# Rapport PAO S2

Ciel mon point d'eau

Rei Ito, Jihane Essakhi, Pierre-Marie Stevenin

Mai 2024

## 1 Introduction

Le sujet consiste à extraire des contours d'objets hydrauliques, à partir des données satellites, par l'utilisation d'outils de deep learning.

Le projet est mené en collaboration avec l'Université de Ceara (Brésil).

Avant tout, nous souhaitons remercier Dr.Marielle Gosset de nous fournir les données et les aides nécessaires à la bonne poursuite du projet. De plus, nous exprimons un remerciement distingué à M.Gilles Gasso de nous encadrer tout au long du projet.

Les données satellites proviennent du satellite SWOT (Surface Water and Ocean Topography) du CNES et de la NASA, le premier satellite à interférométrie micro-onde haute résolution dédié à l'hydrologie. La zone travaillée est la région semi-aride du Ceara dans le Nordeste Brésilien, où le satellite a traversé de manière journalière au cours de sa phase de calibration.

## 2 Sélection de données

### 2.1 Dataset

Nous utilisons des données issues du satellite SWOT (Surface Water and Ocean Topography) du CNES et de la NASA, et des satellites Sentinel S1 et S2. Sentinel S1 est utilisé pour les images optiques tandis que Sentinel S2 fournit des images dans le spectre infrarouge. Ces sources de données sont choisies pour leur capacité à fournir des informations détaillées et complémentaires sur les caractéristiques hydriques des zones étudiées.

### 2.2 Familiarisation avec QGIS

L'utilisation de QGIS, un système d'information géographique open source, nous a permis de visualiser et d'analyser les images satellites. Cette étape de familiarisation est essentielle pour manipuler efficacement les données géospatiales, permettant de superposer, de comparer et de corriger les images selon les besoins du projet.

### 2.3 Critères de sélection des images

Les images sont sélectionnées selon plusieurs critères qualitatifs afin de maximiser la précision de l'étude :

- **Absence de dépassement** : Nous excluons les images sans informations [inclure image ou il ya un dépassement ]
- **Minimisation des pixels invalides** : Les images retenues doivent contenir le moins possible de pixels invalides. Des masques spéciaux sont appliqués pour identifier et exclure les pixels problématiques des analyses ultérieures. Pour Sentinel S1, les pixels invalides sont principalement dus à la présence de nuages, tandis que pour Sentinel S2, ils peuvent également être causés par des dépassements.

### 2.4 Liste de données finale

La sélection finale des images stockée dans un dossier nommé 'Partie 1', qui contient environ 30 Go de données, est stockée dans notre fichier data set et représente 4.5 Go de données. Ces images proviennent des sous-dossiers suivants, tous sélectionnés en fonction des critères établis :

- Sentinel S2 et S1 du dossier 6
- Sentinel S2 et S1 du dossier 9
- Sentinel S2 et S1 du dossier 16
- Sentinel S2 et S1 du dossier 15
- Sentinel S2 et S1 du dossier 13
- Sentinel S2 et S1 du dossier 11

Cette sélection rigoureuse des données garantit que les images utilisées pour l'analyse reflètent le plus fidèlement possible les conditions réelles des zones étudiées, tout en minimisant les interférences et les erreurs potentielles.

## 3 Découpage en imagettes

### 3.1 Taille initiale des images

Les images satellites traitées proviennent des satellites SWOT, Sentinel S1 pour l'optique et Sentinel S2 pour l'infrarouge, couvrant des superficies de plusieurs kilomètres carrés chacune. La grande taille de ces images rend leur traitement direct impraticable pour des analyses détaillées et la segmentation d'image en raison des exigences élevées en termes de puissance de calcul et de gestion de la mémoire.

## 3.2 Processus de découpage des images

Le processus de découpage en imagettes est réalisé par plusieurs fonctions spécialisées en Python utilisant la bibliothèque `rasterio`, qui est efficace pour manipuler des images géospatiales. Voici une explication détaillée de chaque fonction et de leur rôle dans le traitement des images.

## 3.3 Spécification de la façon de découper

Le processus de découpage des images Sentinel S1 et S2 est conçu pour sélectionner des imagettes ayant un équilibre entre les zones aquatiques et terrestres. Seules les parties des images où au moins 10% des pixels représentent de l'eau et au moins 10% des pixels représentent de la terre sont retenues. Cette approche assure une diversité dans les données d'entraînement, évitant ainsi les biais potentiels dans la segmentation finale.

## 3.4 Threshold de sélection

Un seuil de sélection rigoureux est appliqué pour assurer que les imagettes contiennent un mélange représentatif de caractéristiques hydrauliques et terrestres, contribuant à une meilleure généralisation des modèles de segmentation sur différentes scènes.

### 3.4.1 Fonction `split_selected_masks`

Cette fonction est conçue pour traiter des images masquées, sélectionnant des imagettes qui contiennent une proportion équilibrée de zones aquatiques et terrestres. Elle lit une grande image, la divise en sous-images de 256x256 pixels et filtre ces sous-images pour conserver uniquement celles ayant entre 10% et 90% de pixels d'eau, assurant ainsi une variété dans les types de terrain pour l'entraînement du modèle.

### 3.4.2 Fonction `scale_min_max`

Cette fonction simple ajuste la mise à l'échelle des valeurs des pixels dans chaque bande spectrale d'une imagette pour normaliser les valeurs entre 0 et 1, en utilisant un maximum défini (par exemple 10000 pour Sentinel S2). Cette normalisation est cruciale pour le traitement par des modèles de deep learning, qui performe mieux avec des données normalisées.

### 3.4.3 Fonctions `split_selected_s2_images` et `split_selected_s1_images`

Ces fonctions appliquent un processus similaire pour les images de Sentinel S2 et S1 respectivement, où elles recadrent les imagettes basées sur les indices sélectionnés par `split_selected_masks`. Chaque imagette est ensuite re-mise à l'échelle, normalisée, et sauvegardée dans un format approprié pour une utilisation ultérieure. La différence principale entre ces deux fonctions réside dans

le traitement spécifique des bandes spectrales, adapté aux caractéristiques des images S1 et S2.

#### 3.4.4 Fonction `split_data`

Cette fonction orchestre l'ensemble du processus de découpage, en s'assurant que les dossiers nécessaires existent et en appelant les autres fonctions dans un ordre logique. Elle commence par les masques, suit avec les images Sentinel S2, puis les images S1, et retourne le nombre total d'imagettes sélectionnées et traitées.

### 3.5 Importance de la préparation des données

Le soin apporté au découpage et à la préparation des imagettes est essentiel pour garantir la qualité et la pertinence des données pour l'entraînement des modèles de deep learning. Ce processus méthodique permet d'assurer une couverture complète des différentes caractéristiques géographiques de la zone étudiée et améliore significativement les performances du modèle de segmentation des images satellite.

### 3.6 Tailles des imagettes

Les imagettes résultantes mesurent 256 pixels par 256 pixels. Cette dimension est choisie pour optimiser la résolution nécessaire à la capture de détails significatifs tout en conservant une efficacité de traitement acceptable.

### 3.7 Découpage avec superposition des zones

Le découpage peut également être effectué avec une superposition entre les imagettes adjacentes, permettant de maintenir une continuité des caractéristiques géographiques et d'augmenter la quantité de données d'entraînement disponibles sans perte significative d'information. Cette technique est particulièrement utile pour les modèles qui traitent des caractéristiques à la limite des imagettes.

### 3.8 Dataset final

Après le processus de découpage et de sélection, le dataset final comprend 2096 imagettes, chacune analysée et prête à être utilisée pour l'entraînement des modèles de deep learning. Cette quantité substantielle d'imagettes garantit une couverture exhaustive des diverses caractéristiques hydrauliques et topographiques de la région étudiée.

Cette méthode de préparation des données est essentielle pour optimiser les entrées pour les étapes de modélisation et d'analyse, assurant ainsi une efficacité maximale dans la détection et la segmentation des points d'eau à partir des images satellite.

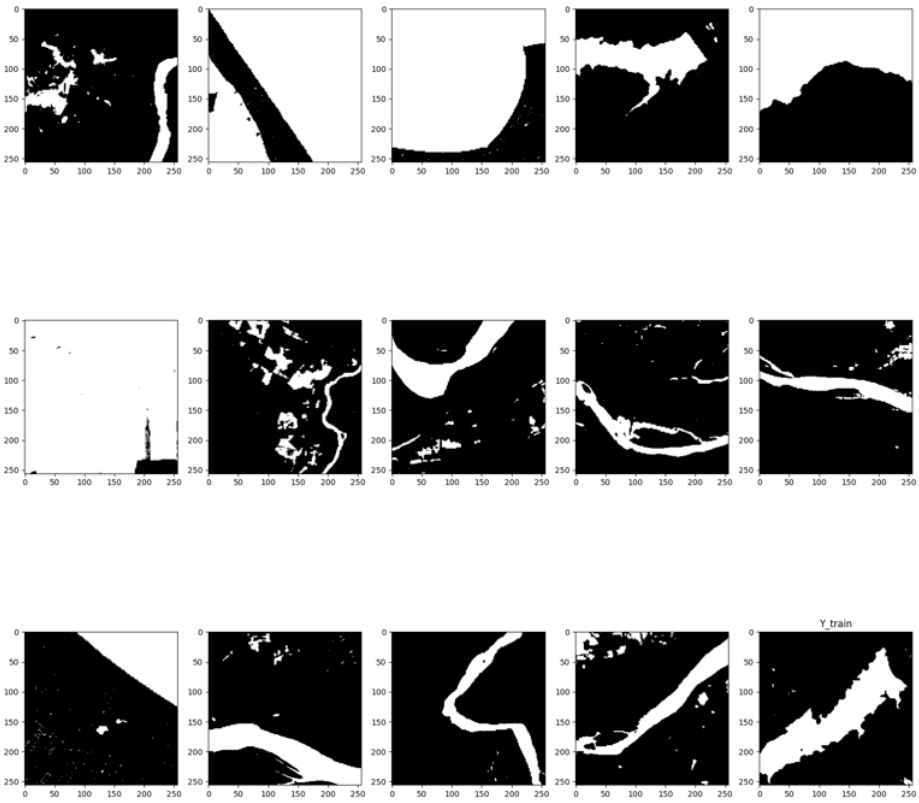



Figure 1: Masque vérité

# 4 Modèle d'entrainement

## 4.1 Introduction aux modèles de segmentation d'images

La segmentation d'images est une branche critique du traitement d'images en deep learning, visant à classer chaque pixel d'une image dans une catégorie spécifique. Cette technique est essentielle dans de nombreux domaines tels que la médecine, la surveillance, et, comme dans notre cas, l'analyse environnementale. Les modèles de deep learning, notamment ceux basés sur les architectures en U comme le U-Net, sont particulièrement efficaces pour cette tâche grâce à leur capacité à travailler avec des images de grande taille tout en capturant les détails au niveau des pixels.

## 4.2 Choix du modèle : U-Net

Pour notre projet de détection des points d'eau, nous avons choisi l'architecture U-Net, initialement développée pour la segmentation médicale. L'U-Net est idéal pour la segmentation d'images satellite où il est crucial de déterminer précisément si un pixel appartient à un point d'eau. Cette architecture est caractérisée par sa structure en forme de "U", qui comprend une phase de contraction pour capturer le contexte et une phase d'expansion pour permettre une localisation précise.

## 4.3 Architecture de notre modèle

Notre modèle U-Net est implémenté avec la configuration suivante:

```
def unet_model(input_shape=(256, 256, 8), num_classes=1, dropout=0.5, batchnorm=True):
    inputs = Input(input_shape)

    # Contracting Path
    c1 = conv2d_block(inputs, 16, batchnorm=batchnorm)
    p1 = MaxPooling2D((2, 2)) (c1)
    p1 = Dropout(dropout*0.5)(p1)
    ...
    u11 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same') (c10)
    u11 = concatenate([u11, c1])
    u11 = Dropout(dropout)(u11)
    c11 = conv2d_block(u11, 16, batchnorm=batchnorm)

    outputs = Conv2D(num_classes, (1, 1), activation='sigmoid') (c11)

    model = Model(inputs=[inputs], outputs=[outputs])
    model.compile(optimizer='adam', loss=dice_bce_loss, metrics=['accuracy'])
```

Dans cette architecture, nous utilisons des convolutions pour réduire progressivement la dimensionnalité de l'espace des caractéristiques, suivies de couches

de pooling pour réduire la taille spatiale de l'image. Chaque étape du chemin de contraction est suivie par une étape correspondante dans le chemin d'expansion, où des convolutions transposées sont utilisées pour augmenter la taille de l'image. Les concaténations avec les caractéristiques correspondantes du chemin de contraction permettent de conserver les informations contextuelles, essentielles pour une segmentation précise. La fonction d'activation sigmoid plutôt que softmax est due à la configuration des données en entrée, mais ce paramètre peut évoluer à l'avenir.

### 4.4 Convolution et Déconvolution

Les couches de convolution permettent d'extraire les caractéristiques pertinentes de l'image, tandis que les déconvolutions (ou convolutions transposées) sont utilisées pour reconstruire l'image segmentée à partir des caractéristiques extraites. Cette combinaison assure que notre modèle peut à la fois analyser et synthétiser l'information spatiale des images.

### 4.5 Choix de l'optimiseur : Adam

Nous avons sélectionné Adam comme optimiseur pour son efficacité dans les tâches de deep learning. Adam combine les avantages des approches adaptatives de taux d'apprentissage, permettant une convergence plus rapide et plus stable même avec des paramètres initiaux loin des valeurs optimales.

### 4.6 Choix de la fonction de perte : Entropie croisée binaire et Dice Loss

Nous utilisons une combinaison de l'entropie croisée binaire et de la Dice Loss pour notre fonction de perte. L'entropie croisée binaire est efficace pour les tâches de classification binaire pixel par pixel, tandis que la Dice Loss est particulièrement adaptée pour équilibrer la segmentation dans les zones de classe déséquilibrée, ce qui est fréquent dans les images où les zones d'eau sont moins présentes que les zones terrestres.

## 5 Entraînement du modèle

### 5.1 Premières expérimentations

Les premières tentatives d'entraînement de notre modèle U-Net comportaient uniquement deux couches de convolution, une décision principalement motivée par les limitations techniques rencontrées, notamment des problèmes de stabilité du noyau (kernel dead). Nous utilisions une faible quantité de données d'entrainement, environ 50 et un peu d'epochs, juste deux ou trois. Cette configuration réduite a été choisie pour tester la viabilité du modèle dans un environnement avec ressources limitées, nos PCs personnels. Evidemment, cette simplification a conduit à des résultats médiocres, indiquant que le modèle n'était

pas suffisamment complexe pour capturer les caractéristiques nécessaires des images pour une bonne segmentation. Cependant, cette configuration nous a permis de construire une base fiable (extraction des données, pré-traitement, post-traitement), sur laquelle s'appuyer pour développer notre modèle.

### 5.2 Variation du nombre d'époques et de la taille du batch

Dans le cadre de nos tests, nous avons également varié le nombre d'époques et la taille des lots (batch size). Une expérience a été réalisée avec 10 époques d'entraînement ainsi qu'une taille du batch ajustée de 64 à 32 puis à 16. L'intérêt de modifier le batch size réside dans la gestion de la mémoire et l'impact sur le processus d'apprentissage : des lots plus petits peuvent améliorer la généralisation du modèle mais augmentent le temps de calcul, tandis que des lots plus grands peuvent accélérer l'entraînement mais au risque de moins bien généraliser. Augmenter le nombre d'époques peut potentiellement améliorer les performances du modèle en lui permettant de mieux apprendre des patterns complexes dans les données, cependant, cela augmente aussi le risque de surajustement, particulièrement avec un petit ensemble de données.

### 5.3 Post-traitement

Afin d'obtenir des résultats exploitables, les images prédites brutes nécessitent d'être binarisées. En effet, le résultat attendu est un masque binaire : 1 pour les pixels contenant de l'eau, 0 sinon. Nous utilisons la librairie OpenCV pour effectuer une binarisation adaptative sur chaque image avec la binarisation d'Otsu:

```
for img in Y_pred:
    thresholded_pred.append(cv2.threshold((255*img).round().astype('uint8'),
                                          0,1,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1])
```

### 5.4 Conclusion sur l'entraînement

Ces expériences ont révélé des contraintes significatives dues à notre environnement matériel, mais ont également fourni des indications précieuses sur les configurations optimales pour notre modèle dans le contexte spécifique de la segmentation d'images satellite. À chaque étape, l'ajustement des paramètres d'entraînement a été crucial pour maximiser les performances tout en contournant les limitations de notre système.

## 6 Résultats

Aujourd'hui notre modèle est capable, sous certaines conditions (selon les données d'entrainement ou le nombre précis d'epochs par exemple), de prédire un masque satisfaisant. L'évolution de la loss reste encore chaotique, mais des progrès clairs sont possibles.

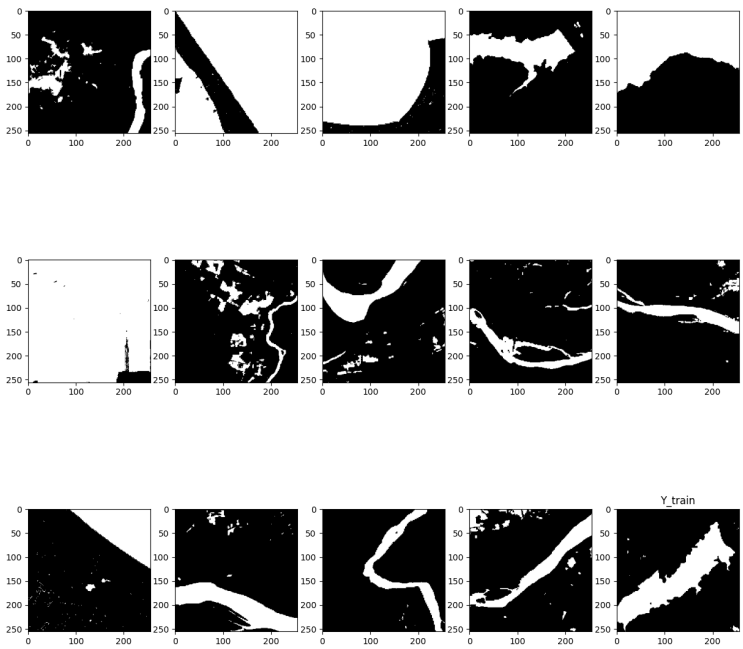



Figure 2: Masque vérité voulu

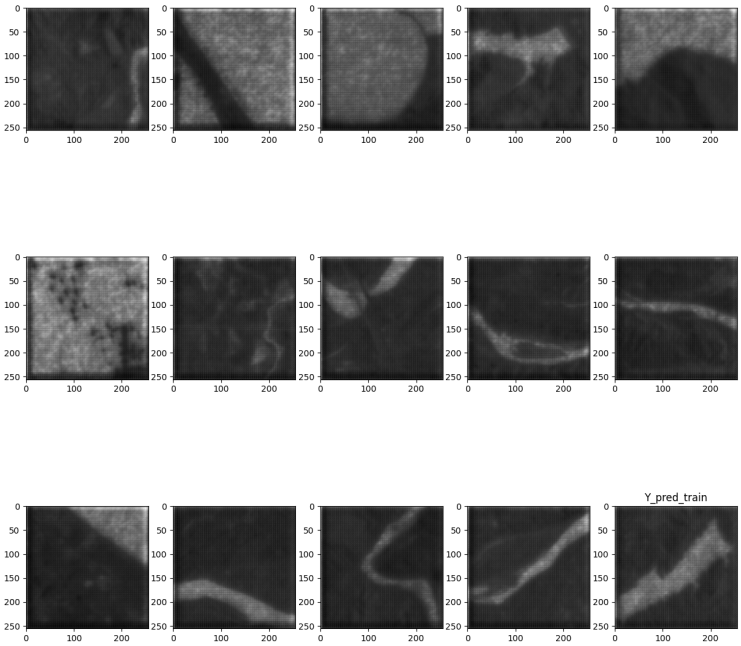



Figure 3: Masque prédit brut

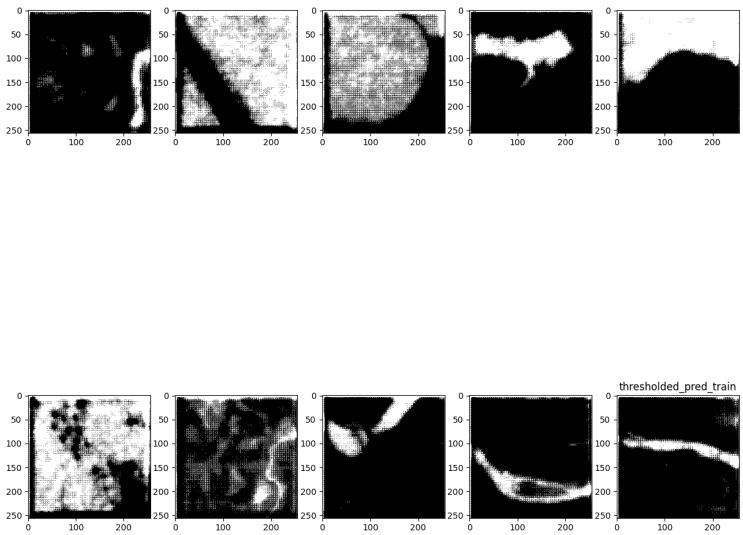
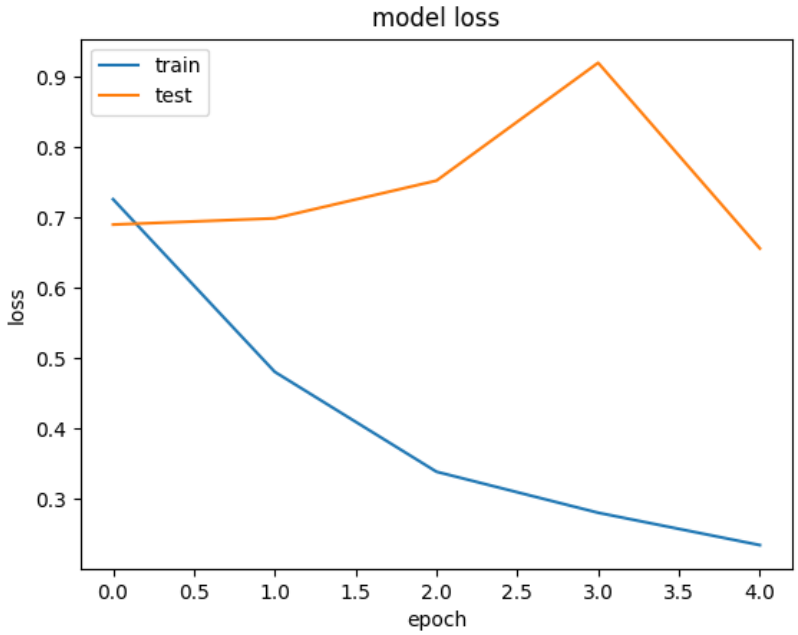



Figure 4: Masque prédit binarisé


model loss
train
test
loss
epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0