# Manuel d’utilisation de FloodML

Le présent manuel d’utilisation a pour but de permettre l’installation de l’environnement de travail et de décrire l’utilisation du code FloodML. Ce code permet, en utilisant une approche via apprentissage machine (Machine Learning) de réaliser des inférences de zones inondées à partir de données satellitaires. Ces données peuvent aussi bien être issues de capteurs optiques (Sentinel-2, Landsat 8/9) que SAR (Sentinel-1, TerraSAR-X).

Dans le cadre du projet FloodDAM-DT, l’environnement de travail était le HPC T-Rex du CNES. Les répertoires utilisés sont indiqués dans ce contexte, mais peuvent être modifiés pour utilisation dans un autre cadre.

# Clonage du projet FloodDAM-DT

Avant toute chose, il convient de cloner le projet FloodDAM dans le répertoire de son choix. Ceci se fait à partir de la commande suivante :

git clone -b flooddamdt https://gitlab.cnes.fr/flooddam/wp3-rapid-mapping.git

# Environnement de travail

Le code s’exécute en python 3. L’environnement de travail associé est résumé dans le fichier requirements -R02108.txt a la racine du projet, généré via le gestionnaire de configuration anaconda.

Commande à lancer depuis la racine du projet pour créer l’environnement de travail avec anaconda :

Sur T-rex, il faut au préalable charger conda :

module load conda

conda activate

Pour tout le monde :

conda create --name rapids-0.21.08 --file requirements-R02108.txt

# Fonctionnement du code d'inférence

* 1. **Données satellitaires en entrée :**

L’entrée principale du code est la donnée satellitaire (dans le répertoire *infold*).

* Pour Sentinel-1, les données doivent être pré-traitées (calibration, correction thermique, orthorectification) et tuilées via S1tiling. C’est une étape indispensable pour l’inférence à partir de données Sentinel-1.
* Les données Sentinel-2 doivent être au niveau L2A via SEN2COR.
* Pour les données Landsat 8&9, il faut utiliser les données de niveau 2 distribuées par l’Earth Explorer de l’USGS (<https://earthexplorer.usgs.gov/>).
* Pour TerraSAR-X, ce sont les données au format EEC qui sont prises en comptes.
  1. **Données annexes nécessaires :**

En sus des données satellitaires en entrée, d’autres répertoires contenant des données annexes mais nécessaires à la bonne exécution du code doivent être mis en argument du fichier d’inférence, tels que suit :

* Répertoire de MNT. On utilisera les données tuilées suivant S2 issues de MERIT ou CopernicusDEM GLO-30 dans notre cas, disponibles dans les répertories suivants sur le HPC T-rex:
  + MERIT:

'/work/FLOODML/data/deliveries/floodml/phase-1-cls/MERIT\_S2/'

Attention, les données MERIT ont été au préalable tuilées suivants les tuiles S2, et portent le nom de ces tuiles, par exemple 31TCH.tif

* + CopernicusGLO-30:

' /work/datalake/static\_aux/MNT/Copernicus\_DSM/'

* Répertoire de GSWO (données d'occurrence en eau, voir Pekel et al., 2016), nécessaire pour la production du mini rapport mis en page :

'/work/datalake/static\_aux/MASQUES/PEKEL/2018/occurrence/'

* Répertoire contenant les geotifs de l'ESA World Cover 2021:

‘/work/FLOODML/data/deliveries/flooddamdt/ESA\_worldcover/2021\_v200'

* Le fichier entrainé, qui dépend de la donnée en entrée (S1, S2, L8/9, TSX), et du MNT d’entrainement pour S1 (MERIT ou CopernicusDEM GLO-30). On a ainsi :
  + Pour S1 avec MERIT:

'/work/FLOODML/data/deliveries/flooddam/trained\_models/DB\_S1\_GPU\_R02108-MERIT.sav'

* + Pour S1 avec CopernicusDEM GLO-30:

'/work/FLOODML/data/deliveries/flooddam/trained\_models/DB\_S1\_GPU\_R02108-CDEM.sav'

* + Pour S2 :

'/work/FLOODML/floodml/data/deliveries/flooddam/trained\_models/DB\_S2\_GPU\_R02108.sav'

* + Pour TSX, on utilisera la base issue de la seule polarisation VV de Sentinel-1, et entrainée uniquement avec CopDEM GLO-30 :

‘/work/FLOODML/data/deliveries/flooddam/trained\_models/DB\_S1\_GPU\_R02108-CDEM\_VVseul.sav'

* FloodML doit également créer un dossier temporaire dans un dossier à indiquer (*tmp\_dir*), situé idéalement dans le dossier de sortie. Le code supprime ce dossier temporaire créé à la fin de chaque itération.
  1. **Dossier et fichiers en sortie :**

Le dossier de sortie doit être précisé à l’exécution du code (*outfold*). Un sous-répertoire sera créé par donnée ingérée en entrée, en suivant la nomenclature suivante :

FloodMapping\_TILE\_AAAAMMDDTHHMMSS\_SAT\_ORBIT

Chaque sous-répertoire contiendra alors trois fichiers respectant la nomenclature ci-après :

* + **FM\_TILE\_DATETHHMMSS\_ SAT\_ORBIT\_POST.tif**

Fichier contenant l’inférence simple des zones détectées en eau, auquel est appliqué un filtre de majorité.

Une image contenant carte, Terre, noir et blanc

Description générée automatiquement

Figure 1 - Exemple d’inférence faite à partir de Sentinel-2 et générée sur a tuile 34SEJ pour l'évènement d'inondation du10 septembre 2023 en Grèce

* + **FM\_TILE\_DATETHHMMSS\_ SAT\_ORBIT\_POST.png**

Fichier contenant un mini rapport mis en page concernant la donnée en entrée, en plus de la contextualisation de la zone détectée comme inondée.

Une image contenant texte, capture d’écran, carte

Description générée automatiquement

Figure 2 - Exemple de mini rapport généré sur a tuile 34SEJ pour l'évènement d'inondation du10 septembre 2023 en Grèce

* + **FM\_TILE\_DATETHHMMSS\_ SAT\_ORBIT\_OCS.tif**

Fichier contenant l’inférence des zones vues en eau (inondées), avec de plus l’indication des classes de forêt et de zones urbaines selon la classification de l’ESA world cover 2021. Les classes sont notées suivant la numérotation suivante :

Une image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant carte, vert, art

Description générée automatiquement

Figure 3 - Exemple d’inférence faite à partir de Sentinel-2 et générée sur a tuile 34SEJ pour l'évènement d'inondation du10 septembre 2023 en Grèce, avec en sus l’indication des zones de forêt et des zones urbaines issues de l’ESA world cover 2021.

|  |  |
| --- | --- |
| **Valeur** | **Classe** |
| 0 | Sol |
| 1 | Inondation/eau |
| 2 | Forêt |
| 3 | Forêt détectée comme inondée |
| 4 | Zone urbaine |
| 5 | Zone urbaine vue comme inondée |
| 6 | Nuage (valable pour les données optiques) |
| 7 | Ombre (valable pour les données optiques) |

Avec :

* TILE : numéro de la tuile pour les données S1, S2, L8/9 (par exemple 31TCJ)
* DATETHHMMSS : date et heure de début d’acquisition (par exemple 20210202174756)
* SAT : type de satellite (S1, S2, L8, L9, TSX)
* ORBIT : numéro d’orbite
  1. **Exemple de lancement**

On commence par charger l’environnement de travail créé précédemment par anaconda :

conda activate rapids-0.21.08

En suivant les diverses entrées présentées précédemment, on a alors une configuration de ce type :

medir='/work/FLOODML/data/deliveries/floodml/phase-1-cls/MERIT\_S2/'

gswdir='/work/datalake/static\_aux/MASQUES/PEKEL/2018/occurrence/'

wc\_dir='/work/FLOODML/data/deliveries/flooddamdt/ESA\_worldcover/2021\_v200'

dbpath='/work/FLOODML/floodml/data/deliveries/flooddam/trained\_models/DB\_S1\_GPU\_R02108-MERIT.sav'

infold= '/work/FLOODML/data/deliveries/flooddamdt/Ohio2018/S1Tiled'

outfold='/work/scratch/data/username/TEST\_FLOODML/'

tmp\_dir='/work/scratch/data/ username /TEST\_FLOODML/'

type='s1' # ‘s1’ ou ‘s2’ ou ‘l8’ ou ‘l9’ ou ‘tsx’

Et la commande python de lancement devient alors

python RDF-3-inference.py -i $infold -o $outfold -m $medir -wc $wc\_dir --satellite $type -db $dbpath -g $gswdir -r 2 -tmp $tmp\_dir

A noter que la valeur 2 de « -r 2 » correspond au rayon du filtre de majorité appliqué à l’inférence, pour éliminer les bruits et améliorer la compacité des zones estimées inondées. Pour S1 une valeur de 2 est recommandée, pour S2 plutôt 1.

Les arguments sont détaillés tels que suit dans RDF-3-inference.py :

parser.add\_argument('-i', '--input', help='Input folder', type=str, required=True)

parser.add\_argument('-o', '--Inf\_ouput', help='Output folder', type=str, required=True)

parser.add\_argument('-m', '--meritdir', help='MERIT DEM folder.'

                                                                                                                                                                             'Either this or --copdemdir has to be set for sentinel 1.', type=str, required=False)

parser.add\_argument('-wc', '--wc\_dir', help='ESA world cover directory', type=str, required=True)

parser.add\_argument('--satellite', help='s1, s2, l8, l9 or tsx', type=str, required=True, choices=["s1", "s2", "l8", "l9", "tsx"])

parser.add\_argument('-db', '--db\_path', help='Learning database filepath', type=str, required=True)

parser.add\_argument('-tmp', '--tmp\_dir', help='Temporary folder', type=str, required=False, default="tmp")

parser.add\_argument('-g', '--gsw', help='Tiled GSW folder', type=str, required=True)

parser.add\_argument('-r', '--rad', help='Post-process MAj filter radius', type=int, required=False)

Pour ce qui est des tests, des dossiers prêts à l’utilisation sont disponibles dans les répertoires du projet FLOODML sur T-rex.

Pour S1, le cas de Nooksack (USA/Canada) peut être utilisé :  
 ‘/work/FLOODML/data/deliveries/flooddam/Study\_cases/Nooksack/ S1Tiled’

Et pour le cas S2, le cas d’inondation au Soudan:  
 ‘/work/FLOODML/data/deliveries/flooddam/Study\_cases/Soudan/S2L2A’

Pour ce qui est des jobs pouvant être lancés via slurm (configuration du HPC T-rex du CNES), quatre exemples de fichiers sont proposés dans le dossier Launchers du projet gitlab. Ils permettent de faire fonctionner le code FloodML pour des données S1, S2, L8 et TSX. Ne pas oublier de modifier les dossiers de sorties, ainsi que le chemin des fichiers de sorties et d’erreurs des jobs (en en-tête de ces fichiers). Enfin, le chemin vers le projet FloodML cloné auparavant doit prendre la place de la variable ‘Path2FLDMLproject’.

1. Dockerisation
   1. Contexte

La pré-opérationnalisation du projet FloodDAM-DT a pour but de démontrer qu’à partir d’une détection d’anomalie de hauteur sur un cours d’eau, une zone et un intervalle temporel autour de cette détection permet de récupérer des données spatiales en entrée. Ces données, une fois préparées, sont utilisées pour inférer les zones inondées avec FloodML, pour utilisation futures par les autres partenaires du consortium. Tout ceci se déroulant sur la plateforme mise en place sur le HPC du CNES, l’algorithme FloodML doit ainsi être conteneurisé (via docker dans notre cas) afin de simplifier son utilisation avec une orientation modulaire.

* 1. Dockerfile et image

Le fichier dockerfile est ce qui décrit les étapes de la création d’une image Docker, qui est ensuite montée pour devenir un conteneur Docker pouvant être utilisé. Dans le cas de FloodML, les problèmes les plus importants rencontrés durant cette phase ont été de trouver des images et une succession de commandes permettant de recréer l’environnement de développement python adapté. Dans ce cadre, l’utilisation de bibliothèques adaptées à un calcul sur GPU (via cuda) a dû être combiné à l’utilisation de GDAL et d’autres bibliothèques python pouvant créer des conflits.

En se plaçant dans le répertoire source du projet, on peut alors créer l’image docker *flooddam* à partir du fichier *dockerfile* avec le tag associé à la version de l’image fdamdtv1.1 (FloodML v1.1) en utilisant la commande suivante :

docker build . -t flooddam: fdamdtv1.1

Si l’utilisation de Docker n’est pas native sur T-rex à l’heure de la rédaction de ce manuel, les partenaires de CS ont créé à partir de cette image docker une image singularity, qui est utilisée dans le cadre de la démonstration citée dans la section 3.1. Cette image Docker peut toutefois être utilisée dans n’importe quel système pourvu d’un GPU et avec Docker installé.

|  |
| --- |
| **!!! Attention !!!:** Pour une utilisation à titre personnelle, il est souhaitable que les fichiers créés appartiennent à la personne selon des droits définis. Par défaut, l’image docker créée avec le dockerfile fourni crée les fichiers en tant que root. Si ceci était un pré-requis dans le cadre du projet FloodDAM-DT pour conversion vers singularity dans le cadre d’une exécution sur le HPC T-rex du CNES, il peut en être différemment dans un autre contexte.  Dans ce cas-là, deux précautions sont à prendre   1. Modifier le dockerfile pour que l’image créée puisse générer les sorties suivant l’utilisateur désiré. A ce moment-là, il faut ajouter avant la ligne contenant l’entrypoint les lignes suivantes :   ARG UID=XXXX  ARG GID=YYYY  ENV USER\_NAME=ZZZZ  ENV HOME\_DIR=/home/${USER\_NAME}  RUN groupadd -g ${GID} ${USER\_NAME} && \      useradd -m -d ${HOME\_DIR} -u ${UID} -g ${GID} -s /bin/bash ${USER\_NAME}  USER ${USER\_NAME}    Avec UID correspondant à l’id de l’utilisateur, GID l’id du groupe, et enfin USER\_NAME le propriétaire des futurs fichiers générés.   1. Il est également à noter que le dossier de sortie indiqué en entrée de l’image à exécuter par la suite doit déjà exister. Dans le cas contraire, la génération risque de présenter des problèmes et/ou de ne pas donner les bons droits aux sorties générées. |

* 1. Utilisation de l’image docker

Il faut connecter les répertoires et fichiers nécessaires au bon fonctionnement du code FloodML aux arguments en entrée du conteneur créé précédemment. C’est pourquoi plusieurs volumes doivent être montés contenant les différents répertoires contenant les données nécessaires cités ci-dessus, pour la bonne exécution de FloodML**.** Ainsi, quatre volumes sont à monter, liés aux inputs, outputs, et données annexes disponibles sur le HPC du CNES. A l’exécution du docker monté à partir de l’image créée au préalable, la commande de lancement idoine du docker doit donc être telle que suit, avec les différents arguments listés :

docker run \

--gpus all \

--rm -it \

-v $[infold:/data](http://infold/data" \t "_blank)  \## chemin vers le répertoire en entrée contenant données S1Tiled ou S2L2A

-v $[dir\_out:/OUTPUT](http://dir_out/OUTPUT" \t "_blank) \## chemin répertoire de sortie à préciser (doit impérativement exister au préalable)

-v /work/FLOODML/data/deliveries/:/FDAMDT \## chemin vers répertoire de travail FloodDAMDT

-v /work/datalake/static\_aux/MASQUES/PEKEL/2018/occurrence/:/GSWO \##Répertorie des GSWO

flooddam: fdamdtv1.1 \# Nom de l’image et tag associé lors de la création de l’image

-i /data/ \##Données en entrée

-o /OUTPUT \## répertoire en sortie

-m /FDAMDT/floodml/phase-1-cls/MERIT\_S2/ \## Répertoire des MNT

--satellite s1 \## Type de satellite (s1 ou s2 dans ce cas-ci).

-db /FDAMDT/flooddam/trained\_models/DB\_S1\_GPU\_R02108-MERIT.sav \## Chemin de la base entraînée à utiliser

-g /GSWO \## répertoire contenant les fichiers GSWO

--tmp /OUTPUT/ \ ## répertoire où sera créé le dossier temporaire, et écrasé lors de l’exécution du code

--wc /FDAMDT/flooddamdt/ESA\_worldcover/2021\_v200 \##Répertoire contenant l’ESA Wold Cover 2021

-r 2 ## rayon du filtre de majorité pour filtrage (laissé à 2 par défaut, mais pouvant être changé)

Note : il faudrait prendre en compte les recommandations suivantes :

* Attention à l’utilisation du tag « latest » (Dockerfile ligne 21) : En dev, l’utilisation de latest se justifie mais en livraison est une mauvaise pratique.
* Selon le réseau sur lequel on se trouve, il peut y avoir un filtrage qui fait planter la ligne 3 du Dockerfile. À savoir que si l’environnement de génération possède un proxy ALORS une conf supplémentaire sera peut-être nécessaire, mais cela est propre à la machine, pas à la livraison.

## Transfert vers CS pour image singularity

Le CNES ne pouvant exécuter directement de conteneurs Docker, il a été nécessaire de déposer l’image docker sur un repository leur appartenant. Ceci a été fait via les commandes suivantes après avoir eu accès aux login/mdp idoines :

docker login 643vlk6z.gra7.container-registry.ovh.net -u ‘username’

docker tag flooddam:fdamdtv1.1 643vlk6z.gra7.container-registry.ovh.net/flooddam/flooddam:fdamdtv1.1

docker push 643vlk6z.gra7.container-registry.ovh.net/flooddam/flooddam:fdamdtv1.1

Enfin, pour convertir cette image docker en conteneur Singularity, voici la commande utilisée :

export IMAGE\_NAME=flooddam

export IMAGE\_VERSION=fdamdt\_v1.1

docker pull 643vlk6z.gra7.container-registry.ovh.net/flooddam/${IMAGE\_NAME}:${IMAGE\_VERSION}

docker run \

    -v /var/run/docker.sock:/var/run/docker.sock \

    -v $(pwd):/output \

    --privileged \

    -t --rm \

    singularityware/docker2singularity \

        --name ${IMAGE\_NAME}-${IMAGE\_VERSION}.simg \

        643vlk6z.gra7.container-registry.ovh.net/flooddam/${IMAGE\_NAME}:${IMAGE\_VERSION}