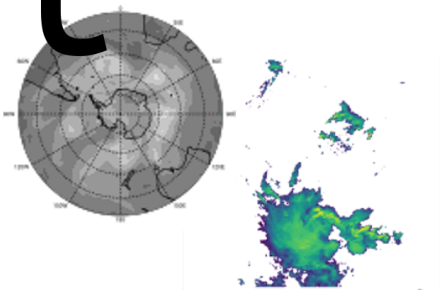
UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

RaiNet



Fait par :
KHADRAOUI M.EL BACHIR
Encadré par :
Pr. LUIZ ANGELO STEFFENEL

Introduction

Le Deep learning est utilisé dans l'ère actuelle afin de faire des prédictions sur des phénomènes particuliers de plusieurs domaines.

Certains modèles prenant des données d'une nature spécifique, il serait possible de les entraîner avec des données ayant une nature proche de celles qui ont servi à leur entraînement initial afin de pouvoir les utiliser, à la prédiction d'autres phénomènes.

C'est précisément le but de ce projet : Utiliser Rainnet, un réseau de Neurones convolutif utilisé pour la prédictions de précipitation, **pour la prédiction des niveaux d'Ozone**.

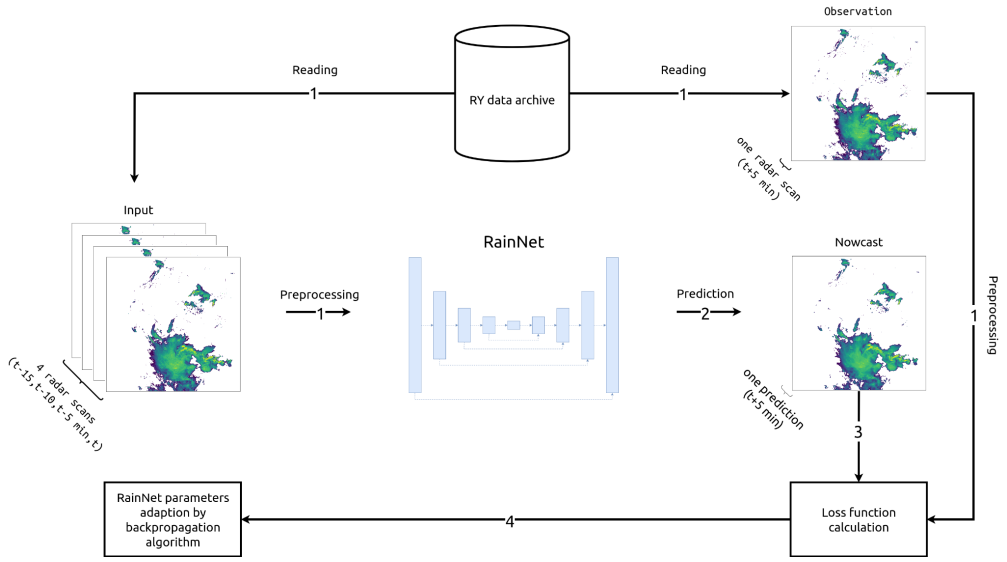
Table des matières

1. Rainnet

Rainnet est un réseau de neurones convolutif entraîné pour la prédiction de précipitations météorologiques basées sur des images radars. Ses prédictions sont de l'ordre de 5 minutes dans le futur. Il a été entraîné avec des images radars fournies par le service de la météo Allemande (DWD)

Rainnet prend en entrée les données images des précipitations aux moments (en minutes) ***t-15, t-10, t-5***, et ***t*** afin de faire une prédiction du moment ***t+5***. La prédiction résultante et les 4 prédictions la précédant sont utilisées récursivement dans le modèle afin de générer des prédictions de l'ordre de 5 minutes dans le futur qui seront réintroduites dans le modèle récursivement jusqu'à l'obtention d'une prédiction de l'ordre d'une heure dans le future (***t+60***).

Le schéma suivant représente une itération des traitement de Rainnet pour générer la prédiction ***t+5*** :



Inspiré par U-Net et SegNet, des models de DeepLearning pour la binay segmentation task, Rainnet utilise une architecture comprenant un encodeur-décodeur dans laquelle l'encodeur réduit la résolution spatiale des données (images) avec du pooling et des convolutional layers et dans laquelle le décodeur augmente la résolution de l'apprentissage grâce à l'upsampling et des convolutional layers

Rainnet est entraîné sur un Dataset riche contenant des images des saisons estivales allant de 2006 à 2013. La répartition des données dans le contexte de l'usage de Rainnet est la suivante :

- Les données de 2006 à 2013 sont utilisées pour l'entraînement.
- Les données de 2014 à 2015 sont utilisées pour la validation
- Les données de 2016 à 2017 sont utilisées pour la vérification des prédictions du modèle.

10 epochs ont été utilisés pour entraîner les 31.4 millions de paramètres de Rainnet.

1.2 Architecture de Rainnet

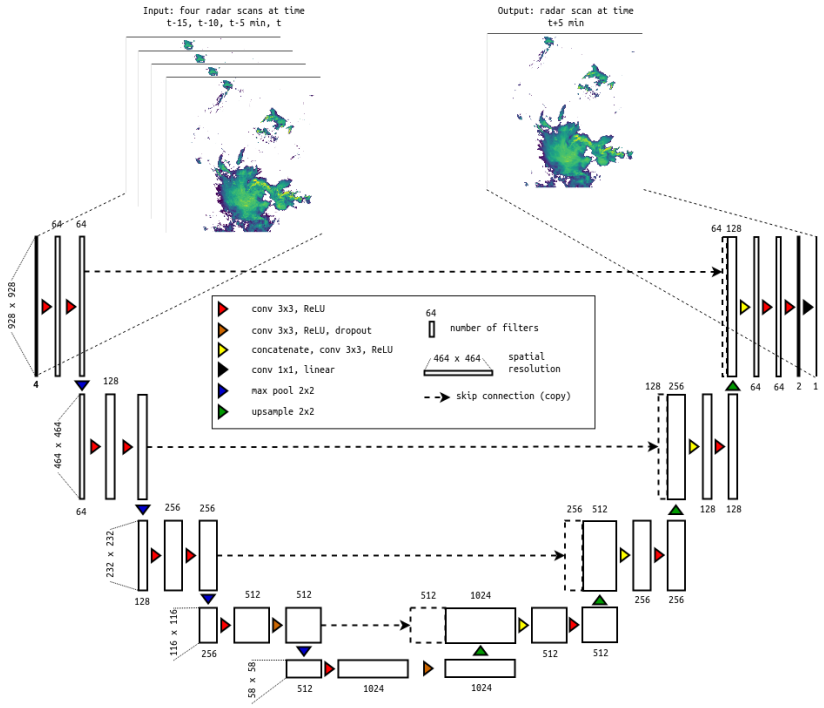
L'architecture de Rainnet est composée des layers suivants :

- 20 convolutional layers
    - 2014 filtres
    - Kernel de taille 1x1 et 3x3
    - Usage de ReLU activation functions
- 4 max pooling
    - Filtre de 2x2
- 4 upsampling
- 2 dropout
- 4 skip connections
    - Utilisées pour éviter des altérations au niveau des Gradients

La résolution des données d'entrées de Rainnet, qui a une hauteur et largeur toujours égales, doit être un multiple de 2^{n+1} avec *n* le nombre de Max pooling Layers du modèle, qui est égale à 4 avec la configuration de base de Rainnet.

Dès lors, les images du Dataset qui ont servi à l'entraînement de Rainnet ont subi un **padding** afin de passer de la résolution 900 à la résolution 928.

La schéma suivant résume l’architecture de Rainnet :



Il est possible d'observer les quatres moments ***t–15, t-10, t-5***, et ***t*** introduits dans le modèle et qui passent par l’intégralité des layers énumérés plus haut afin générer une prédiction du moment ***t+5***.

1.3 Dataset de Rainnet

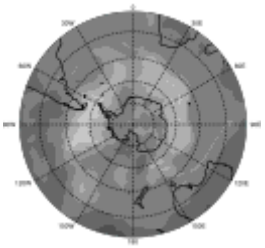
Le Dataset de Rainnet consiste en des images radars de précipitation fournies par le service météorologique Allemand (German Weather Service : DWD). Les données vont de 2006 jusqu’à 2017.

Les images sont stockées dans un Dataset d’une taille de 40 Gigaoctets au format HDF5.

Chaque image est de résolution 900x900 sous forme d’un numpy array qui passera via un padding à une résolution de 928x928 pour l‘entrainement et la prédiction avec Rainnet.

2. Donnés d'Ozone

Les données d'Ozone fournies s'étalent sur une période de 10 ans, de 2010 à 2019 et dont les images ont des intervalles de 6 heures entre elles.



Les images sont des résolution 128x128.

2.1 Adaptation de Rainnet aux données d'Ozone

En sachant que l'un des pré-requis de Rainnet est le fait que la résolution des images d'entrées doit être un multiple de 2^{n+1} avec *n* le nombre de Max pooling Layers du modèle (4 dans ce cas-ci), il n'est donc nécessaire de faire passer les données d'Ozone par l'étape de pré (de 900 à 928 pour les données de base) et post (de 928 à 900 pour les donnés de base) processing avant d'être utilisées pour l'entraînement.

Les images doivent toutefois être représentées sous forme de Dataset au format HDF5 et les clés des images doivent être fournies au format texte.

3.1 Tests avec Rainnet

Afin de se familiariser avec Rainnet, une expérimentation avec le modèle a été faite.

Les tests avec Rainnet ont été effectués sur la plateforme Colab de Google avec l'utilisation de GPU.

Pour pouvoir procéder au tests, il est nécessaire de disposer du modèle disponible à cette adresse. Afin que les tests se fassent de manière efficace, le modèle à été directement téléchargé dans un dossier Drive. Pour ce faire, il a d'abord été nécessaire de connecter le notebook au sein duquel le traitement s'effectue au Drive utilisateur via les instructions suivantes :

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Une fois l'espace de travail (Notebook) connecté au répertoire de travail, le téléchargement du modèle peut s'effectuer via ce qui suit :

```
import requests
#Adresse URL de téléchargement (directe) du modèle de Rainnet
file_url="https://zenodo.org/record/3630429/files/rainnet.h5?download=1"

r = requests.get(file_url, stream = True)

#Ecriture du modele dans le dossier Drive
with open("/content/gdrive/My Drive/rainnet.h5", "wb") as file:
    for block in r.iter_content(chunk_size = 1024):
        if block:
            file.write(block)
```

L'utilisation d'un GPU a été vérifiée avec les commandes suivantes :

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
from keras.models import load_model
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

Si un GPU n'est pas attribué, l'erreur suivante survient :

```
---------------------------------------------------------------------------
SystemError                               Traceback (most recent call last)
<ipython-input-1-e913886d5819> in <module>()
      3 from keras.models import load_model
      4 if device_name != '/device:GPU:0':
----> 5   raise SystemError('GPU device not found')
      6 print('Found GPU at: {}'.format(device_name))

SystemError: GPU device not found
```

Le chargement du modèle disponible sur le drive ce fait comme suit :

```
model = load_model("/content/gdrive/My Drive/rainnet.h5")
model.summary()
```

model.summary() permet l'affichage d'informations sur le model, notamment les layers le composant (Citées plus haut) et le nombre de paramètres entraînables qui sont au nombre de 31 380 613.

Une fois le modèle chargé, vient la préparation des données pour la prédiction.
Comme cité précédemment, Rainnet prend en entrée les moments ***t-15, t-10, t-5, t*** consécutivement afin de générer la prédiction du moment ***t+5*** (Qui elle-même sera utilisée par la suite afin de générer des prédictions récursivement jusqu'au moment ***t+60***). Dès lors, il faut représenter les moments ***t-15, t-10, t-5, t*** sous forme d'array à introduire au modèle. Les données de ces moments, représentant des images radars, sont contenues à l'intérieur du Dataset ayant servi à l'entraînement de Rainnet. Un moment aléatoire hors des données périodes utilisées pour l'entraînement (2012-2016) et la validation (2017) a été choisi : le 2009/03/09 **09:50.** Il représente le moment ***t***

Les 4 données (images) à extraires sont donc les suivantes :

t-15	***t-10***	***t-5***	***t***
2009/03/09 **09:35**	2009/03/09 **09:40**	2009/03/09 **09:45**	2009/03/09 **09:50**

Les données des images sont représentées sous forme de numpy array de taille 900x900. Elles sont extraites puis écrites dans des fichiers textes afin de permettre leur utilisation directe dans le notebook Colab. Le script suivant lit le fichier hdf5 du dataset, extrait les données correspondantes et les écrit dans des fichiers.

```
import h5py
import numpy as np

dataset_dict= h5py.File("RYDL.hdf5", "r")

data_t_15 = dataset_dict['200903090935'][:]
np.savetxt("text.txt",data)
data_t_10 = dataset_dict['200903090940'][:]
np.savetxt("text2.txt",data2)
data_t_5 = dataset_dict['200903090945'][:]
np.savetxt("text3.txt", data3)
data_t_0 = dataset_dict['200903090950'][:]
np.savetxt("text4.txt", data4)

dataset_dict.close()
```

Après la génération des fichiers des données vient leur lecture au niveau du Notebook.

```
data = np.loadtxt('text.txt')
data2 = np.loadtxt('text2.txt')
data3 = np.loadtxt('text3.txt')
data4 = np.loadtxt('text4.txt')
```

Le modèle ne prend en entrée que des images de résolution de 928x928 alors que les images du Dataset sont de 900x900. Il est toutefois possible de constater au niveau du Notebook d'entraînement de Rainnet que la phase de prétraitement des données consiste en l'application d'un padding afin de passer d'une résolution de 900x900 à 928x928.

Dès lors, il faut appliquer un padding identique aux données chargées des fichiers.

```
#Pad de (14,14) pour obtenir des images passant de 900 à 928
data = np.pad(data, (14,14),'constant', constant_values=(0))
data2 = np.pad(data2, (14,14),'constant', constant_values=(0))
data3 = np.pad(data3, (14,14),'constant', constant_values=(0))
data4 = np.pad(data4, (14,14),'constant', constant_values=(0))
```

Les données prêtes, elles doivent être mises dans un array de dimension 4 afin d'être fournies au modèle en entrée.

```
full = np.zeros((4,1,928,928))

full[0] = data #2009/03/09 09:35
full[1] = data2 #2009/03/09 09:40
full[2] = data3 #2009/03/09 09:45
full[3] = data4 #2009/03/09 09:50

full = full.reshape(1,928,928,4)
```

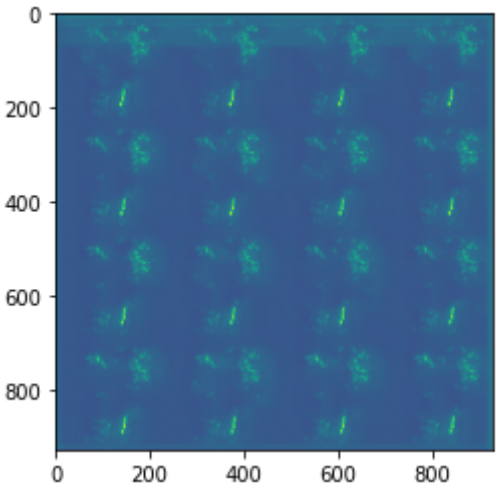
Le vecteur doit respecter le format d'entrée du modèle qu'est *(1,928,928,4)*
La prédiction peut dès lors d'effectuer

```
prediction = model.predict(full)
```

3.2 Analyses des résultats de Rainnet

La prédiction en sortie de Rainnet est une image composée de quatre images (Qui se répètent) qui désigne les quatres prochaines estimation des précipitations correspondantes au moment ***t+5, t+10, t+15, t+20***

Le format de la prédiction en sortie est : *(1,928,928,1)***.** Après application d'un reshape l'on peut observer l'image suivante qui contient les quatre futures prévisions citées plus haut.



La première ligne composée de quatre images se répète sur les 3 lignes qui la succèdent. Les quatres images correspondent donc aux moments suivants :

t	***t+5***	***t+10***	***t+15***
2009/03/09 **09:55**	2009/03/09 **10:05**	2009/03/09 **10:10**	2009/03/09 **10:15**

Pour pouvoir évaluer le modèle avec la fonction *model.evaluate(X,Y)* et valider les prédictions obtenues, il est nécessaire d'avoir les données réelles des moments qui ont été prédis. Pour ce faire, la même procédure que celle utilisée pour les données d'entrée du modèle est appliquée, à savoir, récupérer les données à partir du Dataset au format hdf5 et les écrire dans des fichiers. Le script suivant permet de le faire.

```
import h5py
import numpy as np

dataset_dict= h5py.File("RYDL.hdf5", "r")

data_t_p_5 = dataset_dict['200903090955'][:]
np.savetxt("text5.txt",data_t_p_5)
data_t_p_10 = dataset_dict['200903091000'][:]
np.savetxt("text6.txt",data_t_p_10)
data_t_p_15 = dataset_dict['200903091005'][:]
np.savetxt("text7.txt", data_t_p_15)
data_t_p_20 = dataset_dict['200903091010'][:]
np.savetxt("text8.txt", data_t_p_20)

dataset_dict.close()
```

Les données passent par le même traitement que précédemment, à savoir le padding, la mise en place dans un array de dimension 4 et un reshape pour convenir avec le format d'entrée de la fonction *model.evaluate(X,Y).*

```
full2 = np.zeros((4,1,928,928))

data = np.loadtxt('text5.txt')
data2 = np.loadtxt('text6.txt')
data3 = np.loadtxt('text7.txt')
data4 = np.loadtxt('text8.txt')

#Pad de (14,14) pour obtenir des images passant de 900 à 928
data = np.pad(data, (14,14),'constant', constant_values=(0))
data2 = np.pad(data2, (14,14),'constant', constant_values=(0))
data3 = np.pad(data3, (14,14),'constant', constant_values=(0))
data4 = np.pad(data4, (14,14),'constant', constant_values=(0))

full2[0] = data
full2[1] = data2
full2[2] = data3
full2[3] = data4

full2 = full2.reshape(1,928,928,4)
```

```
model.evaluate(full, full2)
```

```
1/1 [==============================] - 1s 1s/step - loss: 7.7024e-04
0.0007702427101321518
```

Après évaluation, l'on peut constater que la perte est infime est que les prédictions du modèle sont **particulièrement précises.**

4.1 Entraînement de Rainnet avec les données d'Ozone :

L'entraînement de Rainnet avec les données D'Ozone s'est fait sur Colab avec l'utilisation de GPU.
Le Notebook fourni par les développeurs de Rainnet a été repris afin d'effectuer l'entraînement avec les données d'Ozone. Tout comme pour le test effectué avec le modèle de Rainnet, le Notebook de l'entraînement est connecté avec un Drive contenant les données nécessaires à l'opération, à savoir :

- Le Dataset hdf5 des images d'Ozone
- Les clés d'images

Les images radars d'Ozone sont organisées en 10 dossiers chacun contenant des images d'une année allant de 2010 à 2019. Il a fallu inspecter le type et format des données du Dataset de base de Rainnet (Météo allemande) afin d'en reproduire les caractéristiques sur les données d'Ozone.

```
('200801131340', <HDF5 dataset "200801131340": shape (900, 900), type "<f2">)
('200801131345', <HDF5 dataset "200801131345": shape (900, 900), type "<f2">)
('200801131350', <HDF5 dataset "200801131350": shape (900, 900), type "<f2">)
('200801131355', <HDF5 dataset "200801131355": shape (900, 900), type "<f2">)
('200801131400', <HDF5 dataset "200801131400": shape (900, 900), type "<f2">)
('200801131405', <HDF5 dataset "200801131405": shape (900, 900), type "<f2">)
('200801131410', <HDF5 dataset "200801131410": shape (900, 900), type "<f2">)
```

Pour produire le Dataset des données Ozone, les images ont été lues, converties en grayscale grâce à OpenCV et insérées dans le Dataset. Le script suivant à permis de réaliser cela.

```python
import numpy as np
import h5py
import cv2
import os
import re

hf = h5py.File('Ozone.hdf5', 'w')


rootdir = 'Data/'
for file in os.listdir(rootdir):
    d = os.path.join(rootdir, file)
    if os.path.isdir(d):
        for image in os.listdir(d):
            numbers = re.findall(r"\d+", image)
            numbers.__delitem__(0)

            im = cv2.imread(d+"/"+image)

            img_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
            dataset = np.array(img_gray.astype(np.float16))
            hf.create_dataset(numbers[0], data=dataset)


hf.close()
```

Les clés des images sont obtenues en parcourant chaque image et en écrivant son nom dans une liste qui sera par la suite mise en fichier.

```
import re
import os
import numpy as np

i=0
keys=[]
#array = np.zeros()
rootdir = 'Data/'
for file in os.listdir(rootdir):
    d = os.path.join(rootdir, file)
    if os.path.isdir(d):
        for image in os.listdir(d):
            numbers = re.findall(r"\d+", image)
            numbers.__delitem__(0)
            keys.append(numbers[0])

with open("Ozone_keys.txt", "w") as output:
    output.write(str(keys))
```

La chargement du Dataset d'Ozone s'effectue avec les commandes suivantes :

```
import h5py
dataset_dict = h5py.File('drive/MyDrive/OzoneData/Ozone.hdf5', 'r')
```

Les images entre 2010 et 2017 seront utilisées pour l'entraînement, celles de 2018 pour la validation, et celles de 2019 pour la vérification. Cette répartition se résume dans ce qui suit.

```
import ast
with open('drive/MyDrive/OzoneData/Ozone_keys.txt','r') as f:
  image_names = ast.literal_eval(f.read())
image_names = [name for name in image_names if name[:4]>'2010']

train_images = [name for name in tqdm(image_names) if "2018" not in name and "2019" not in name] #2010-2017
val_images = [name for name in tqdm(image_names) if name[0:4]=="2018"] #2018-2019
```

La résolution des images du Dataset pour l'entraînement selon les développeurs de Rainnet doit être un multiple de 2^{n+1} avec *n* le nombre de Max pooling Layers du modèle. En sachant que n est égale à 4 et que la résolution des images est de 128x128 et que 128 est un multiple de 2^{4+1} (32), d'où la résolution des images du Dataset des précipitations qui doivent être de résolution 928x928 avant d'être utilisées en entrée dans le modèle de Rainnet. Dans le cas des images d'Ozone, il est possible de passer outre l'étape de padding qui consiste en le fait de rajouter des éléments (0) dans les arrays d'images afin qu'ils aient des dimensions conformes à la condition posée. Les fonctions du Notebook ont été modifiées afin que les données ne subissent pas de changements.

```
def pad_to_shape(array, from_shape=128, to_shape=128, how="mirror")
def pred_to_rad(pred, from_shape=128, to_shape=128)
```

Le type de layers et leur nombre qui caractérisent la structure de Rainnet n'ont été modifiés.

Le nombre d'epochs utilisés pour l'entraînement est de 2. Au delà de 2 (Notamment 3), il a été constaté une saturation du modèle (Over-fitting) et donc l'obtention de prédictions ayant un taux d'erreur particulièrement élevé qui se traduisait en image entièrement noir.

Les statistiques de l'entraînement avec 2 epochs est le suivant :

```
Epoch 1/2
11684/11684 [==============================] - 1456s 123ms/step - loss: 0.0098 - val_loss: 0.0011
Epoch 2/2
11684/11684 [==============================] - 1437s 123ms/step - loss: 0.0018 - val_loss: 0.0013
<keras.callbacks.History at 0x7f9ebf23ded0>
```

La durée totale de l'entraînement était de ~48 minutes (~24 minutes par epoch)

4.2 Analyse des résultats de l'entraînement de Rainnet

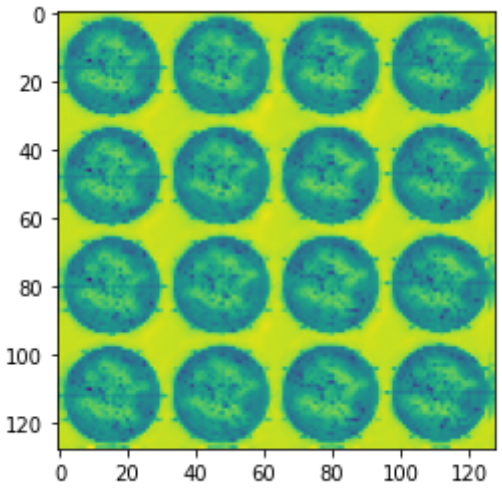
Comme pour l'étape de la prédiction avec le modèle de Rainnet, il a fallu préparer les images correspondantes aux moments ***t-18, t-12, t-6, t*** afin de faire une prédiction.

t-18	***t-12***	***t-6***	***t***
2019/11/30 **00:00**	2019/11/30 **06:00**	2019/11/30 **12:00**	2019/11/30 **18:00**

```
#POUR PREDICTION
data = np.loadtxt('text.txt') #t-18 : 2019113000
data2 = np.loadtxt('text2.txt') #t-12 : 2019113006
data3 = np.loadtxt('text3.txt') #t-6 : 2019113012
data4 = np.loadtxt('text4.txt') #t : 2019113018
#POUR EVALUATION
data5 = np.loadtxt('text5.txt') #t+6 : 2019120100
data6 = np.loadtxt('text6.txt') #t+12 : 2019120106
data7 = np.loadtxt('text7.txt') #t+18 : 2019120112
data8 = np.loadtxt('text8.txt') #t+24 : 2019120118
```

```
full = np.zeros((4,1,128,128))
full[0] = data #t-18 : 2019113000
full[1] = data2 #t-12 : 2019113006
full[2] = data3 #t-6 : 2019113012
full[3] = data4 #t : 2019113018
full = full.reshape(1,128,128,4)
prediction = model.predict(full) #Result contains prediction for t+6 t+12 t+18 t+24
prediction_reshaped = prediction.reshape(128,128)
plt.imshow(prediction_reshaped)
plt.show()
```

Voici le résultat obtenu.



Comme dans le cas de l'analyse des résultats obtenus avec la modèle de Rainnet avec le dataset fourni de base, la première ligne contient les quatres images correspondant à la concentration d'Ozone aux moments ***t+6, t+12, t+16, t+18***.

Pour l'évaluation, les données des moments t+6, t+12, t+16, t+18 ont été mis dans des fichiers qui ont été chargés dans le notebook pour l'évaluation du modèle.

t+6	***t+12***	***t+18***	***t+24***
2019/12/01 **00:00**	2019/12/01 **06:00**	2019/11/13 **12:00**	2019/11/13 **18:00**

```
full2 = np.zeros((4,1,128,128))
full2[0] = data5 #t+6 : 2019120100
full2[1] = data6 #t+12 : 2019120106
full2[2] = data7 #t+16 : 2019120112
full2[3] = data8 #t+18 : 2019120118
```

```
full2 = full2.reshape(1,128,128,4)
model.evaluate(full, full2)
```

```
1/1 [==============================] - 0s 327ms/step - loss: 15.6586
15.65856647491455
```

Suite à l'évaluation, il est possible de constater que l'indicateur loss est assez élevé. Néanmoins, Il n'en reste pas moins que le modèle ait permis d'obtenir des images de prédiction très proches à celles d'origine.

Conclusion

Rainnet s'est montré être efficace afin d'obtenir des prédictions de la concentration d'Ozone jusqu'à 1 jour dans le futur. Davantage d'expérimentations et d'ajustements permettrait d'obtenir des résultats dont la perte serait drastiquement moindre et qui seraient sans nul doute particulièrement précis vu l'architecture de Rainnet qui est adaptée aux traitement des images de concentration radars.