate pour la méthode de descente de gradient choisie

Cependant, certaines difficultés ont été rencontrées.

# DESCRIPTION DE LA MÉTHODE

## *Rigidité du modèle*

Tâche très précise en raison de l'absence de background laissant **peu de place à l'innovation**

Prédiction du masque :  
modèle **descendant puis ascendant** quasi obligatoire

Modèle **UNet** avec peu de souplesse car les couches sont composées de plusieurs **couches élémentaires**

Voir annexe page 29

# DESCRIPTION DE LA MÉTHODE

## *Métrique customisée*



La loss utilisée est la **Sparse Categorical Crossentropy** ce qui est en accord avec l'objectif de prediction de masque de segmentation mais pas avec l'évaluation du challenge.

Il faut donc modifier la métrique pour sélectionner le modèle minimisant la **divergence de Kullback-Leibler** sur le validation set.

# RÉSULTATS EXPÉIMENTAUX

## *Evaluation du challenge*

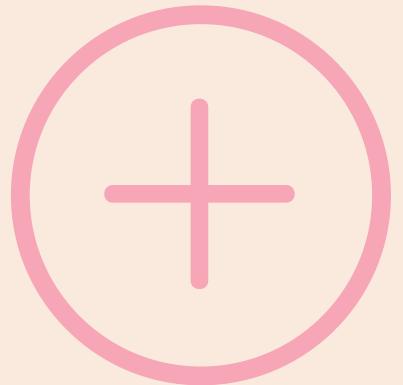
Pour l'évaluation, un fichier .csv contenant **l'identifiant** et la **proportion** de chaque classe pour les test set est **soumis sur le site de l'ENS** et un classement sur la moitié publique du test set est alors disponible.

Classement au 12/03 :

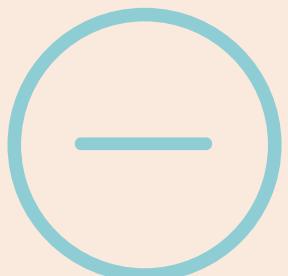
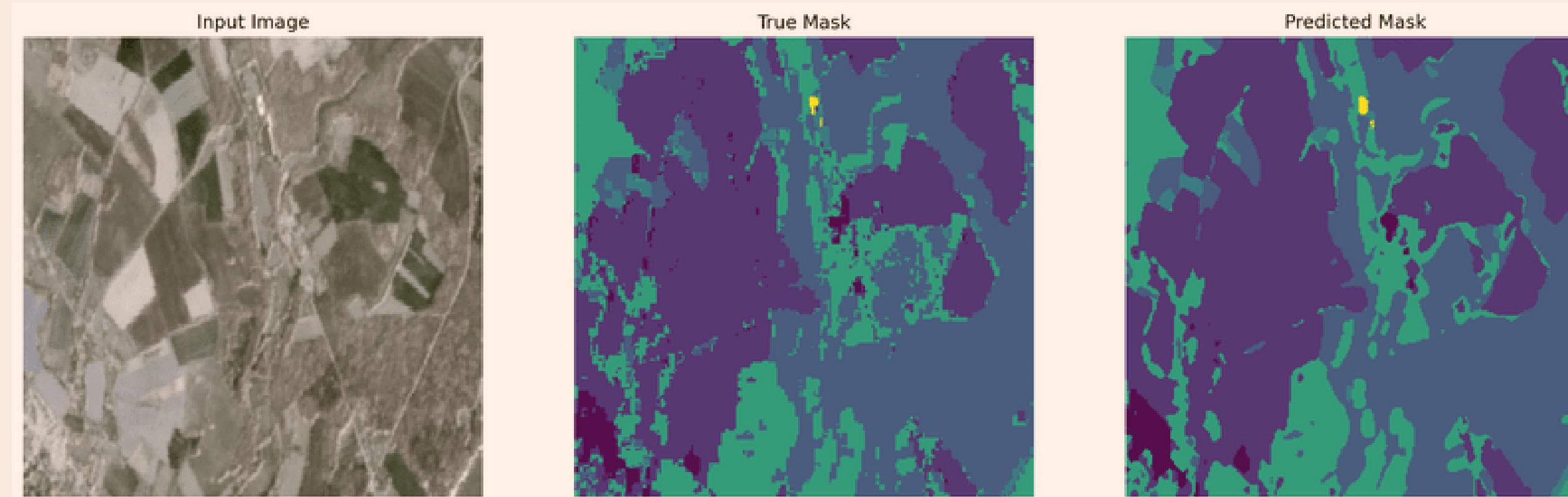
4ème place : KLD = 0,0477

5ème place Benchmark : KLD = 0,0593

# CONCLUSION



La performance atteinte est satisfaisante car le benchmark a été nettement amélioré notamment grâce à la correction d'une erreur.



L'idée de base de conserver le benchmark plutôt que de repartir de 0 nous a enfermé dans la librairie tensorflow que nous ne maîtrisions pas et contenant des types Dataset sans documentation.

# ANNEXES



# Gridsearch

Le dataset est du type `tf.data.Dataset` avec un tuple contenant l'image d'entrée et le masque de sortie.

Cette forme de donnée n'est pas compatible avec les algorithmes de gridsearch.

Transformer directement le dataset en liste est trop couteux pour les GPUs de l'école : le process est interrompu pour les préserver.

```
(base) [gpusdil_5@sh15:~]$ free -h
              total        used        free      shared  buff/cache   available
Mem:          31G         30G       258M        8.0M      163M        71M
Swap:         32G         13G        18G

(base) [gpusdil_5@sh15:~]$ nvidia-smi
Fri Mar 12 20:04:13 2021
+-----+
| NVIDIA-SMI 450.102.04    Driver Version: 450.102.04    CUDA Version: 11.0 |
+-----+
| GPU  Name      Persistence-M | Bus-Id      Disp.A  | Volatile Uncorr. ECC | | | |
| Fan  Temp     Perf  Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
|          |          |          |          |                | MIG M. |
+-----+
| 0  GeForce GTX 108... Off  00000000:65:00.0 Off | N/A          |
| 20%   31C     P8    9W / 250W | 10715MiB / 11176MiB | 0%     Default N/A |
+-----+
+-----+
| Processes:                               GPU Memory |
| GPU  GI  CI   PID  Type  Process name        Usage  |
|       ID  ID
+-----+
| 0  N/A  N/A  1966    G  /usr/lib/xorg/Xorg      9MiB  |
| 0  N/A  N/A  2024    G  /usr/bin/gnome-shell    6MiB  |
| 0  N/A  N/A  17917   C  python3                  10695MiB |
+-----+
```

```
(challenge-ens) [gpusdil_5@sh15:~/Documents/LandCover_Preligens]$ python3 framework/train_gridsearch.py --config train_config.yaml
Using TensorFlow backend.
GPUs : [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]

Config:
YamlNamespace(batch_size=16, dataset_folder=PosixPath('/usr/users/gpusdil/gpusdil_5/Documents/LandCover_Preligens/dataset'), epochs=90, lr=0.0075, num_folds=5, seed=42, val_samples_csv=None, xp_rootdir=PosixPath('/usr/users/gpusdil/gpusdil_5/Documents/LandCover_Preligens/code/experiments'))
Instantiate train and validation datasets
Will use class weights: [0.00000000e+00 0.00000000e+00 2.48438814e+00 3.74234916e-01
 5.46819219e-01 1.57013485e+00 4.54326487e-01 1.54957668e+01
 1.41433886e+05 5.15142753e+00]
Killed
```

On peut essayer de faire la séparation inputs/targets à la main mais on obtient des OoM errors si l'on considère trop d'images.

Si l'on considère suffisamment peu d'images l'algorithme de gridsearch plante à cause du type d'entier utilisé par tf (uint8) qui n'est pas valable en tant qu'indice.



```
inputs = tf.Variable(np.empty((0,256, 256, 4), dtype=np.float32))
targets = tf.Variable(np.empty((0,256, 256, 1), dtype=np.uint8))
print(trainset_size)
for x,y in train_dataset:
    inputs = tf.concat([inputs,x], axis = 0)
    targets = tf.concat([targets,y], axis = 0)
#print(inputs.shape)
```

```
(challenge-ens) [gpusdil_5@sh15:~/Documents/LandCover_Preligens]$ python3 framework/train.py --config train_config.yaml
Using TensorFlow backend.
Config:
YamlNamespace(batch_size=16, dataset_folder=PosixPath('/usr/users/gpusdil/gpusdil_5/Documents/LandCover_Preligens/dataset'), epochs=90, lr=0.0075, num_folds=5, seed=42, val_samples_csv=None, xp_rootdir=PosixPath('/usr/users/gpusdil/gpusdil_5/Documents/LandCover_Preligens/code/experiments'))
Instantiate train and validation datasets
18491
Traceback (most recent call last):
  File "framework/train.py", line 247, in <module>
    inputs = tf.concat([inputs,x], axis = 0)
  File "/usr/users/gpusdil/gpusdil_5/miniconda3/envs/challenge-ens/lib/python3.7/site-packages/tensorflow_core/python/util/dispatch.py", line 180, in wrapper
    return target(*args, **kwargs)
  File "/usr/users/gpusdil/gpusdil_5/miniconda3/envs/challenge-ens/lib/python3.7/site-packages/tensorflow_core/python/cps/array_ops.py", line 1431, in concat
    return gen_array_ops.concat_v2(values=values, axis=axis, name=name)
  File "/usr/users/gpusdil/gpusdil_5/miniconda3/envs/challenge-ens/lib/python3.7/site-packages/tensorflow_core/python/ops/gen_array_ops.py", line 1249, in concat_v2
    _six.raise_from(_core._status_to_exception(e.code, message), None)
  File "<string>", line 3, in raise_from
tensorflow.python.framework.errors_impl.ResourceExhaustedError: OOM when allocating tensor with shape[3472,256,256,4] and type float on job b:localhost(replica:0/task:0/device:GPU:0 by allocator GPU_0_bfc [Op:ConcatV2] name: concat
```

```
TypeError: Only integers, slices (`:`), ellipsis (`...`), tf.newaxis (`None`) and scalar tf.int32/tf.int64 tensors are valid indices, got
array([ 617,  618,  619, ..., 3078, 3079, 3080])
```

Une méthode permet de convertir le type Dataset en NumPy array mais elle n'est disponible que a partir de tf 2.1 or un warning apparait disant que les GPUs doivent tourner avec tf 2.0.0.

```
(challenge-ens) [gpusdil_5@sh15:~/Documents/LandCover_Preligens]$ conda update tensorflow
Collecting package metadata (current_repodata.json): done
Solving environment: \

Updating tensorflow is constricted by

tensorflow-gpu -> requires tensorflow==2.0.0
```

Pour éviter de rendre la machine inutilisable tf n'a pas été upgrade.

```

Creating U-Net with arguments: {'input_shape': (256, 256, 4), 'num_classes': 10, 'num_layers': 2}
Model: "unet"
Layer (type)          Output Shape       Param #  Connected to
=====
input_1 (InputLayer)   [(None, 256, 256, 4)] 0
conv2d (Conv2D)        (None, 256, 256, 64) 2368    input_1[0][0]
batch_normalization (BatchNorm) (None, 256, 256, 64) 256    conv2d[0][0]
conv2d_1 (Conv2D)      (None, 256, 256, 64) 36928   batch_normalization[0][0]
batch_normalization_1 (BatchNorm) (None, 256, 256, 64) 256    conv2d_1[0][0]
conv2d_2 (Conv2D)      (None, 256, 256, 64) 36928   batch_normalization_1[0][0]
max_pooling2d (MaxPooling2D) (None, 128, 128, 64) 0    conv2d_2[0][0]
batch_normalization_2 (BatchNorm) (None, 128, 128, 64) 256    max_pooling2d[0][0]
conv2d_3 (Conv2D)      (None, 128, 128, 64) 36928   batch_normalization_2[0][0]
batch_normalization_3 (BatchNorm) (None, 128, 128, 64) 256    conv2d_3[0][0]
conv2d_4 (Conv2D)      (None, 128, 128, 64) 36928   batch_normalization_3[0][0]
batch_normalization_4 (BatchNorm) (None, 128, 128, 64) 256    conv2d_4[0][0]
conv2d_5 (Conv2D)      (None, 128, 128, 64) 36928   batch_normalization_4[0][0]
max_pooling2d_1 (MaxPooling2D) (None, 64, 64, 64) 0    conv2d_5[0][0]
batch_normalization_5 (BatchNorm) (None, 64, 64, 64) 256    max_pooling2d_1[0][0]
conv2d_6 (Conv2D)      (None, 64, 64, 64) 36928   batch_normalization_5[0][0]
batch_normalization_6 (BatchNorm) (None, 64, 64, 64) 256    conv2d_6[0][0]
conv2d_7 (Conv2D)      (None, 64, 64, 64) 36928   batch_normalization_6[0][0]
batch_normalization_7 (BatchNorm) (None, 64, 64, 64) 256    conv2d_7[0][0]
conv2d_8 (Conv2D)      (None, 64, 64, 64) 36928   batch_normalization_7[0][0]
max_pooling2d_2 (MaxPooling2D) (None, 32, 32, 64) 0    conv2d_8[0][0]
batch_normalization_8 (BatchNorm) (None, 32, 32, 64) 256    max_pooling2d_2[0][0]
conv2d_9 (Conv2D)      (None, 32, 32, 64) 36928   batch_normalization_8[0][0]
batch_normalization_9 (BatchNorm) (None, 32, 32, 64) 256    conv2d_9[0][0]
conv2d_10 (Conv2D)     (None, 32, 32, 64) 36928   batch_normalization_9[0][0]
batch_normalization_10 (BatchNorm) (None, 32, 32, 64) 256    conv2d_10[0][0]
conv2d_transpose (Conv2DTranspose) (None, 64, 64, 64) 36928  batch_normalization_10[0][0]
concatenate (Concatenate) (None, 64, 64, 128) 0    conv2d_transpose[0][0]
                                         conv2d_7[0][0]
batch_normalization_11 (BatchNorm) (None, 64, 64, 128) 512    concatenate[0][0]
conv2d_11 (Conv2D)      (None, 64, 64, 96) 110688  batch_normalization_11[0][0]
batch_normalization_12 (BatchNorm) (None, 64, 64, 96) 384    conv2d_11[0][0]
conv2d_12 (Conv2D)      (None, 64, 64, 64) 55360   batch_normalization_12[0][0]

```

Une "couche"  
descendante  
de UNet

	(None, 64, 64, 64)	55360	batch_normalization_12[0][0]
conv2d_12 (Conv2D)	(None, 64, 64, 64)	55360	batch_normalization_12[0][0]
batch_normalization_13 (BatchNorm)	(None, 64, 64, 64)	256	conv2d_12[0][0]
conv2d_transpose_1 (Conv2DTranspose)	(None, 128, 128, 64)	36928	batch_normalization_13[0][0]
concatenate_1 (Concatenate)	(None, 128, 128, 128)	0	conv2d_transpose_1[0][0]
conv2d_4 (Conv2D)			conv2d_4[0][0]
batch_normalization_14 (BatchNorm)	(None, 128, 128, 128)	512	concatenate_1[0][0]
conv2d_13 (Conv2D)	(None, 128, 128, 96)	110688	batch_normalization_14[0][0]
batch_normalization_15 (BatchNorm)	(None, 128, 128, 96)	384	conv2d_13[0][0]
conv2d_14 (Conv2D)	(None, 128, 128, 64)	55360	batch_normalization_15[0][0]
batch_normalization_16 (BatchNorm)	(None, 128, 128, 64)	256	conv2d_14[0][0]
conv2d_transpose_2 (Conv2DTranspose)	(None, 256, 256, 64)	36928	batch_normalization_16[0][0]
concatenate_2 (Concatenate)	(None, 256, 256, 128)	0	conv2d_transpose_2[0][0]
conv2d_1 (Conv2D)			conv2d_1[0][0]
batch_normalization_17 (BatchNorm)	(None, 256, 256, 128)	512	concatenate_2[0][0]
conv2d_15 (Conv2D)	(None, 256, 256, 96)	110688	batch_normalization_17[0][0]
batch_normalization_18 (BatchNorm)	(None, 256, 256, 96)	384	conv2d_15[0][0]
conv2d_16 (Conv2D)	(None, 256, 256, 64)	55360	batch_normalization_18[0][0]
conv2d_17 (Conv2D)	(None, 256, 256, 10)	650	conv2d_16[0][0]
Total params:	987,242		
Trainable params:	984,234		
Non-trainable params:	3,008		
None			
Compile model			
Will use class weights:	[0.0000000e+00 0.0000000e+00 2.48438814e+00 3.74234916e-01		
	5.46819219e-01 1.57013485e+00 4.54326487e-01 1.54957668e+01		
	1.41433886e+05 5.15142753e+00]		