KICT AI based RAIN prediction model User's manual

윤성심

한국건설기술연구원 수자원하천연구본부

v.KICT-RAIN-AI2023.11
2023.11

KICT 한국건설기술연구원
KOREA INSTITUTE of CIVIL ENGINEERING and BUILDING TECHNOLOGY

목 차

1. KICT-RAIN-AI 모형의 개요

1.1 KICT-RAIN-AI_v1

격자 강우예측에 사용되는 기본 딥러닝 모델 구조는 U-Net 구조를 갖는 합성곱 심층신경망을 이용한 예측모델인 RainNet(Ayzel et al., 2020)이다. RainNet에 사용된 신경망 구조는 branch간의 스킵-연결(skip connection)을 포함한 인코더(encoder)-디코더(decoder) 구조를 갖는 U-Net 및 SegNet을 기반으로 한다. RainNet은 인코더가 풀링을 사용하여 공간 해상도를 점진적으로 축소시킨 후 컨볼루션 레이어가 뒤따르는 인코더-디코더 아키텍처를 갖는다. 디코더는 업 샘플링을 사용하여 학습된 영상의 패턴을 높은 공간 해상도로 점진적으로 업 스케일링 한 후 컨볼루션 레이어를 사용한다. 레이어 간의 특징들 사이의 의미적 연결을 보장하기 위해 그래디언트 소실문제를 피하기 위해 제안된 인코더에서 디코더로의 스킵 연결을 포함하고 있다

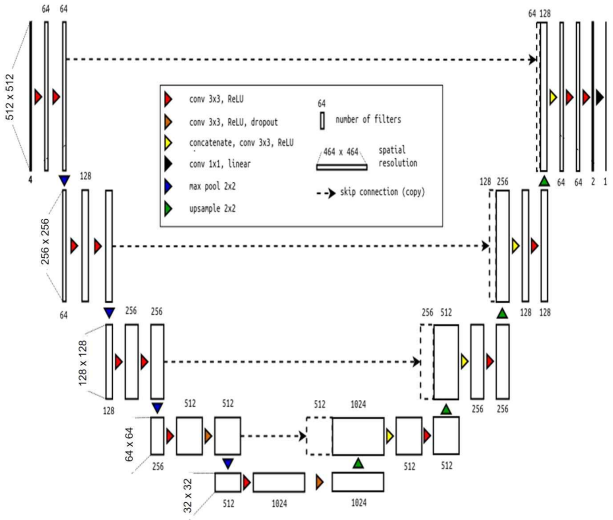


Fig. 1. RainNet architecture (Ayzel et al., 2020)

KICT-RAIN-AI에서는 기존 국내 RainNet의 연구를 참조하여 최대 512의 합성곱 필터를 사용하였고, 1×1 및 3×3의 커널 크기, 합성곱층에는 ReLU (Rectified Linear unit) 활성화 함수를 사용하였다(Yoon et al., 2020)

RainNet 모델은 예측 모의 시점(T)를 기준으로 10분 간격으로 과거 30분 전까지 관측된 레이더 격자 강우자료(Observation T-3, T-2, T-1, T) 4개를 입력으로 받아 다음 10분 후의 예측(Predict T+1)을 수행하고, 관측된 레이더 격자 강우자료(Observation T+1)과 비교하여 오차가 최소가 되도록 학습을 수행한다. 따라서 사전학습된 모델은 10분 예측에 최적화 된 것이다. 이에 오차가 선행시간이 길어질수록 누적되고, 평활화 영향이 크게 나타났다.

KICT-RAIN-AI_v1 모델에서는 기존의 10분 예측으로 사전학습된 모델을 이용하여 재귀적으로 반복 예측하는 과정에서 발생하는 오차 누적 및 평활화 효과를 완화시킬 수 있도록 U-Net 네트워크를 그림 2와 같이 재귀적 예측 전략으로 구성하였다. 예측의 구체적인 과정

은 다음과 같다. 우선 예측 모의 시점(T)를 기준으로 과거 30분 전까지 관측된 10분 간격의 레이더 격자 강우자료(Observation T-3, T-2, T-1, T) 4개를 입력으로 사용한다. 입력자료를 r 전술한 RainNet 모델 구조에 적용하여 10분 예측 레이더 강우자료(Output1)를 생성한다. 다음 예측 시점을 다음 10분 후로 변경하여 생성된 Output1을 관측 강우자료(Observation T-2, T-1, T)와 합쳐서(concatenate) 다음 예측을 위한 입력으로 사용한다. 이 러한 과정을 반복하여 선행 10분에서 부터 180분까지의 예측 강우를 얻을 수 있다. 마지막으로 각 시간별 예측결과(Output1, Ouput2, … , Output18)를 연결하여 관측된 레이더 격자강우(Observation T+1~T+18)와의 오차를 계산하여, 오차가 최소가 되도록 학습을 수행하도록 한다. 따라서 사전학습된 모델은 10분부터 180분 예측 모두에 대해 최적화된 결과가 된다.(Yoon, 2022)

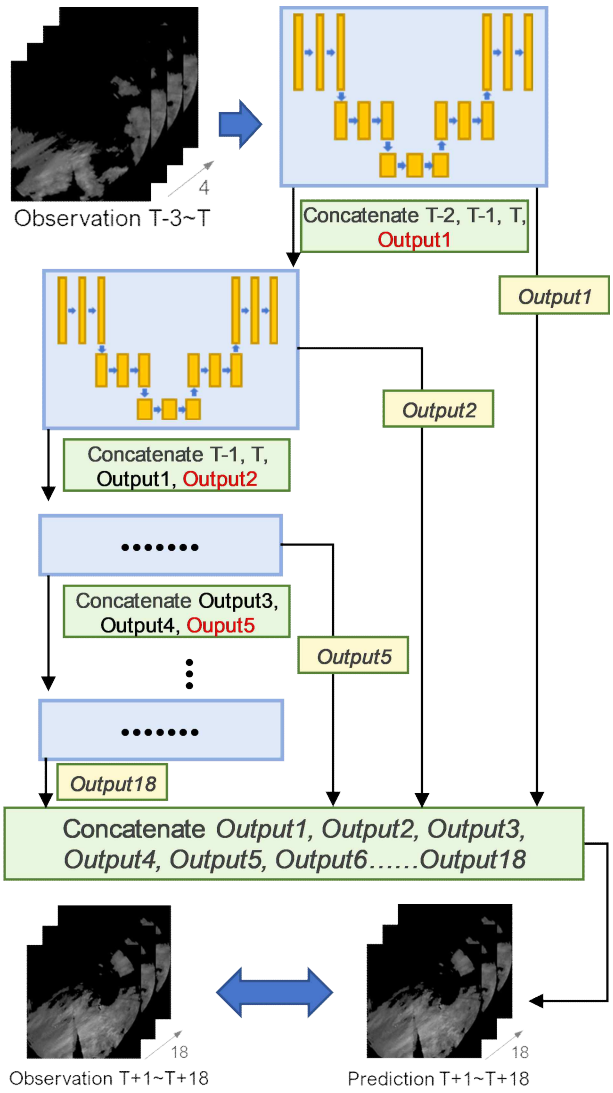


Fig. 2. Recursive RainNet(KICT-RAIN-AI_v1) architecture

1.2 KICT-RAIN-AI_v2

KICT-RAIN-AI_v1의 경우 선행 1시간까지는 정확도가 개선되는 것을 확인하였으나, 1시간 이후의 예측에서는 예측정확도가 급격히 저하는 되었다. 그림 3과 같이 정확도 개선을 위해 각 선행시간별 최적화된 18개의 딥러닝 모델을 구성하였다. 기본 딥러닝 모델 구조는 동일한 RainNet 구조를 사용하였다.

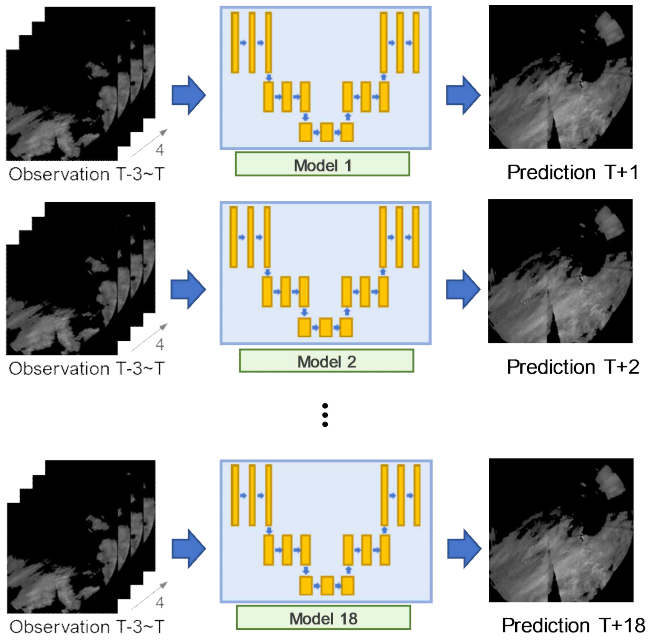


Fig. 3. KICT-RAIN-AI_v2 process

KICT-RAIN-AI 모형을 이용한 강우예측은 사전학습된 모형을 사용하며, 전체적인 예측강우 절차는 github를 통해 제공하는 주피터노트(Inference_KICT_RAIN_AI_525x625_github.ipynb)로 실행된다.

2. 모형 최적화 결과

본 연구에서는 그림 3과 같이 환경부에서 운영하고 있는 대형 S-band 강우레이더로 관측된 2차원 격자 강우자료를 이용하여 딥러닝 기반의 강우예측 모델을 학습하고, 평가하였다. 환경부에서는 관측 반경 125km 이내에서 지표에 근접하게 내리는 비의 양을 집중 관측하여 홍수예보에 활용하기 위해 비슬산(BSL), 소백산(SBS), 모후산(MHS), 서대산(SDS), 가리산(GRS), 예봉산(YBS), 감악산(GAS) 레이더를 운영 중에 있다. 본 연구에서는 2022년 6월 부터 운영을 시작한 감악산 레이더를 제외한 6개의 레이더 자료를 이용하여 전국 합성한 격자 레이더 강우자료(quantitative precipitation estimates)를 이용하였으며, 격자 범위는 그림 3의 파란색 사각형 영역이다. 자료 기간은 2018년부터 2021년 중 강우가 발생한 호우사례 274일을 선정하였다. 또한, 무강우 학습을 수행하지 않기 위해 매 10분단위로 강우면적을 계산하여 0mm 초과하는 강우면적이 1% 이상인 학습용 강우레이더 자료 구축 시점 21,495개를 선정하였다. 자료의 시간 해상도는 10분 간격이며, 공간 해상도는 1 km로 격자 배열은 525×625이다. 강우자료의 단위는 mm/hr이다. 구축된 자료 중 모델의 학습에 사용된 데이터셋은 17,200개이고, 검정에 사용된 데이터셋은 2,149개 이다. 나머지는 2,147개 데이터셋은 평가를 위해 제외하였다.

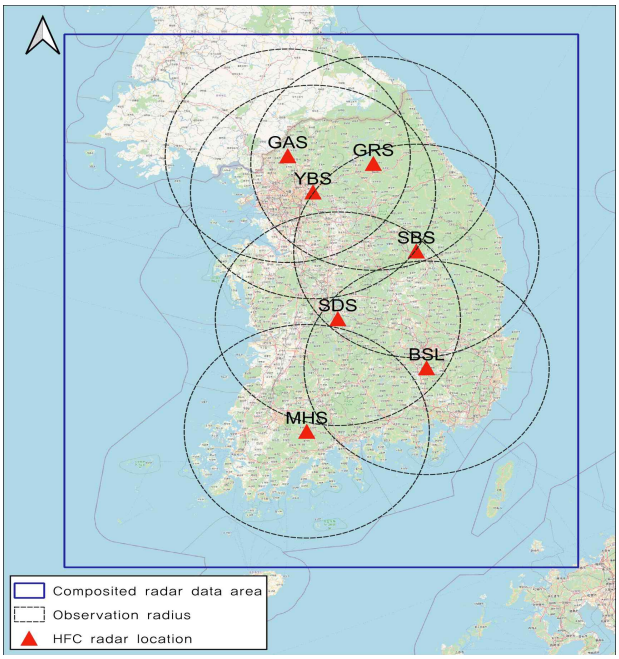


Fig. 3. 환경부 S-band 강우레이더 관측망 현황

각 모형의 최적화를 위해 손실함수(loss function)는 다음 식 (1)과 같이 평균절대오차(MAE, Mean Absolute Error)를 사용하였다.

$$MAE = \frac{\sum_{i=1}^{n} |now_i - obs_i|}{n} \quad (1)$$

여기서, now_i와 obs_i는 i 위치에 예측 및 관측강우강도(mm/hr)이며, n은 레이더 격자 수이다. 학습 및 검정에 대한 측정함수(metric)로는 임계성공지수(Critical Success Index, CSI)와 0.1 mm/hr 이상의 값에 대한 절대평균오차를 사용하였다. 임계성공지수는 표 1과 같이 강수분할표(rain contingency table)에 기초하여 식 (2)로 산출되는 지수로, 정확히 예측했던 사건수를 예측 시 사건 발생과 관련된 총 수를 합하여 나눈 것이다.

$$CSI = \frac{H}{H+M+F} \quad (2)$$

손실함수를 최소화하면서 딥러닝 모델의 매개변수를 갱신하기 위해 Nadam(Nesterov-accelerated Adaptive Moment Estimation)를 사용하였으며, Nadam optimizer의 학습률은 0.0001, beta_1=0.9, beta_2=0.999를 사용하였다[.

본 연구의 딥러닝 기반 강우 예측모델은 케라스(Keras) 프레임워크를 사용하였으며, 듀얼 GPU(NVIDIA RTX A6000)에서 학습을 수행하였다. 각 예측모델(RainNet, Recursive RainNet)의 학습 결과는 그림 4와 표 2에 제시하였다.

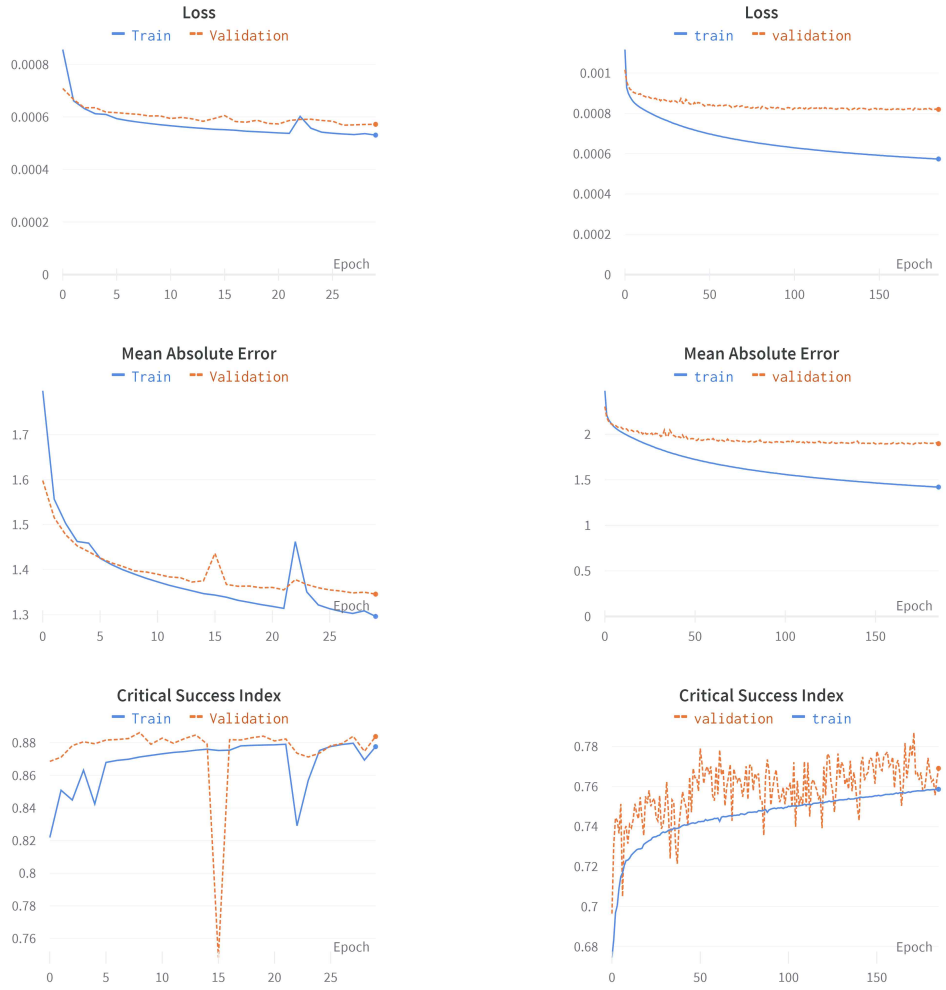
RainNet 모델 학습은 epoch 200 및 배치크기 32로 수행되었으며, 26 epoch에서 손실함수와 측정함수가 최소화되었다. KICT-RAIN-AI_v1 모델 학습은 epoch 200 및 배치크기 8로 수행되었으며, 133 epoch에서 최소화되었다.

Table 1. Rain contingency table(Choi et al, 2005)

Verifying analysis		Forecast rainfall	
		no rain	rain
Observed rainfall	no rain	Z (zero)	F (false)
	rain	M (miss)	H (hit)

Table 2. Results of the deep learning model training

Model	Training Subset			Validation Subset		
	Loss	MAE	CSI	Loss	MAE	CSI
RainNet	0.0005	1.31	0.88	0.0006	1.35	0.88
KICT-RAIN-AI_v1	0.0006	1.49	0.75	0.0008	1.90	0.76



RainNet KICT-RAIN-AI_v1

Fig. 4. Model training result of each epoch (Left:RainNet, Right: KICT-RAIN-AI_v1)

3. 입력자료

KICT-RAIN-AI 모형의 입력자료는 그림 5와 같이 ASCII 포맷의 래스터 데이터를 입력자료로 이용한다.

격자크기는 525x625 이며, 공간해상도는 1km 이다. 입력 값은 mm/hr 단위의 강우강도이며, 예측 실행을 위해 예측하고자 하는 시점을 기준으로 과거 30분부터 총 4개(T-3~T)의 입력자료가 있어야 한다. 현재 좌표계는 EPSG 5186기준이며, 사용자에 따라 좌표계는 변경 사용 가능하다.

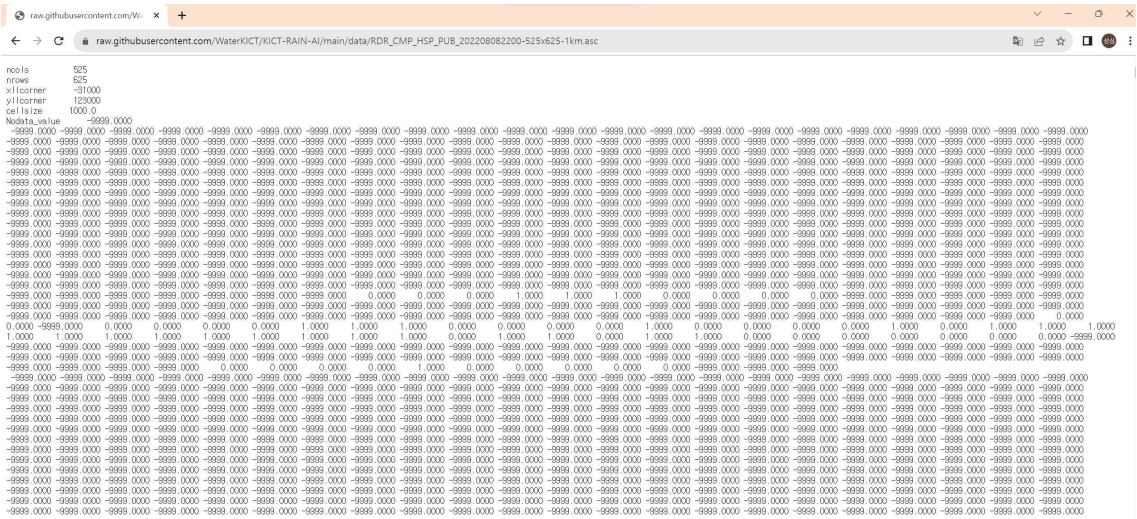


Fig. 4. 입력자료 형식

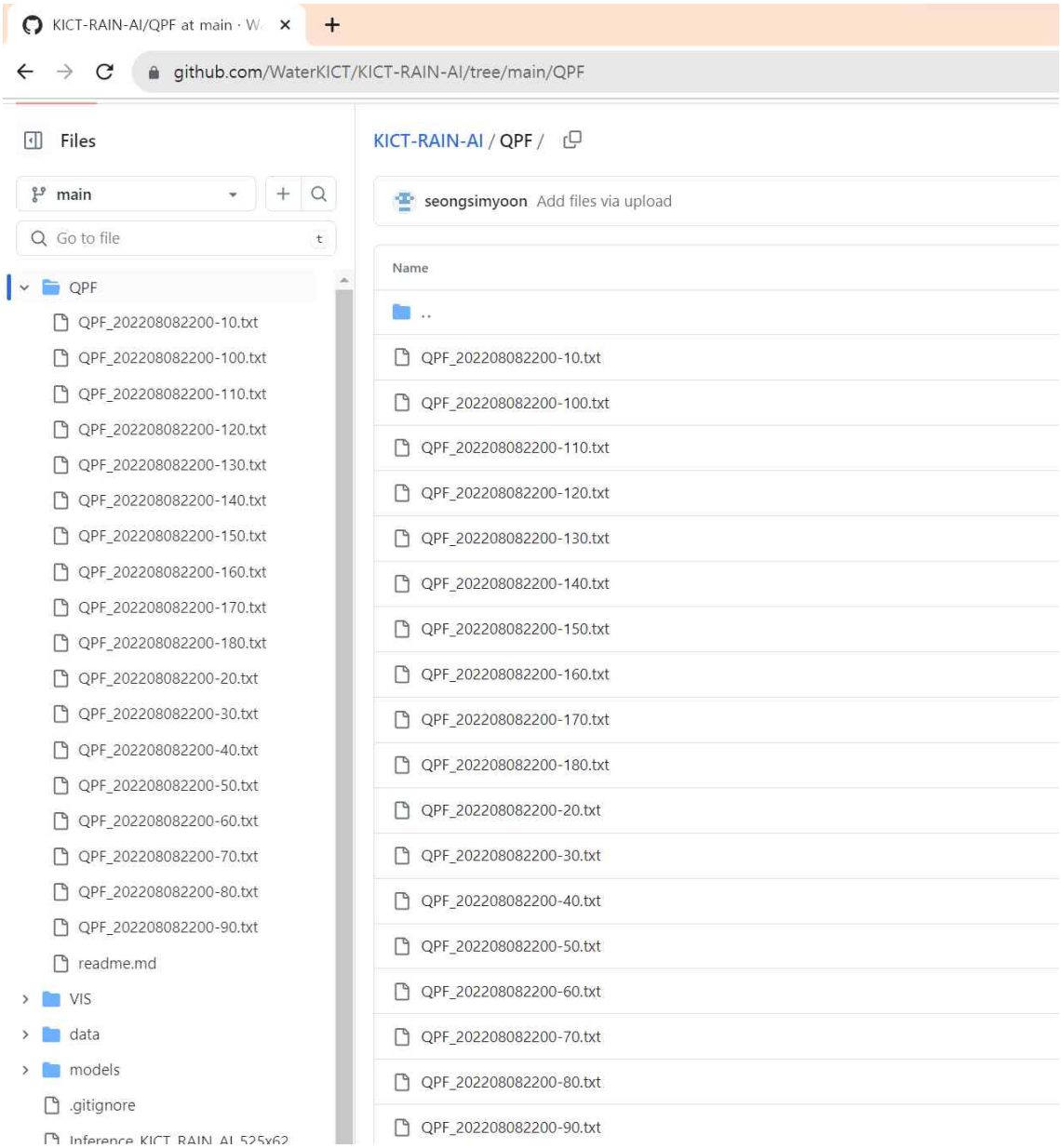
실행 테스트를 위해 2022년 8월 8일부터 10일까지의 10분 간격의 레이더 강우강도 자료를 샘플 자료로 제공한다.

샘플자료는 아래 구글 드라이브 링크를 통해 다운받을 수 있다.

https://drive.google.com/file/d/1n-gi8MI1lTpmF17Mr-epPvtIlbzuno_a/view?usp=drive_link

4. 출력자료

KICT-RAIN-AI 모형의 출력 자료는 선행 10분부터 180분까지 10분 간격으로 18개의 예측강우가 생성된다. 출력 형식은 입력과 동일한 ASCII 포맷의 래스터 데이터이다.



참 고 문 헌

S. S. Yoon, H. S. Park and H. J. Shin, "Very short-term rainfall prediction based on radar image learning using deep neural network," *Journal of Korea Water Resources Association,* Vol. 53, No. 12, pp. 1159-1172, December 2020. doi:10.3741/JKWRA.2020.53.12.1159

S. S. Yoon, "Recursive model based on U-net for very short range forecast." *J. Digit. Contents Soc.* 2022, 23, 2481-2488. https://doi.org/10.9728/dcs.2022.23.12.2481.

G. Ayzel, T. Scheffer, and M. Heistermann, "RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting," *Geoscientific Model Development*, Vol. 13, No. 6, pp. 2631-2644, June 2020.

부 록

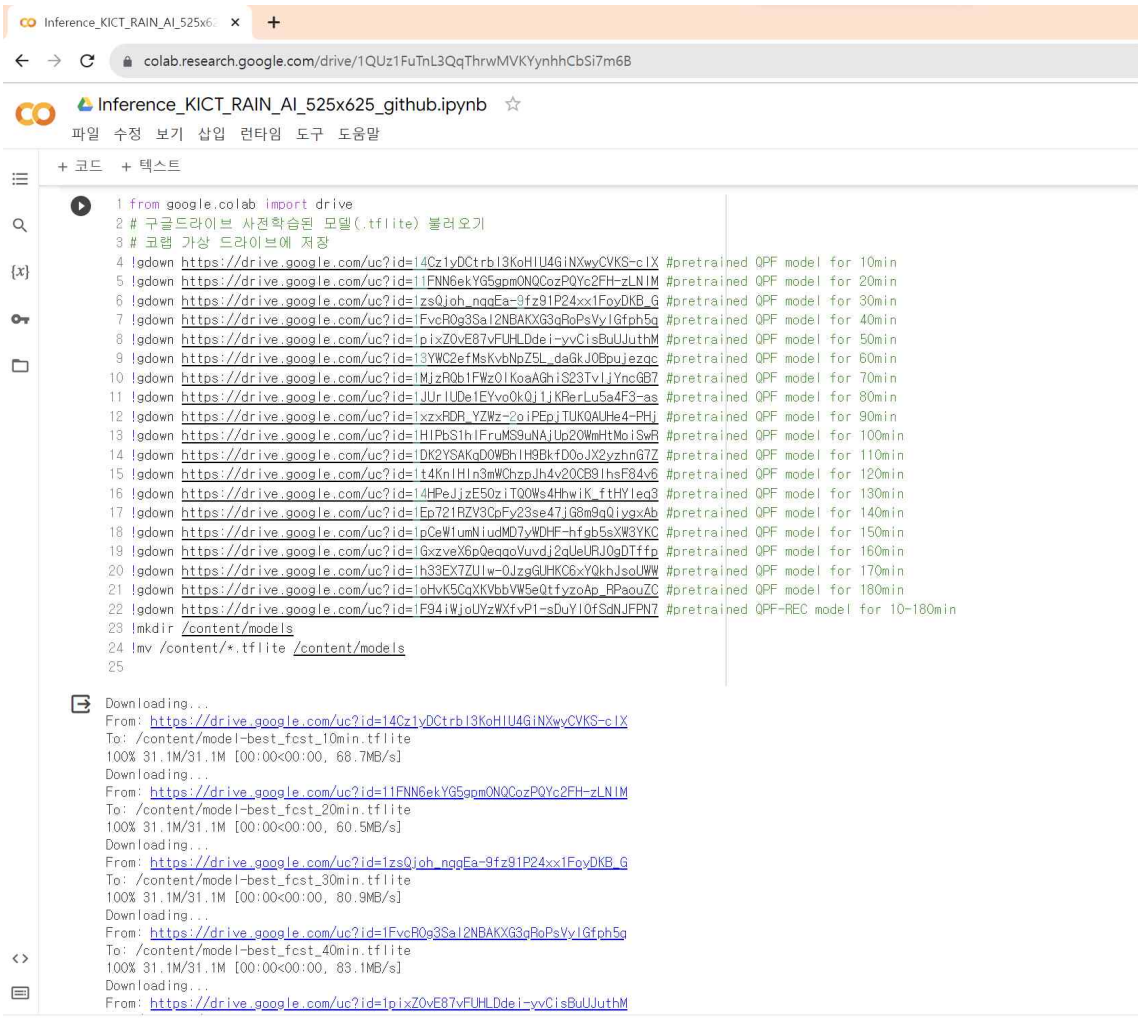
가. 다운로드

1. 실행파일 다운로드

(1) Google colab에서 실행가능한 주피터노트를 다운받아 사용한다.

- https://github.com/WaterKICT/KICT-RAIN-AI에서 Inference_KICT_RAIN_AI_525x625_github.ipynb을 다운로드 한다.

(2) 다운로드한 파일을 Google colab에서 실행한다.



2. 샘플 데이터 다운로드

(1) 샘플자료(INPUT_ASC.zip) 는 아래 구글 드라이브 링크를 통해 다운받을 수 있다.
https://drive.google.com/file/d/1n-gi8MI1ITpmF17Mr-epPvtIIbzuno_a/view?usp=drive_link

(2) 자료 설명

입력자료: 환경부 레이더 추정강우

단위: mm/hr (변경 가능)

격자 해상도: 1km / 10분

격자 크기: 625x525

샘플데이터링크: 2022년 8월 8일~13일 사례

나. 실행 방법(Google Colab)

다운받은 주피터 노트(Inference_KICT_RAIN_AI_525x625_github.ipynb)를 셀별 실행한다. 실행 과정에서 다운로드되는 사전학습된 모델파일, 입력자료 샘플, 출력파일은 /content.에 임시 저장된다. Colab 종료 후에 삭제되므로 필요시 별도로 다운 받아야 한다.

(1) 구글드라이브 사전학습된 모델(.tflite) 불러오기. 코랩 가상 드라이브에 저장

```
from google.colab import drive
# 구글드라이브 사전학습된 모델(.tflite) 불러오기
# 코랩 가상 드라이브에 저장
!gdown https://drive.google.com/uc?id=14Cz1yDCtrbI3KoHIU4GiNXwyCVKS-clX #pretrained QPF model for 10min
!gdown https://drive.google.com/uc?id=11FNN6ekYG5gpmONQCozPQYc2FH-zLNIM #pretrained QPF model for 20min
!gdown https://drive.google.com/uc?id=1zsQjoh_nqqEa-9fz91P24xx1FoyDKB_G #pretrained QPF model for 30min
!gdown https://drive.google.com/uc?id=1FvcR0g3Sal2NBAKXG3qRoPsVyIGfph5q #pretrained QPF model for 40min
!gdown https://drive.google.com/uc?id=1pixZ0vE87vFUHLDdei-yvCisBuUJuthM #pretrained QPF model for 50min
!gdown https://drive.google.com/uc?id=13YWC2efMsKvbNpZ5L_daGkJOBpujezqc #pretrained QPF model for 60min
!gdown https://drive.google.com/uc?id=1MjzRQb1FWz0IKoaAGhiS23TvIjYncGB7 #pretrained QPF model for 70min
!gdown https://drive.google.com/uc?id=1JUrIUDe1EYvo0kQj1jKRerLu5a4F3-as #pretrained QPF model for 80min
!gdown https://drive.google.com/uc?id=1xzxRDR_YZWz-2oiPEpjTUKQAUHe4-PHj #pretrained QPF model for 90min
!gdown https://drive.google.com/uc?id=1HIPbS1hIFruMS9uNAjUp20WmHtMoiSwR #pretrained QPF model for 100min
!gdown https://drive.google.com/uc?id=1DK2YSAKqDOWBhIH9BkfDOoJX2yzhnG7Z #pretrained QPF model for 110min
!gdown https://drive.google.com/uc?id=1t4KnIHIn3mWChzpJh4v20CB9IhsF84v6 #pretrained QPF model for 120min
!gdown https://drive.google.com/uc?id=14HPeJjzE50ziTQOWs4HhwiK_ftHYIeq3 #pretrained QPF model for 130min
!gdown https://drive.google.com/uc?id=1Ep721RZV3CpFy23se47jG8m9qQiygxAb #pretrained QPF model for 140min
!gdown https://drive.google.com/uc?id=1pCeW1umNiudMD7yWDHF-hfgb5sXW3YKC #pretrained QPF model for 150min
!gdown https://drive.google.com/uc?id=1GxzveX6pQeqqoVuvdj2qUeURJOgDTffp #pretrained QPF model for 160min
!gdown https://drive.google.com/uc?id=1h33EX7ZUIw-0JzgGUHKC6xYQkhJsoUWW #pretrained QPF model for 170min
!gdown https://drive.google.com/uc?id=1oHvK5CqXKVbbVW5eQtfyzoAp_RPaouZC #pretrained QPF model for 180min
!gdown https://drive.google.com/uc?id=1F94iWjoUYzWXfvP1-sDuYI0fSdNJFPN7 #pretrained QPF-REC model for 10-180min
!mkdir /content/models
!mv /content/*.tflite /content/models
```

(2) Input data 불러오기

```
# Input data 불러오기
# 코랩 가상 드라이브에 저장
!gdown https://drive.google.com/uc?id=1b2Thq96CqzYzaT34w10GZ-vPV9-q3XwU #INPUT_ASC_20220808.zip 파일
!unzip -qq '/content/INPUT_ASC_20220808.zip' -d '/content/data'
```

(3) 필요 라이브러리 불러오기

```
[ ]
import os
from tqdm import tqdm
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import  train_test_split, KFold
from sklearn.metrics import mean_absolute_error
from sklearn.utils import shuffle
from tqdm import tqdm
import time
import glob
import tensorflow as tf
import cv2
import gc
from PIL import Image
from keras.models import load_model
import datetime
import keras
import keras.backend as K
from keras.layers import *
from keras.models import Model
from keras.losses import mean_absolute_error
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Nadam
from keras.metrics import AUC
```

(4) 필요 함수 정의

```
def Rain(array):
    R=(array*255.)
    return R

def CSI_m(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    y_true = y_true.reshape(1, -1)[0]
    y_pred = y_pred.reshape(1, -1)[0]
    remove_NAs = y_true > 0
    y_true = np.where(y_true[remove_NAs] >= 0.1, 1, 0)
    y_pred = np.where(y_pred[remove_NAs] >= 0.1, 1, 0)
    right=np.sum(y_true * y_pred == 1)
    #print(right,np.sum(y_pred),np.sum(y_true),right)
    CSI = right/(np.sum(y_pred)+np.sum(y_true)-right+1e-07)
    return CSI

def CSI_custom(y_true, y_pred):
    score = tf.py_function(func=CSI_m, inp=[y_true, y_pred], Tout=tf.float32,  name='CSI_custom')
    return score
```

(5) 예측 시점 입력

```
[ ]
# 예측 시점 입력
!mkdir /content/QPF
date_nm='202208082200' #날짜 수정
run_date=date_nm[2:12]
print(run_date)
```

(6) 예측시점 기준으로 필요한 입력자료 구성 및 reading

```
[ ]
# READ INPUT DATA (PAST 4 data)
def read_ASC(timestamp):
#    print(timestams)
    YY = timestamp[0:2]
    MM = timestamp[2:4]
    DD = timestamp[4:6]
    HH = timestamp[6:8]
    MN = timestamp[8:10]
    RDR_time = timestamp[0:10]
    YYYY = '20'+YY
    print(RDR_time)
    path='/content/data'
    filename = path+'/RDR_CMP_HSP_PUB_20'+RDR_time+'-525x625-1km.asc'
    data = np.zeros([625,525], np.float32) #ASCII랑배열 반대
    data = np.loadtxt(filename,skiprows=6)
    data[data < 0] = 0.
    attr = datetime.datetime(int(YYYY),int(MM),int(DD),int(HH),int(MN))
    return data, attr

def qpe_data(filetime):
    YY = filetime[0:2]
    MM = filetime[2:4]
    DD = filetime[4:6]
    HH = filetime[6:8]
    MN = filetime[8:10]
    YYYY = '20'+YY
    latest_datetime = datetime.datetime(int(YYYY),int(MM),int(DD),int(HH),int(MN))
    #print(latest_datetime)
    list_for_qpf = [ts.strftime("%y%m%d%H%M") for ts in
    [latest_datetime - datetime.timedelta(minutes=t) for t in [30, 20, 10, 0]]]
    input_scans = np.array([read_ASC(ts)[0] for ts in list_for_qpf])
    qpe_scans = np.concatenate([input_scans], axis=0)
    print(qpe_scans.shape)
    return qpe_scans
```

(7) 예측시점 기준으로 KICT-RAIN-AI_ver1 실행

```
[ ]
#재귀적 학습 기반 강우예측 모델을 이용한 예측강우 생산
!gdown https://drive.google.com/uc?id=1F94iWjoUYzWXfvP1-sDuYlOfSdNJFPN7 #pretrained QPF-REC model for 10-180min
!mv /content/*.h5 /content/models

def prediction_rec(model_instance, input_data): #예측시 recursive 안함
    input_data = data_preprocessing(input_data)
    pred = model_instance.predict(input_data)
    pred=np.squeeze(pred)
    nwcst = np.transpose(pred,(2,0,1))
    return nwcst

model_rec = load_model('/content/models/model-best_rec_180min_f.h5',custom_objects={"CSI_custom": CSI_custom,"mae_custom":mae_custom})

# 데이터 준비하기
#FROM ASC (KMA mm/hr)
dataset0=qpe_data(run_date)
print('1',dataset0.shape)
dataset = np.transpose(dataset0,(1,2,0))
print('2',dataset.shape)
dataset = np.pad(dataset, [(7, 8), (9, 10), (0, 0)], mode='constant', constant_values=0)
feature0 = dataset[:, :, :]
feature = np.where(feature0 < 0., 0, feature0)
nwcst = prediction_rec(model_rec, feature/255.)
nwcst = nwcst*255
print('3',nwcst.shape)
#매 예측시점 자료를 ASCII GRID 형식으로 저장(EPSG518 기준)
for j in range(10, 190, 10):
    output_file_path = '/content/QPF/QPF_REC_'+date_nm+'-'+str(j)+'.asc'
    k=int(j/10-1)
    qpf_np = nwcst[k,7:632, 9:534]
    save_as_ascii_grid(qpf_np, output_file_path)
qpf_all = nwcst[:,7:632, 9:534]
print(qpf_all.shape)
```

(8) 예측시점 기준으로 KICT-RAIN-AI_ver2 실행

```
[ ]
import numpy as np
from tensorflow.lite.python import interpreter as interpreter_wrapper

def data_preprocessing(X):
    # 0. Right shape for batch
    X = X[np.newaxis, :, :, :]
    return X

def data_postprocessing(nwcst):
    # 0. Squeeze empty dimensions
    nwcst = np.squeeze(np.array(nwcst))
    nwcst = nwcst
    nwcst = np.where(nwcst>0, nwcst, 0)
    nwcst = nwcst[7:632, 9:534]
    return nwcst

def prediction(model_instance, input_data):
    input_data = data_preprocessing(input_data)
    print(input_data.shape)
    pred = model_instance.predict(input_data)
    print(pred.shape)
    nwcst = np.squeeze(pred)  # np.transpose() 대신 np.squeeze()를 사용하세요.
    return nwcst

def save_as_ascii_grid(data, output_file):
    """
    2D NumPy array를 ASCII grid 파일로 저장하는 함수
    :param data: 2D NumPy array (격자 자료)
    :param output_file: 저장할 파일 경로 및 이름 (예: 'output.asc')
    """
    #EPSG 5186 기준
    nrows, ncols = data.shape
    header = f"ncols {ncols}₩n"
    header += f"nrows {nrows}₩n"
    header += "xllcorner -31000₩n"
    header += "yllcorner 123000₩n"
    header += "cellsize 1000.0₩n"  # 셀 크기
    header += "NODATA_value -9999.0₩n"  # 누락된 데이터를 나타내는 값

    np.savetxt(output_file, data, fmt="%.2f", header=header, comments="", delimiter=" ")
```

```python
def prediction_tflite(feature):
    qpf_all_each = np.empty((625, 525, 0))
    for j in range(10, 190, 10):
    # 이전 코드에서 필요한 설정과 모델 로딩을 진행한 후
        pre_model = f'/content/models/model-best_fcst_{j}min.tflite'
        interpreter = interpreter_wrapper.Interpreter(model_path=pre_model)
        interpreter.allocate_tensors()

        # 입력 텐서 정보 가져오기
        input_details = interpreter.get_input_details()
        output_details = interpreter.get_output_details()
        input_shape = input_details[0]['shape']

        # 입력 데이터 변환 및 설정
        feature = np.array(feature/255., dtype=np.float32)  # FLOAT64에서 FLOAT32로 변환
        interpreter.set_tensor(input_details[0]['index'], feature)
        # 모델 실행
        interpreter.invoke()

        # 출력 데이터 가져오기
        nwcst = interpreter.get_tensor(output_details[0]['index'])
        nwcst = data_postprocessing(nwcst)*255.
        # 예측하기
        qpf_np = nwcst

        #매 예측시점 자료를 ASCII GRID 형식으로 저장(EPSG518 기준)
        output_file_path = '/content/QPF/QPF_'+date_nm+'-'+str(j)+'.asc'
        save_as_ascii_grid(qpf_np, output_file_path)
        #이미지 처리를 위한 3차원 배열 예측강우 생성
        qpf_np = np.expand_dims(qpf_np, axis=-1)  # 차원을 늘리세요.
        qpf_all_each = np.concatenate([qpf_all_each, qpf_np], axis=-1)
    return qpf_np, qpf_all_each
#RUN
dataset0=qpe_data(run_date)
dataset = np.transpose(dataset0,(1,2,0))
dataset = np.pad(dataset, [(7, 8), (9, 10), (0, 0)], mode='constant', constant_values=0)
feature0 = dataset[:, :, :]
feature = np.where(feature0 < 0., 0, feature0)
feature=data_preprocessing(feature)
print(feature.shape)
```