Adaptation d'un Algorithme CNN U-Net pour la Reconnaissance de Lacs sur Images Landsat

1 Téléchargement et Prétraitement des Images

1.1 Images Prétraitées

Délimitation des zones d'intérêt : La première étape pour créer des données de vérité terrain consiste à sélectionner des zones à partir des tuiles Sentinel-2 incluant des surfaces d'eau. Les zones doivent être sélectionnées à partir des fuseaux UTM 29, 30 et 31. En effet, la projection Universelle Transverse de Mercator est un type de projection cartographique conforme de la surface terrestre. La Terre est découpée en 60 fuseaux de 6 degrés, séparant l'hémisphère nord de l'hémisphère sud. Cela fait un total de 120 zones (60 pour le Nord et 60 pour le Sud) et donc 60 projections transversales différentes.

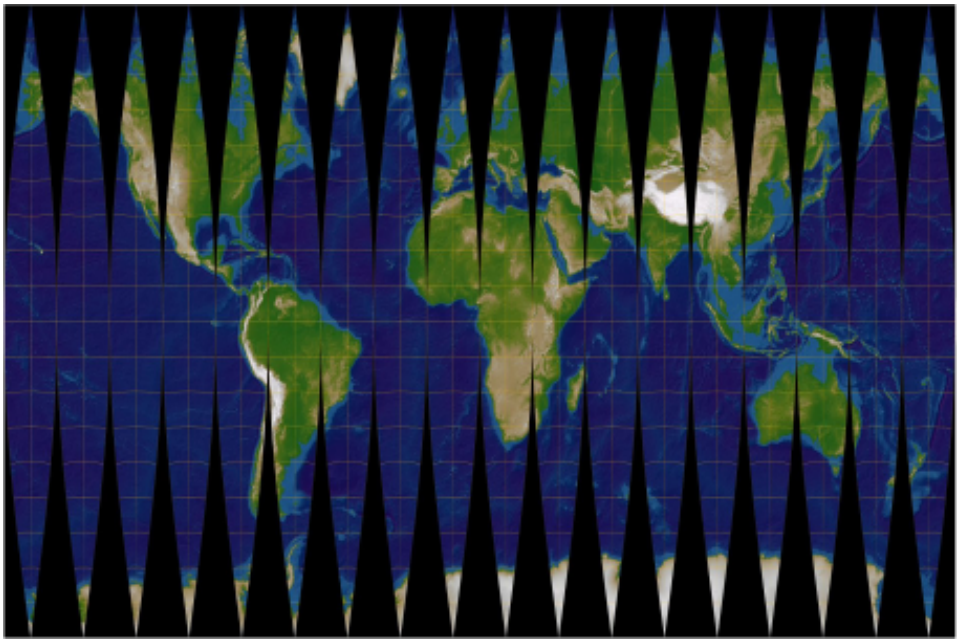

FIGURE 1 – Le principe de l'UTM pour 18 projections de fuseaux de 20° [Wikipédia].

Polygones des surfaces d'eau : Ensuite, nous devons délimiter les surfaces d'eau de différentes tailles et formes afin d'ajouter de la diversité à nos données. Cette tâche n'est pas aussi évidente qu'il n'y paraît, car parfois nous ne sommes pas complètement certains qu'un objet spécifique est bien une surface d'eau ou non. Pour plus de détails, vous pouvez consulter la thèse de Mathilde de Fleury : *Chapitre 4, Partie 1.1, p97.*
[Internship_report_DL4SahelLakes__Wassim_CHAKROUN]

Pour annoter les données, nous utilisons QGIS pour tracer les polygones qui segmentent les lacs, mais aussi les rectangles qui définissent les images qui vont être utilisées pendant l'entraînement.

Toutes les images de la tuile ne sont pas annotées, car cela serait compliqué d'annoter tous les lacs d'une tuile.

1.2 Téléchargement des Images

Avant de passer à l'algorithme, il faut télécharger les images tuiles de satellites depuis `https://earthexplorer.usgs.gov/` manuellement. Nous pouvons automatiser ce processus et trouver d'autres méthodes pour télécharger les images directement, comme l'a fait Mathilde dans le code `/mnt/md0/mathilde/sentinel2/theia_download.py`. Vous pouvez trouver les images utilisées dans le dossier `/mnt/md0/abduh` avec les codes pour ajouter les bandes MNDWI et NDWI.

2 Structure du Code

Le code est structuré autour de trois étapes principales du pipeline : `preprocessing.py`, `training.py`, `prediction.py`. Ces étapes sont appelées depuis `main.py`, et reposent sur des méthodes dans le dossier `/core`. La classe de configuration générale est stockée dans le dossier `/config/`, et est initialisée et passée au pipeline dans `main.py`.

Listing 1 – Commande pour le setup de l'environnement Python

```
# Activer l'environnement conda
conda activate tf
```

2.1 Prétraitement

Pendant la phase de prétraitement, les zones d'entraînement sont extraites des images d'entraînement et stockées dans des trames prétraitées temporaires. Chaque trame contient les canaux d'images ainsi que les deux canaux d'annotations (étiquettes et poids de frontière).

2.2 Entraînement

Pendant l'entraînement, le modèle UNet est entraîné avec les trames prétraitées et sauvegardé dans un fichier. Les métadonnées concernant le modèle et ses paramètres de configuration sont stockées dans le fichier `.h5` en tant qu'attribut.

Pour faciliter le suivi et la visualisation du processus d'entraînement, les logs sont enregistrés dans le dossier `logs`. Ces logs peuvent ensuite être visualisés avec Tensorboard, un outil pratique pour monitorer la performance des modèles d'apprentissage profond pour détecter le sur ou sous apprentissage. Pour lancer Tensorboard, il suffit d'exécuter la commande suivante dans le terminal :

Listing 2 – Lancer Tensorboard

```
tensorboard --logdir [chemin_log]
```

2.3 Prédiction

Pendant la prédiction, le modèle entraîné est utilisé pour prédire les lacs dans les images de prédiction. Les prédictions sont stockées sous forme d'images raster à canal unique.

J'ai modifié le code de prédiction original pour améliorer la gestion des valeurs de nodata. Dans le cadre de l'analyse des images satellitaires, les zones sans données (nodata) peuvent parfois être interprétées à tort comme des surfaces d'eau ou d'autres classes d'intérêt, générant ainsi des faux positifs.

Dans la version originale du code, les valeurs de nodata n'étaient pas spécifiquement gérées, ce qui pouvait entraîner des erreurs dans les prédictions. Pour résoudre ce problème, j'ai ajouté une étape dans la fonction predict_using_model où les valeurs nodata présentes dans les images d'entrée sont remplacées par zéro (0) dans les prédictions. Cela garantit que ces zones sont correctement identifiées et exclues des résultats finaux de la prédiction

3 Configuration

Avant d'exécuter `main.py`, les chemins d'accès et les paramètres généraux doivent être configurés dans un fichier de configuration `config_default.py` où vous pouvez modifier, par exemple, `prediction_batch_size=64/32` pour un `patch_size=64` car cela accélère la prédiction.

Vous pouvez créer des fichiers de configuration distincts pour différents ensembles de données ou cas d'utilisation, qui peuvent être sélectionnés dans le premier import de `main.py`. J'ai également ajouté un fichier `main_custom.py` où vous pouvez spécifier le fichier de configuration comme argument, puis lancer de nombreuses prédictions et/ou entraînements, comme montré dans l'exemple du script `script.sh`.

3.1 Exemple de Configuration dans *config_ abduh.py*

Pour faciliter l'utilisation du pipeline, il est possible de configurer les paramètres essentiels dans un fichier de configuration :

- `trained_model_path` : Le chemin vers le modèle entraîné que vous souhaitez utiliser pour les prédictions.

- `prediction_output_dir` : Le répertoire où les prédictions seront sauvegardées.

- `prediction_stride` : Le pas de prédiction, c'est-à-dire la distance entre deux patchs consécutifs sur l'image d'entrée.

- `prediction_patch_size` : La taille des patchs d'image sur lesquels le modèle effectuera des prédictions.

- `prediction_batch_size` : Le nombre de patchs d'image traités simultanément pendant la prédiction.

- `channels_used` : Une liste booléenne indiquant les canaux d'image à utiliser (ici, les trois canaux RGB).

Les autres paramètres de ce fichier de configuration sont également bien expliqués avec des commentaires.

3.2 Exécution du Pipeline

L'exécution de `main.py` lancera l'ensemble du pipeline de prétraitement, d'entraînement et de prédiction. Par défaut, l'entraînement utilisera les données de prétraitement les plus récentes qu'il peut trouver, la prédiction utilisera le modèle entraîné le plus récent, et le post-traitement les prédictions les plus récentes, mais cela peut également être configuré dans le fichier de configuration.

Pour continuer à entraîner un modèle précédemment entraîné, spécifiez son chemin dans `config.continue_model_path`.

4 Tests

Pour évaluer notre modèle, j'ai écrit le fichier `testing_stride.py`, qui utilise le même principe de configuration dont je n'ai pas trop modifié. Il y a des parties de code non nécessaires dans les configurations pour le test. Il suffit de préciser le modèle à tester via `self.trained_model_path` et de préciser le dossier prétraité qui contient les annotations et les images. Il ne faut pas oublier de changer le `prediction_stride`.

5 Deroulé et compte-rendu hebdomadaire de stage

https ://docs.google.com/document/d/1ytEwemE_ yDDJF7bCyrAjE1gXsHqTX2QhppHQYJcpGUk/edit

Lien overleaf pour ce rapport : https ://fr.overleaf.com/4331851673cdnqnhmhdxgy#7e90cc