

โครงการวิศวกรรมไฟฟ้า วิชา 2102499 ปีการศึกษา 2565

การวิเคราะห์ภัยพิบัติน้ำท่วม
Flood disaster analysis

นายกฤษฎ์ ช่างเมืองชัย รหัสนิสิต 6230011221

อาจารย์ที่ปรึกษา ผศ.ดร. วิทยากร อัครวิเศษ

ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

Outline

• บทนำ (Introduction)

- ที่มาและความสำคัญ
- วัตถุประสงค์ ขอบเขตโครงการ และผลลัพธ์ที่คาดหวังจากโครงการ

• ขั้นตอนและหลักการ (Procedure)

- ภูมิศาสตร์น้ำท่วมของประเทศไทย
- ดาวเทียม Sentinel-1 จาก Copernicus
- Google earth engine
- สถาปัตยกรรมแบบ U-Net และแบบ 3D Dimension Reducer U-Net (3DDR U-Net)
- เครื่องมือวัดรูปทำนายและรูปจริง

• ผลลัพธ์จากการดำเนินงาน (Result)

- ผลลัพธ์แบบจำลอง (Visualization)
- ผลลัพธ์จากการเก็บข้อมูล (Collect data)
- ผลลัพธ์จากการทำนายโดยใช้โมเดล 3DDR U-Net (Prediction)

• สรุปผลและข้อเสนอแนะ (Conclusion)

หมายเหตุ :



รูปสัญลักษณ์นี้ที่พบในแต่ละหน้านั้นหมายถึง
ผู้จัดทำได้ทำการนำเสนอไปแล้วครั้งก่อน

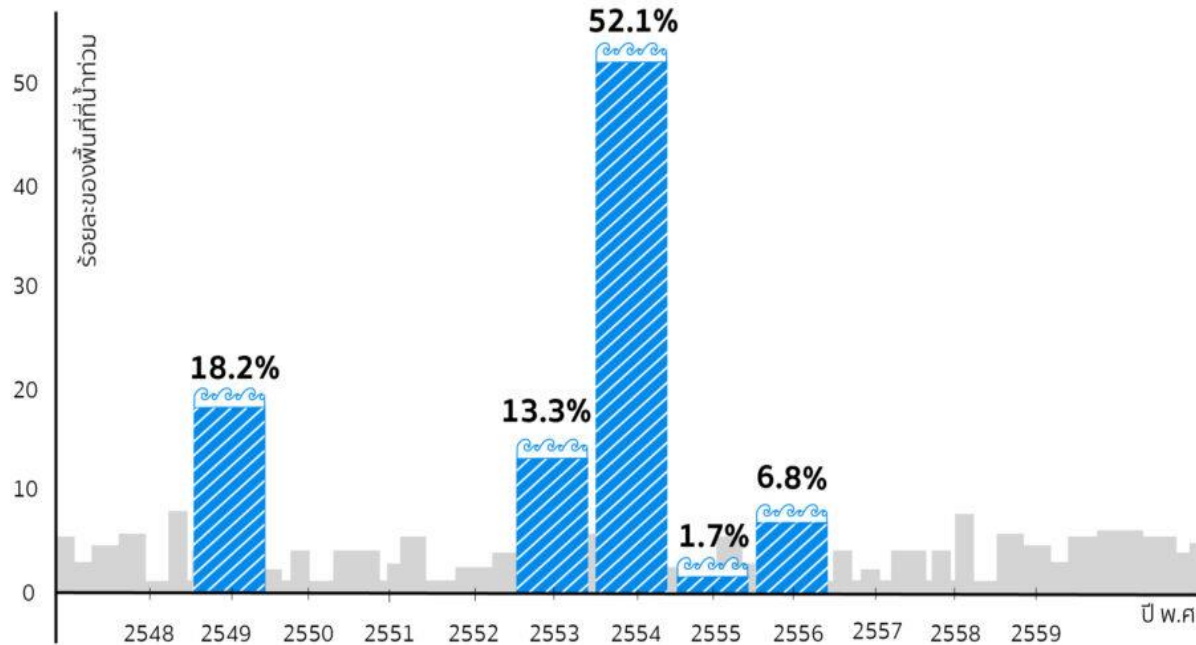
บทนำ (Introduction)

Introduction



ที่มาและความสำคัญ

สรุปสถานะน้ำท่วมในพื้นที่กรุงเทพมหานครย้อนหลัง 12 ปี



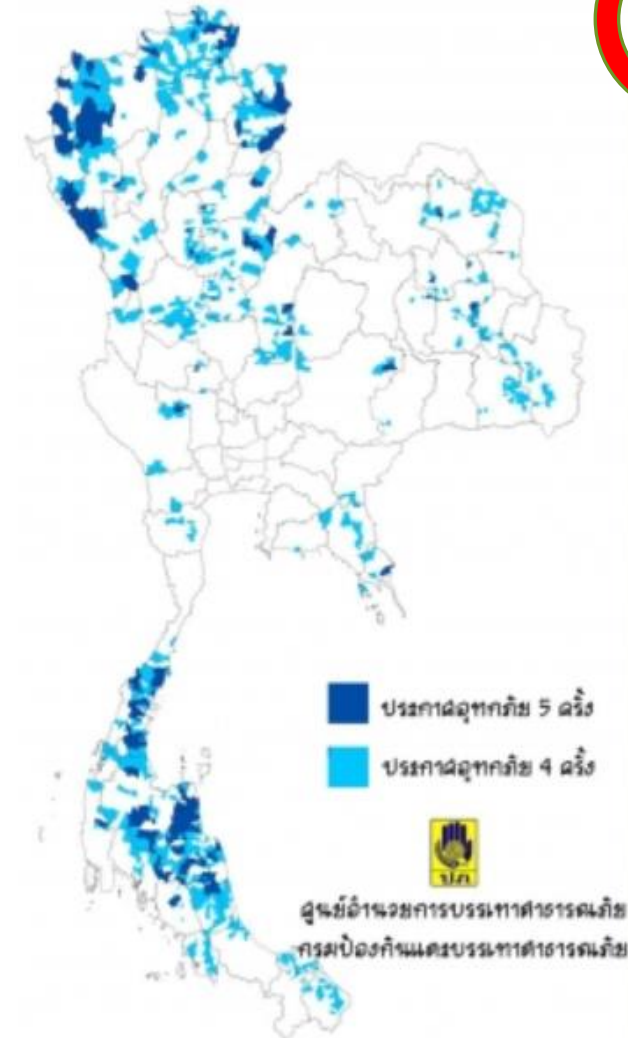
- น้ำท่วมนั้นเป็นภัยธรรมชาติที่อยู่กับประเทศไทยมาอย่างยาวนาน
- ประเทศไทยนั้นอยู่ในอิทธิพลของมรสุมของภูมิภาคเอเชียตะวันออกเฉียงใต้ ส่งผลให้ประเทศไทยได้รับมรสุมทุกปี
- ประชาชนจำนวนมากอาศัยอยู่ตามตำแหน่งที่เป็นราบแอ่งน้ำหรือติดกับริมคลอง ซึ่งเหมาะแก่การเกิดน้ำขังและน้ำล้นตลิ่ง
- เหตุการณ์น้ำท่วมนั้นสร้างความเสียหายต่อชีวิตและทรัพย์สินของประชาชน

ที่มา: <https://theurbanis.com/environment/>

สถิติการประกาศอุทกภัยบ่อยครั้ง 2555-2561 (มากกว่า 4 ครั้ง)

ภาค	จังหวัด	อำเภอ	ตำบล	หมู่บ้าน
ภาคเหนือ	17	106	350	1,660
ภาคตะวันออกเฉียงเหนือ	14	48	141	619
ภาคกลาง	10	23	38	146
ภาคใต้	13	86	389	2,152
รวม	54	263	918	4,577

- น้ำท่วมในประเทศไทยนั้นส่งผลกระทบต่อทั้ง 4 ภาคของประเทศไทย
- ปัจจุบันการประกาศเตือนภัยน้ำท่วมในประเทศไทยนั้นใช้วิธีการคาดการณ์ฝนและการเพิ่มขึ้นของระดับน้ำตามริมคลองเป็นหลัก
- การประกาศแจ้งเตือนน้ำท่วมจะอาศัยการเฝ้าตามข่าวบนสื่อ
- การคาดการณ์พื้นที่น้ำท่วมด้วย Satellite Image ผสมผสานกับ Deep learning ยังไม่เป็นที่แพร่หลายในไทย



ที่มา: <http://direct.disaster.go.th/cmsdetail.directing>

วัตถุประสงค์



- เพื่อศึกษาและพัฒนาองค์ความรู้ในการใช้งานแบบจำลอง U-Net
- เพื่อวิเคราะห์ปัญหาของสภาพภูมิศาสตร์ของพื้นที่ที่เกิดน้ำท่วมบ่อยครั้ง
- เพื่อสร้างแบบจำลองทำนายพื้นที่ที่มีโอกาสเกิดน้ำท่วมแบบอัตโนมัติจากชุดภาพถ่ายดาวเทียมของการเกิดน้ำท่วมในอดีต

ขอบเขตโครงการ

- ศึกษา Application Program Interface (API) ของการเรียกใช้ภาพถ่ายดาวเทียม
- ศึกษาโปรแกรม Python หรือ MATLAB ในการสร้างแบบจำลองการเรียนรู้เครื่อง (Machine Learning Model)
- นำแบบจำลองการเรียนรู้เครื่องที่เรียนรู้แล้วมาทำนายน้ำท่วมกับพื้นที่ขนาด 10 กิโลเมตร x 10 กิโลเมตร หรือ ความละเอียด (Resolution) ที่น้อยกว่านี้

ผลลัพธ์ที่คาดหวังจากโครงการ

- ทดลองใช้แบบจำลอง U-Net ในการทำนาย (prediction) น้ำท่วม
- ชุดซอฟต์แวร์ที่รับชุดของภาพถ่ายดาวเทียม และทำนายบริเวณที่น้ำท่วมจะเกิดขึ้น

ขั้นตอนและหลักการ (Procedure)

1. ภูมิศาสตร์น้ำท่วมของประเทศไทย

ประเภทของน้ำท่วม



ที่มา : <https://disaster.go.th>

1. น้ำท่วมแบบล้นตลิ่ง

สาเหตุ ระดับน้ำในแม่น้ำขึ้นสูงผิดปกติ

2. น้ำท่วมแบบป่าไหลหลาก

สาเหตุ ฝนตกหนักบริเวณหุบเขา

3. น้ำท่วมแบบท่วมขัง

สาเหตุ พื้นดินมีลักษณะเป็นแอ่ง

ตารางเปรียบเทียบน้ำท่วมของแต่ละภูมิภาค

ภูมิภาค	แม่น้ำสำคัญที่ไหลผ่าน	พื้นที่น้ำท่วม กันยา 2011 (ไร่)		พื้นที่น้ำท่วม กันยา 2021 (ไร่)	
ภาคเหนือ	แม่น้ำปิง, แม่น้ำวัง, แม่น้ำยม, แม่น้ำน่าน	350,015	2.19%	45,538	0.81%
ภาคตะวันออกเฉียงเหนือ	แม่น้ำชี, แม่น้ำมูล, แม่น้ำพรม	4,412,704	27.59%	2,222,388	39.35%
ภาคกลาง	แม่น้ำป่าสัก, แม่น้ำเจ้าพระยา, แม่น้ำท่าจีน	9,709,429	60.70%	3,209,329	56.82%
ภาคตะวันออก	แม่น้ำบางปะกง, แม่น้ำนครนายก	1,207,294	7.55%	164,094	2.91%
ภาคตะวันตก	แม่น้ำกลอง	258,127	1.61%	6,903	0.12%
ภาคใต้	แม่น้ำตาปี, แม่น้ำหลังสวน, แม่น้ำตรัง	65,581	0.40%	0	0%
รวม		15,996,150	100%	5,648,252	100%

ที่มา : https://Gistda.or.th/news_view.php?n_id=6360&lang=EN



2. ดาวเทียม Sentinel-1 จาก Copernicus

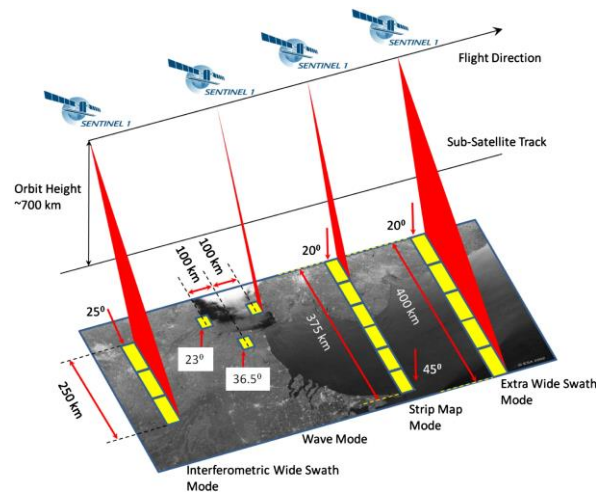
1. Understanding Data

2. Collect Data

3. Data analyze

4. Train Data

5. Test and Evaluate



ที่มา : <https://sentinels.copernicus.eu/>

ที่มา : <http://link.springer.com>

Water Features

Backscatter (dB)

	HH	HV	VH	VV
Flood Water	-8 to -12	-15 to -24	-15 to -24	-6 to -15
River Water	-16 to -30	-24 to -36	-24 to -36	-19 to -32
Tank Water	-13 to -26	-22 to -40	-22 to -40	-16 to -28
Oxbow Lake	-16 to -24	-21 to -32	-21 to -32	-24 to -32
Partially Submerged Features	-18 to -30	-24 to -34	-24 to -34	-8 to -18

- ทำความเข้าใจระบบของดาวเทียม Sentinel-1 ที่ใช้เรดาร์รับแสงแบบ Synthetic Aperture Radar (SAR) และ เลือกแบนด์ความถี่และ Product เป็น C-band และ Ground Range Detected (GRD) ตามลำดับ
- เลือกโพลาไรเซชัน (Polarization) ของเรดาร์โดยอ้างอิงจากงานวิจัย โดย VV และ HH จะเหมาะสมกับการตรวจจับน้ำท่วมมากที่สุด

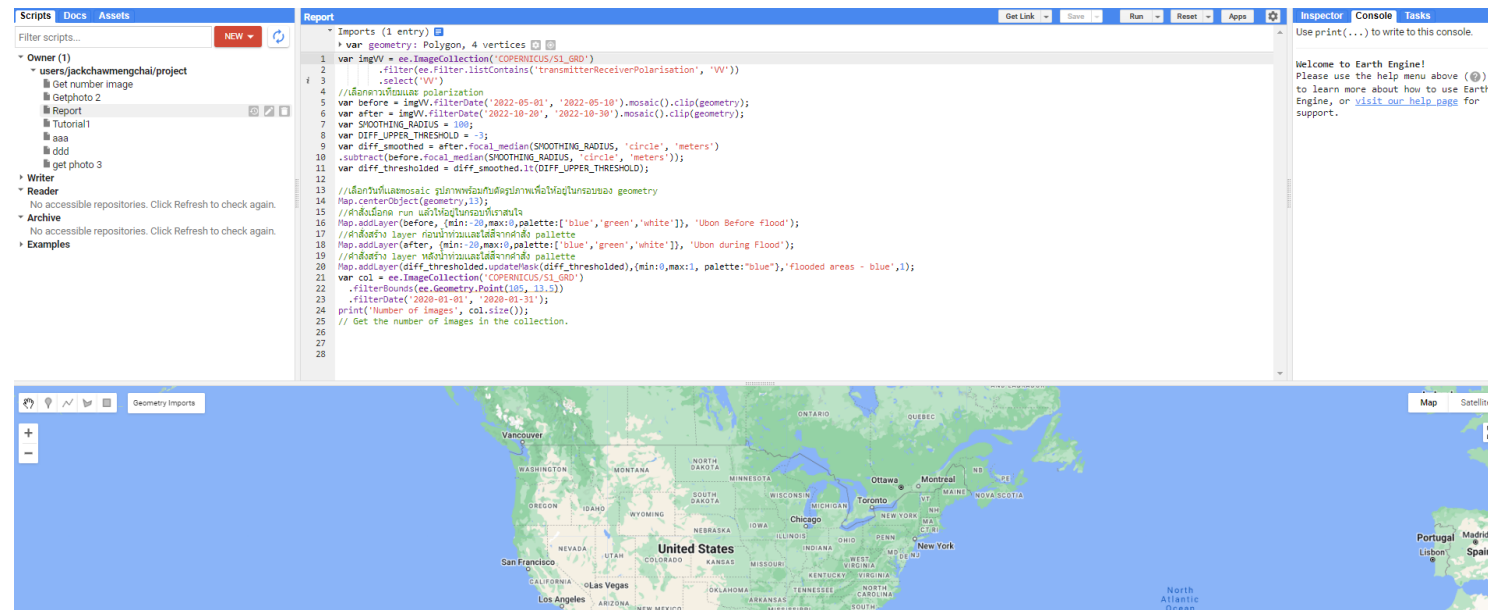
ขั้นตอนการทำ Machine learning



3. Google earth engine



- เลือกใช้ Google Earth Engine เนื่องจากตัว Google Earth Engine นั้นมีความสามารถที่จะเขียน Code เพื่อตัดแปลงให้ได้ Data ตามที่ต้องการได้
- ทำการทดลองโดยลอง Coding ภาษา Java เพื่อดูผลลัพธ์ที่อยากได้ในงานอนาคตก่อน (Visualization) หลังจากนั้นก็จะใช้ Google Colab

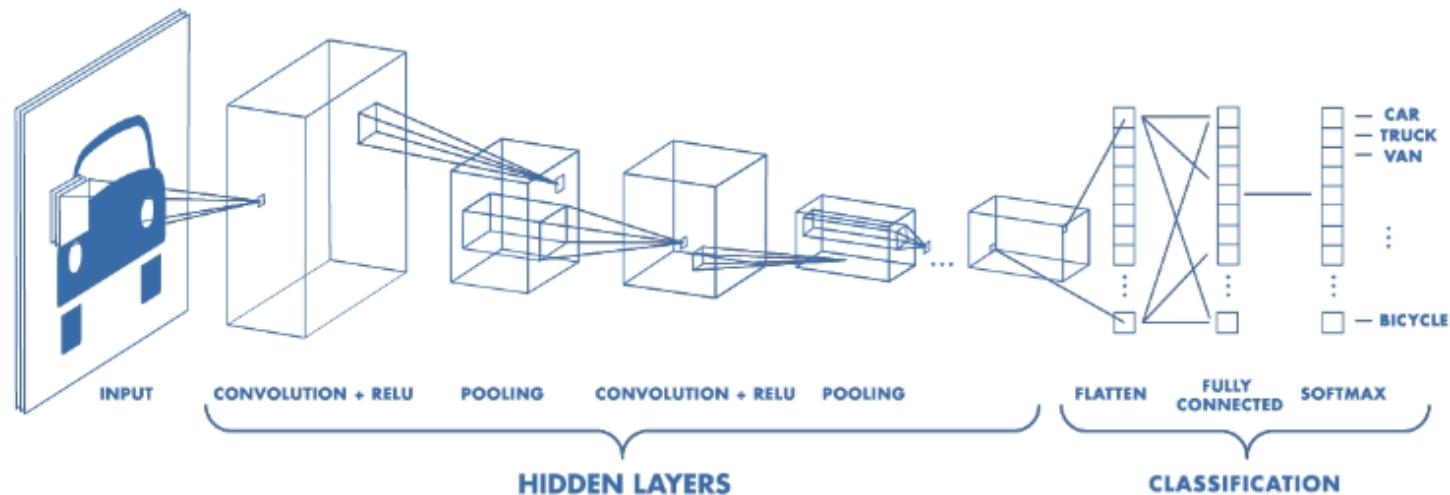


ที่มา : <https://earthengine.google.com/>



4. สถาปัตยกรรมแบบ U-Net และแบบ 3DDR U-Net

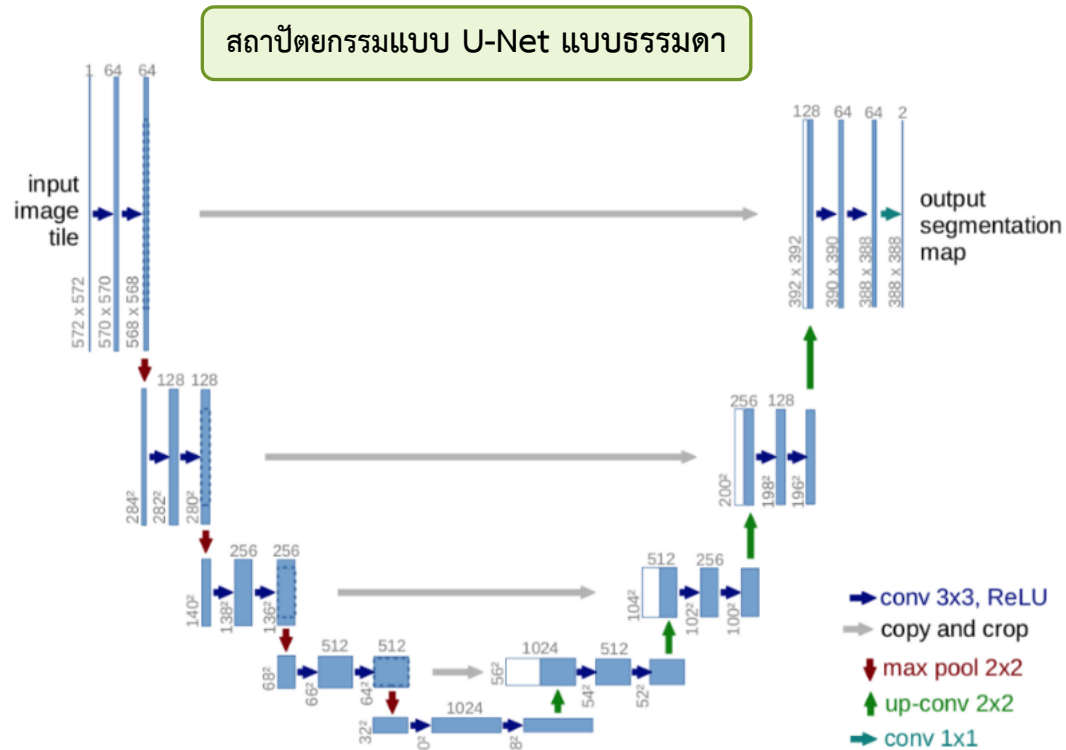
- เลือกใช้ Deep learning แบบ CNN เนื่องจากข้อมูลที่จะนำไปฝึกนั้นจะเป็นแบบ ข้อมูล 2 มิติ
- หลักการทำงานจะจำลองคล้ายกับการมองเห็นของมนุษย์ แบ่งการประมวลผลเป็นพื้นที่ย่อยและขยับพื้นที่นั้นไปให้ครบทั้งรูป
- U-Net เป็นหนึ่งในชนิดของ CNN ถูกคิดค้นขึ้นครั้งแรกโดยใช้กับทางการแพทย์เพื่อระบุจุดของความผิดปกติในภาพถ่าย



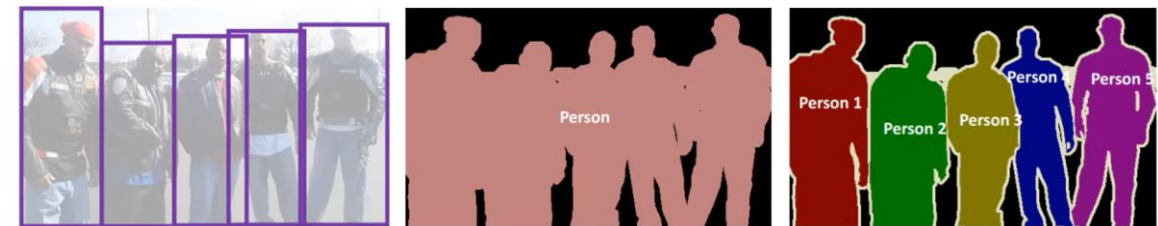
ที่มา : <https://towardsdatascience.com/>



- หลักการทำงานของ U-Net คือนำรูปมาวางเป็นชั้นๆหลาย layer โดยที่แต่ละชั้นจะถูกเชื่อมโยงกัน มี CNN สองขา คือ ย่อ (encoder) กับขยาย (decoder) ขั้นตอนของ Model จะคล้ายตัว U
- U-Net จะเน้นไปที่การทำงานแบบ Semantic Segmentation ซึ่งสามารถทำ Instance Segmentation เพิ่มเติมซึ่งคือการแยกส่วนแต่ละวัตถุในภาพแยกจากกันอย่างสิ้นเชิงโดยกำหนดคลาสในแต่ละจุดพิกเซลว่าเป็นคลาสแต่ละชนิดใด
- U-Net มีการนำไปใช้ในการทำ Segmentation ที่สามารถนำมาใช้ในการคาดการณ์ (Prediction) นำท่วมได้



ที่มา : <https://lmb.informatik.uni-freiburg.de/>



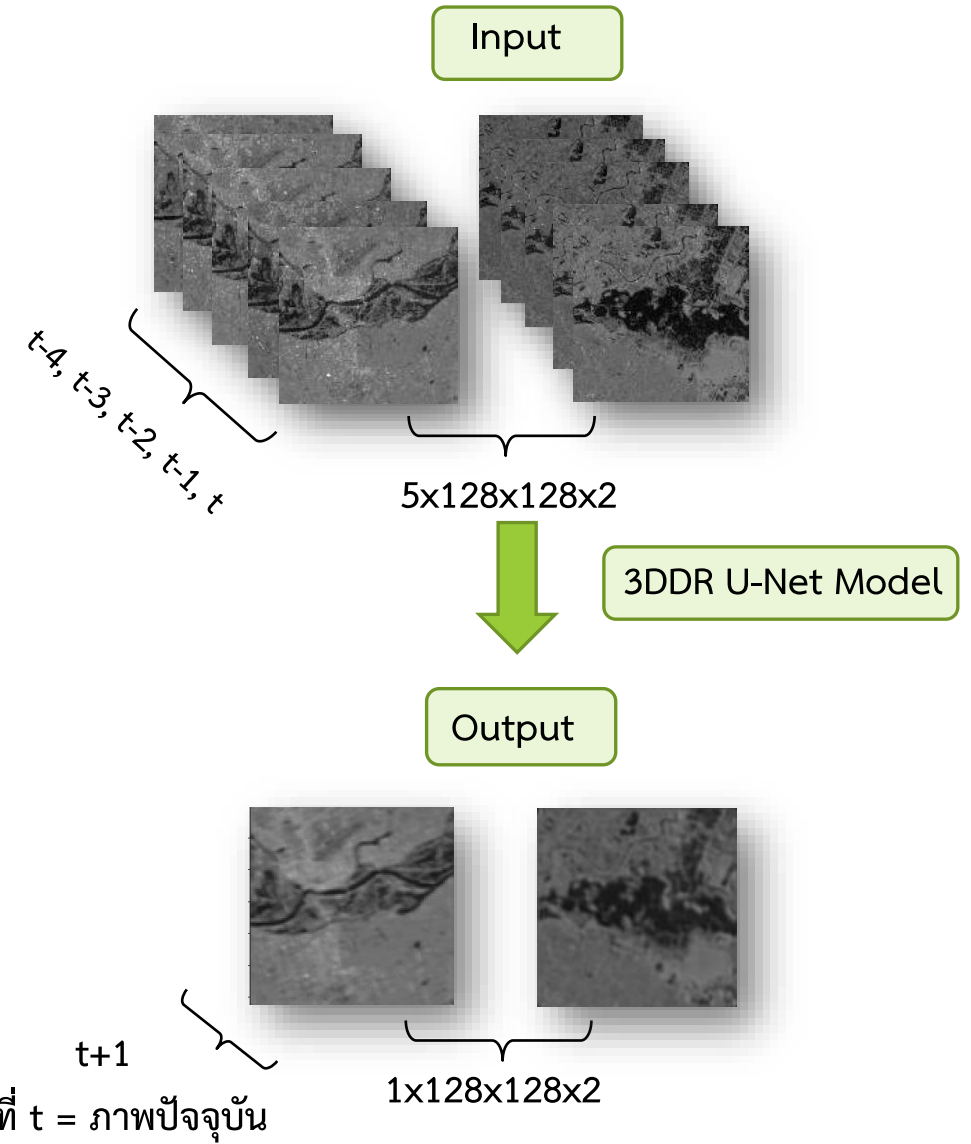
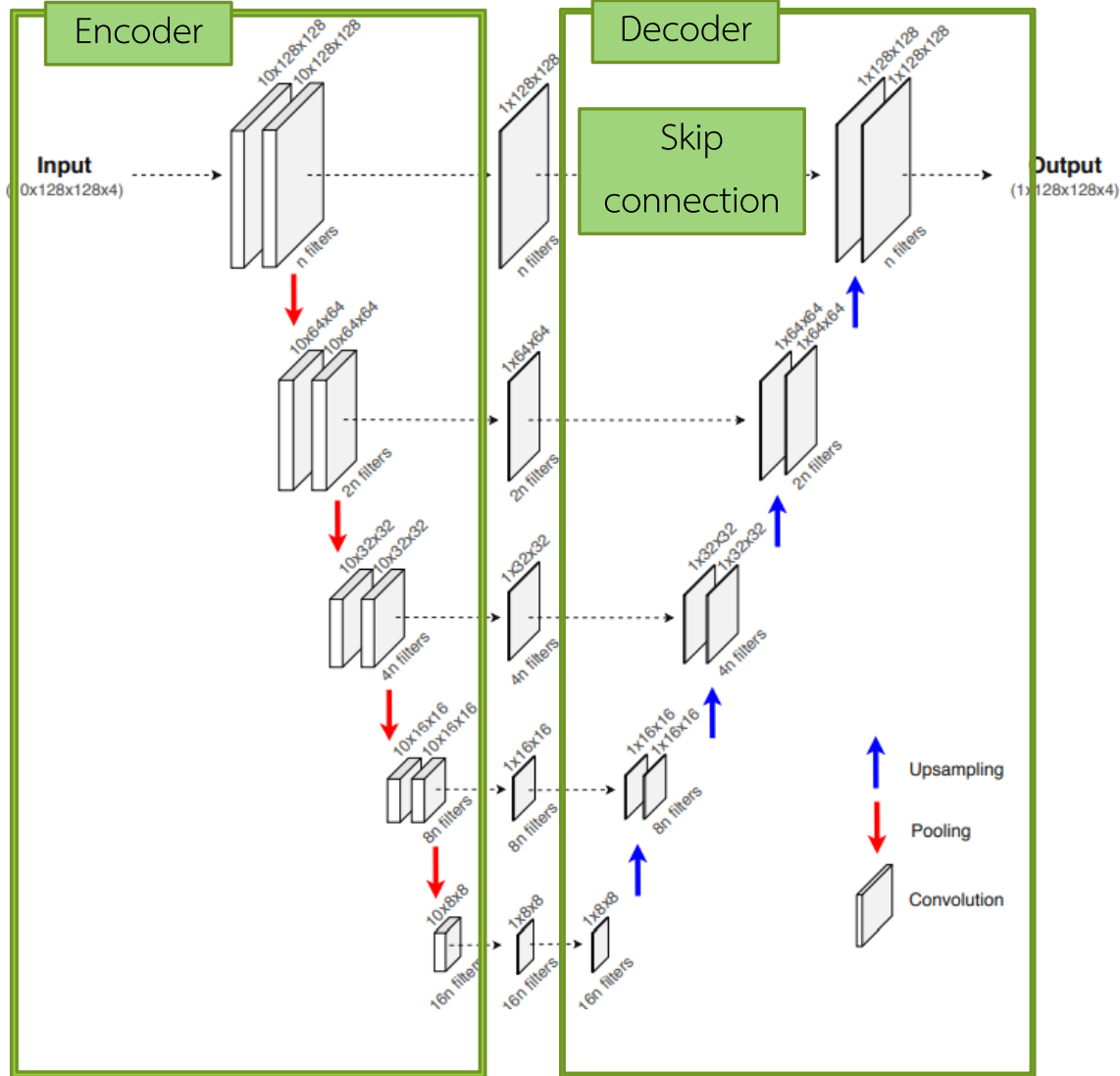
Object Detection

Semantic Segmentation

Instance Segmentation

ที่มา : <https://youtube/DigitalSreeni>

สถาปัตยกรรม 3D Dimension Reducer U-Net (3DDR U-Net)



- 3DDR U-Net จะสามารถ Input เป็นแบบ 4 มิติและลดขนาดมิติ Channel ของ Output เป็นขนาดที่ต้องการได้ เช่นจากตัวอย่างขนาด Channel หรือ Lag ของ Input คือ 5 และ Output คือ 1

5. เครื่องมือวัดรูปทำนายและรูปจริง

- Mean square error (MSE)

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

โดยที่

m คือจำนวนแถว (Row) ของรูปภาพ

n คือจำนวนคอลัมน์ (Column) ของรูปภาพ

$I(i,j)$ คือค่าของ pixel รูปจริง ณ แถวที่ i และคอลัมน์ j

$K(i,j)$ คือค่าของ pixel รูปทำนาย ณ แถวที่ i และคอลัมน์ j

- Peak signal-to-noise ratio (PSNR)

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

โดยที่

MAX_I คือค่าจากสมการ (2^b-1) โดยที่ b นั้นหมายถึงจำนวนบิตที่ใช้แทนหนึ่งจุดภาพ

MSE คือค่า Mean Square error

- Structural similarity index measure (SSIM)

$$SSIM(A,B) = \frac{(2\mu_A\mu_B + C_1)(2\sigma_{AB} + C_2)}{(\mu_A^2 + \mu_B^2 + C_1)(\sigma_A^2 + \sigma_B^2 + C_2)}$$

โดยที่

μ_A, μ_B คือค่าเฉลี่ยของ Pixel รูป A และ B ตามลำดับ (Pixel sample mean)

σ_A^2, σ_B^2 คือค่าเบี่ยงเบนมาตรฐาน (Standard deviations) ของ Pixel รูป A และ B ตามลำดับ




σ_{AB} คือค่าความแปรปรวนร่วมเกี่ยว (Cross Covariance) ของ Pixel A และ B

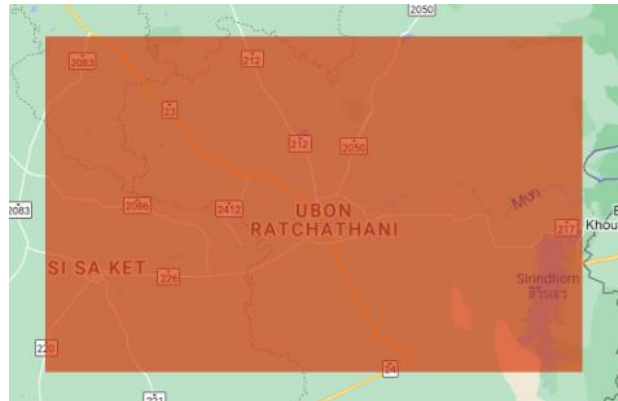
ผลลัพธ์จากการดำเนินงาน (Result)



1. ผลลัพธ์แบบจำลอง (Visualization)

1. รับภาพข้อมูลจากดาวเทียมผ่าน Image collection

```
Imports (1 entry)   
  ▶ var geometry: Polygon, 4 vertices    
  1 var imgVV = ee.ImageCollection('COPERNICUS/S1_GRD')  
  2   .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VW'))  
  3   .select('VW')  
  4 //เลือกดาวเทียมและ polarization
```



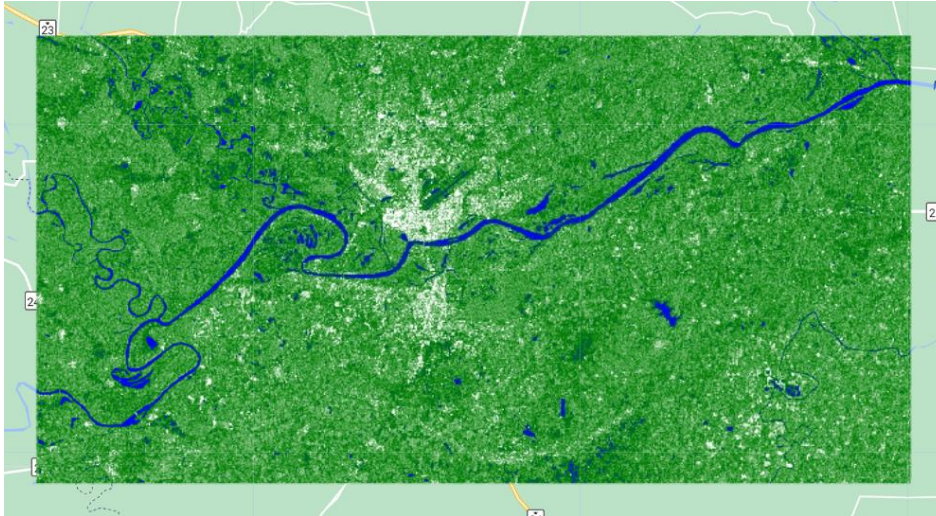
2. เลือกวันที่ของ before และ after

```
4 //เลือกดาวเทียมและ polarization  
5 var before = imgVV.filterDate('2022-05-01', '2022-05-10').mosaic().clip(geometry);  
6 var after = imgVV.filterDate('2022-10-20', '2022-10-30').mosaic().clip(geometry);
```

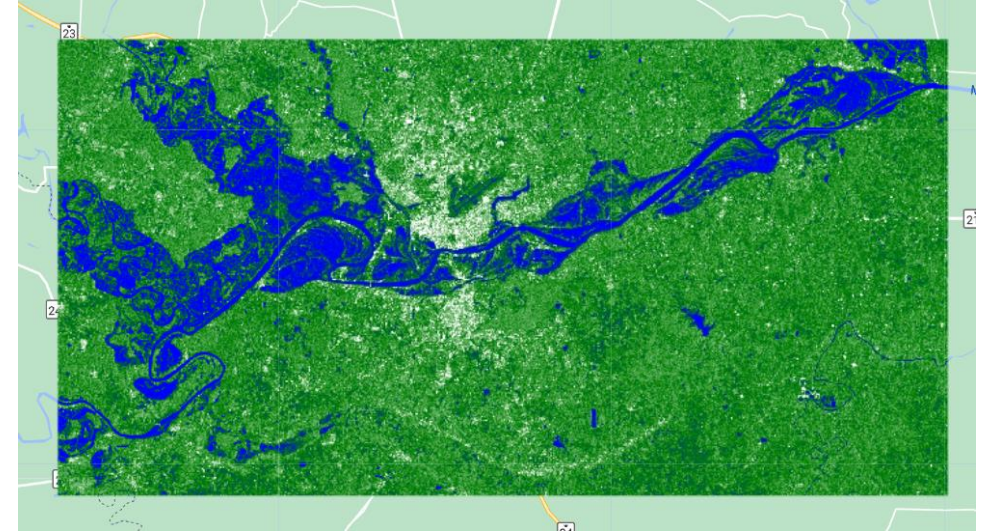
- ทำการ Visualization บน Google Earth Engine โดยเลือกดาวเทียม sentinel-1 และ Polarization แบบ VV
- เลือกพื้นที่เป็นขอบเขตผ่านคำสั่ง geometry เป็นจังหวัดอุบลราชธานี
- เลือกช่วงวันที่ before เป็นเดือน พฤษภาคม และ after เป็นเดือน ตุลาคม

3. ใช้คำสั่ง add layer
เพื่อโชว์ภาพ before
และ after

```
14 Map.centerObject(geometry,13);  
15 //คำสั่งเมื่อกด run แล้วให้อยู่ในกรอบที่เราสนใจ  
16 Map.addLayer(before, {min:-20,max:0,palette:['blue','green','white']}, 'Ubon Before flood');  
17 //คำสั่งสร้าง layer ก่อนน้ำท่วมและใส่สีจากคำสั่ง palette  
18 Map.addLayer(after, {min:-20,max:0,palette:['blue','green','white']}, 'Ubon during Flood');  
19 //คำสั่งสร้าง layer หลังน้ำท่วมและใส่สีจากคำสั่ง palette
```



ก่อนเกิดน้ำท่วม จังหวัดอุบลราชธานี



ช่วงเกิดน้ำท่วม จังหวัดอุบลราชธานี

- ใช้คำสั่ง add layer เพื่อโชว์ image ของ before และ after ที่ได้จากคำสั่ง image collection
- ภาพดาวเทียมที่ได้ของ before (ก่อนเกิดน้ำท่วม) เป็น ช่วงต้นเดือนพฤษภาคม จังหวัดอุบลราชธานีจะเห็นว่ายังมีสีน้ำเงินน้อย ส่วนภาพ after (ก่อนเกิดน้ำท่วม) ที่ได้ซึ่งเป็น ช่วงเดือนตุลาคม จะมีสีน้ำเงินเยอะ



4. นำภาพของ
before และ after
ลบกัน

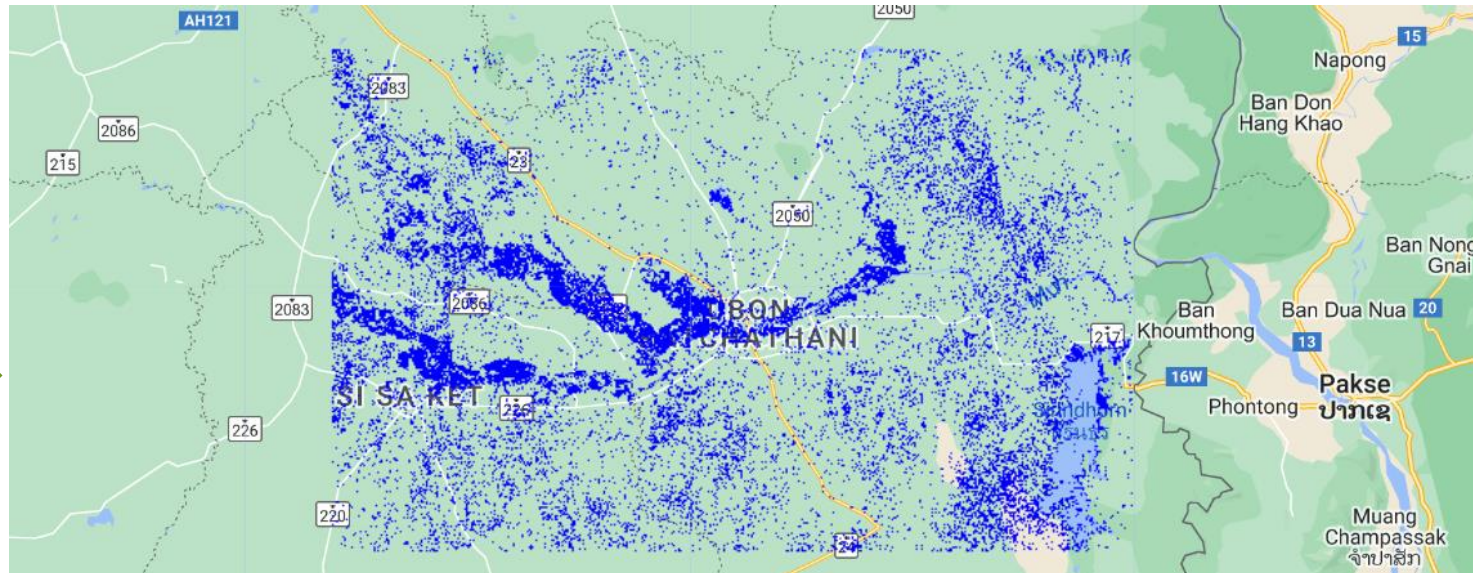


```
7 var SMOOTHING_RADIUS = 100;  
8 var DIFF_UPPER_THRESHOLD = -3;  
9 var diff_smoothed = after.focal_median(SMOOTHING_RADIUS, 'circle', 'meters')  
10 .subtract(before.focal_median(SMOOTHING_RADIUS, 'circle', 'meters'));  
11 var diff_thresholded = diff_smoothed.lt(DIFF_UPPER_THRESHOLD);
```

5. ใช้คำสั่ง add layer
เพื่อแสดงผลลัพธ์ของ
การลบกัน

```
20 Map.addLayer(diff_thresholded.updateMask(diff_thresholded),{min:0,max:1, palette:"blue"},'flooded areas - blue',1);
```

ผลลัพธ์



- ใช้คำสั่ง subtract และ focal_median เพื่อหาความแตกต่างของทั้ง 2 ภาพ
- ใช้คำสั่ง palette เพื่อใส่สีใน layer ให้เข้าใจภาพได้ง่ายขึ้น
- ผลลัพธ์จากการ ใช้คำสั่งนำรูปก่อนเกิดน้ำท่วมมาลบกับช่วงเกิดน้ำท่วมในจังหวัดอุบลราชธานี



2. ผลลัพธ์จากการเก็บข้อมูล (Collect data)

3. ใช้ for loop เพื่อ load data ให้ครบเดือน

1. เลือก Data ผ่าน Image collection

2. ใช้คำสั่ง Export image เข้าสู่ google drive

```
import time
geometry = ee.Geometry.Polygon(
    [[[105.11879124445844,15.098458530820098],
      [105.11879124445844,15.33036097900656],
      [104.64912571711469,15.33036097900656],
      [104.64912571711469,15.098458530820098]]]])
for day in range(1,30,1):
    print(day)
    myimage = ee.Image(ee.ImageCollection('COPERNICUS/S1_GRD')
        .filterBounds(geometry)
        .filterDate(ee.Date('2022-09-'+str(day)), ee.Date('2022-09-'+str(day+1)))
        .mosaic()
        .select('VV')
        .clip(geometry))
    print('load image complete')
    task = ee.batch.Export.image.toDrive(image=myimage,
        description='lookmee_SAR_Septem'+str(day),
        region=geometry,
        folder='September',
        scale=100,
        crs='EPSG:3857')

    task.start()

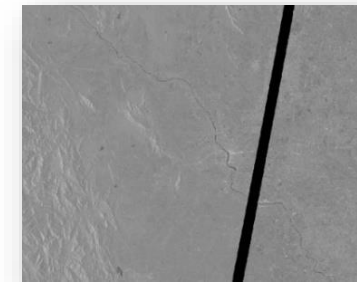
    while task.active():
        print('Waiting on (id: {}).'format(task.id))
        time.sleep(10)
```

ตารางเปรียบเทียบจำนวนรูปภาพดาวเทียม

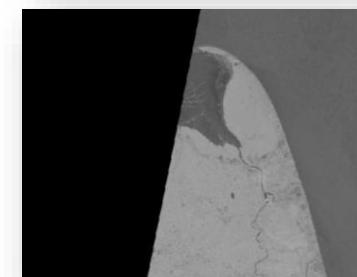
ภูมิภาค รูปภาพ	ภาคเหนือ*	ภาคตะวันออกเฉียงเหนือ	ภาคกลาง				ภาคใต้*
		อุบลราชธานี	พระนครศรีอยุธยา	นครสวรรค์	ปริมณฑล	นครศรีธรรมราช	
รูปดาวเทียมที่เก็บได้ ปี 2015 ถึง 2022	126	558	499	483	497	262	
รูปภาพที่ใช้งานได้	56	558	499	481	492	82	
จำนวนรูปที่มีเหตุการณ์น้ำท่วม	4	40	104	81	56	8	
จำนวนครั้งที่ท่วม (2015-2022)	-	3	7	5	4	-	
จำนวนภาพที่ใช้เทรนโมเดล	-	456	373	481	369	-	

- ใช้ Google Colab เพื่อ export image จาก Google Earth เข้าสู่ Google Drive
- Download data ภาพดาวเทียมย้อนหลังตั้งแต่ปี 2015 ถึงปี 2022
- ไฟล์ที่ได้คือไฟล์ tiff มีบางภูมิภาคที่ใช้งานไม่ได้เนื่องจากดาวเทียมโคจรไม่ผ่าน
- ขนาดรูปจาก 128x128 pixels จะมีขนาดตามแผนที่จริงคือ 13x13 กิโลเมตร

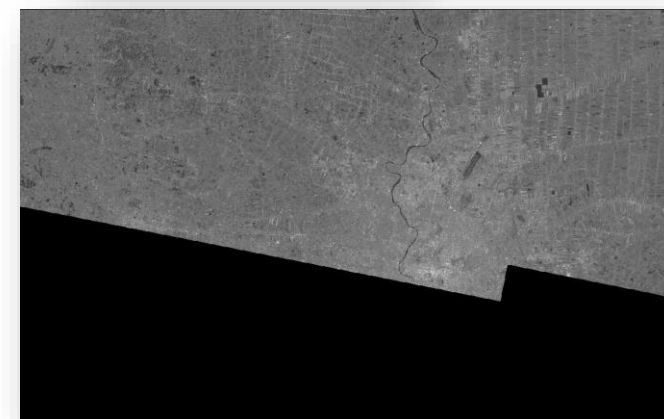
ตัวอย่างรูปภาพ ที่ไม่ได้นำมาเทรน



ภาคเหนือ



จังหวัดนคร
ศรีธรรมราช



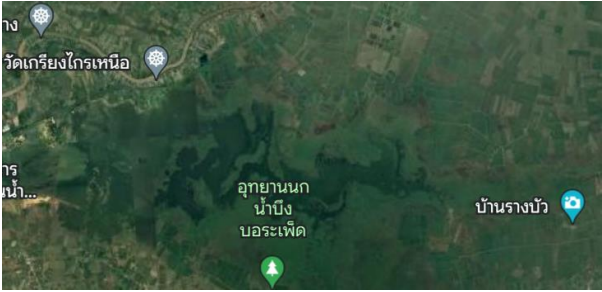
กรุงเทพและปริมณฑล

จำนวนครั้งน้ำท่วม

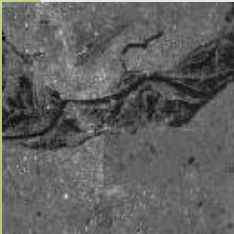
น้ำท่วมล้นตลิ่ง
ชุมชนรอบบึงบอระเพ็ด
จังหวัดนครสวรรค์



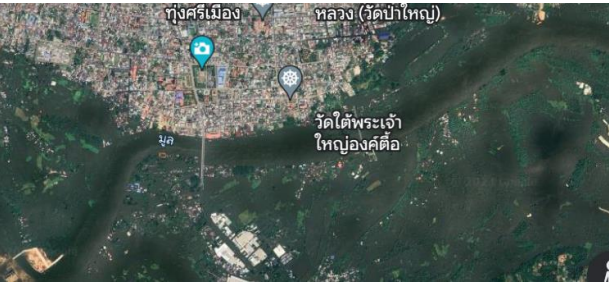
Google map



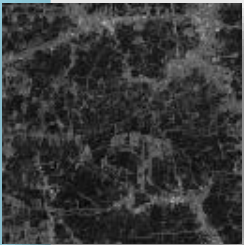
ชุมชนรอบแม่น้ำมูล
จังหวัดอุบลราชธานี



Google map



น้ำท่วมขัง
ชุมชนอำเภอผักไห่
จังหวัดพระนครศรีอยุธยา



Google map



ชุมชนอำเภอกำแพงแสน
จังหวัดนครปฐม



Google map



2015	2016	2017	2018	2019	2020	2021	2022
		ท่วม	ท่วม	ท่วม		ท่วม	ท่วม
		ท่วม		ท่วม			ท่วม
		ท่วม	ท่วม	ท่วม	ท่วม	ท่วม	ท่วม
						ท่วม	ท่วม

3. ผลลัพธ์จากการทำนายโดยใช้โมเดล 3DDR U-Net (Prediction)

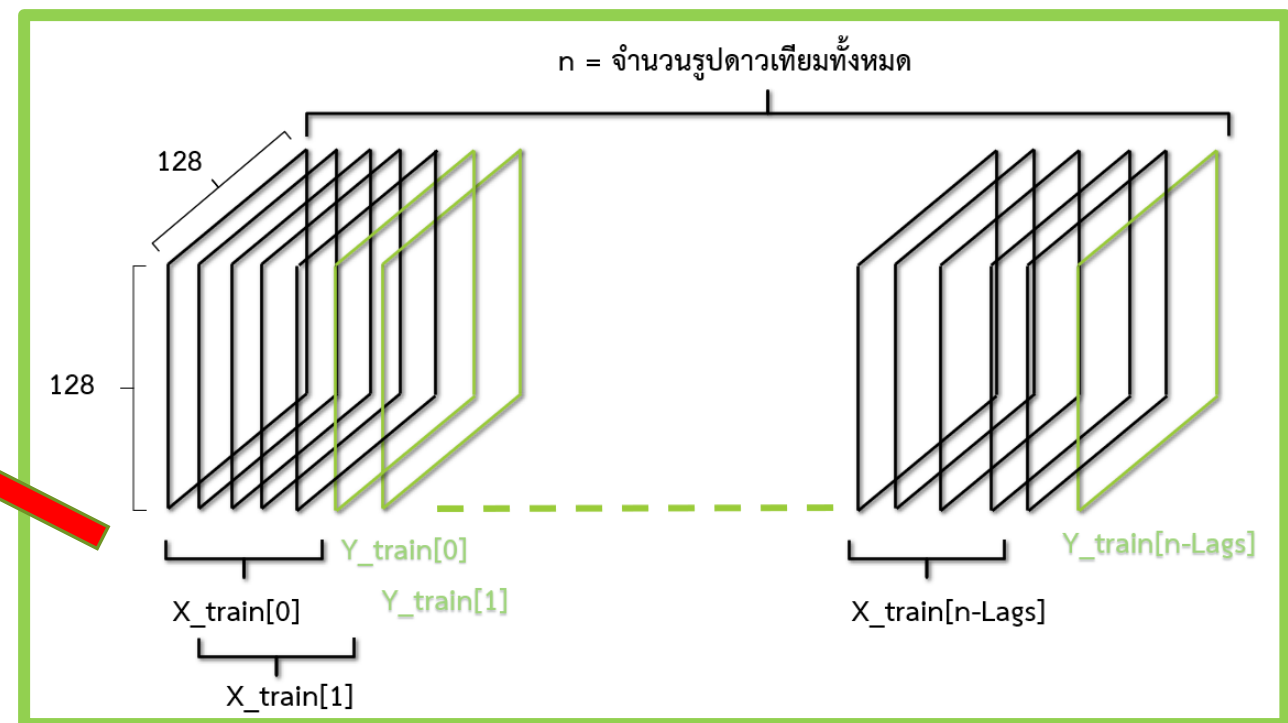
1. Code ที่ใช้ในการจัด Data input

```
IMG_WIDTH = 128
IMG_HEIGHT = 128
IMG_CHANNELS = 1
LAGS = 5
# province = "Ubon"
# province = "Nakhonsawan"
# province = "Phranakorn"
province = "Metropolitan"
TRAIN_PATH = 'X_train/128/Data_%s_128/'%(province)
train_ids = natsorted(os.listdir(TRAIN_PATH))
X_train = np.zeros((len(train_ids),LAGS, IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
Y_train = np.zeros((len(train_ids)-LAGS,1, IMG_HEIGHT, IMG_WIDTH,1))
```

1.1 สร้างตัวแปร X_train และ Y_train เพื่อเก็บรูป

```
import cv2
for n, id_ in tqdm(enumerate(train_ids), total=len(train_ids)):
    path = TRAIN_PATH + id_
    img = imread(path,as_gray= True)
    # minVal = np.min(img_x)
    # maxVal = np.max(img_x)
    # img = normalize(img_x, maxVal, minVal)
    img = np.expand_dims(img, axis=-1)
    if n >= LAGS:
        Y_train[n-LAGS][0] = img
        # Y_train[n-3][0] = img
        for i in range(min(n+1, LAGS)):
            X_train[n-i][i] = img
```

วิธีการจัด X_train และ Y_train



Code Model 3DDR U-Net

```
def UNet_3DDR():
    lags = 5
    filters = 4
    dropout = 0.5 #0.5
    kernel_init=tf.keras.initializers.GlorotUniform(seed=50)
    features_output = 1
    inputs = Input(shape = (lags, 128, 128, 1))
    conv1 = Conv3D(filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(inputs)
    conv1 = Conv3D(filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv1)
    pool1 = MaxPool3D(pool_size=(1, 2, 2))(conv1)

    conv2 = Conv3D(2*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(pool1)
    conv2 = Conv3D(2*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv2)
    pool2 = MaxPool3D(pool_size=(1, 2, 2))(conv2)

    conv3 = Conv3D(4*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(pool2)
    conv3 = Conv3D(4*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv3)
    pool3 = MaxPool3D(pool_size=(1, 2, 2))(conv3)

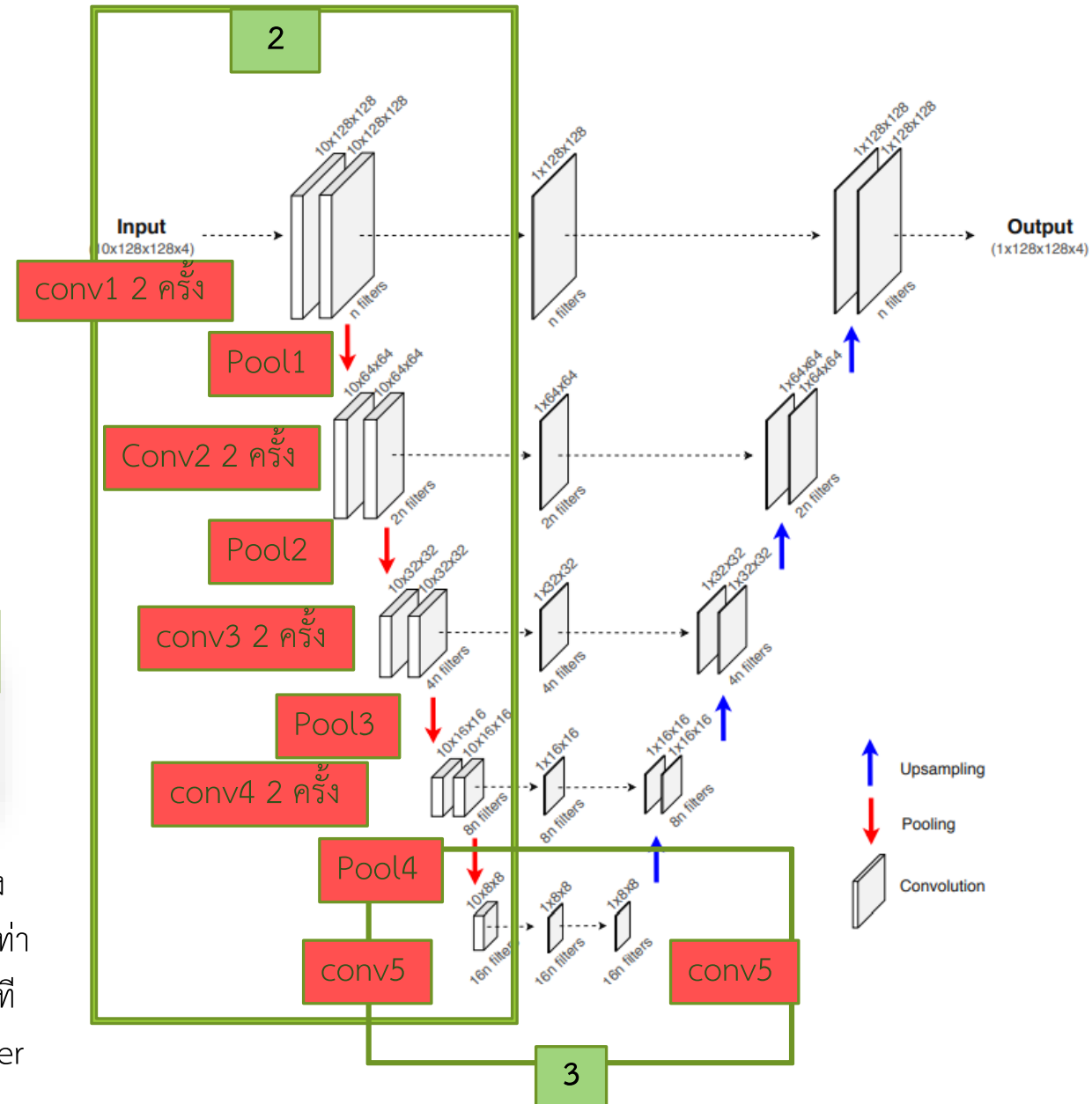
    conv4 = Conv3D(8*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(pool3)
    conv4 = Conv3D(8*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv4)
    drop4 = Dropout(dropout)(conv4)
```

1.2 Encoder

```
#--- Bottleneck part ---#
pool4 = MaxPool3D(pool_size=(1, 2, 2))(drop4)
conv5 = Conv3D(16*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(pool4)
compressLags = Conv3D(16*filters, (lags,1,1), activation = 'relu', padding = 'valid')(conv5)
conv5 = Conv3D(16*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(compressLags)
drop5 = Dropout(dropout)(conv5)
```

1.3 Bottleneck

- แต่ละชั้นของการ Encoder หรือเรียกว่าการ Pooling จะมีการ Convolution 2 ครั้ง
- เมื่อ Pool ลงไปแต่ละชั้นจะใช้ Filter เท่ากับ 4 โดยจะขยาย filter ไปเรื่อยๆเป็น 2 เท่า
- เมื่อถึงชั้น Bottleneck จะมี Convolution เพียง 1 ครั้งและ Skip connection ทันทีเป็นชั้นที่เปลี่ยนจาก kernel size (3x3) เป็น (lags,1,1) เพื่อลดขนาด Channel ของ layer



- Part ของการ Decoder จะคล้ายกับ Part Encoder คือ Convolution แต่ละชั้น 2 ครั้ง
- ใน Code ของ Part Decoder จะมี Compresslags หรือเรียกว่าส่วน Skip connection โดยการ Convolution โดยใช้ kernel size เป็น (lags,1,1) พร้อม Merge layer จาก Encoder เข้ากับ layer ของ Decoder
- เมื่อ Upsampling ในแต่ละชั้นจะมีการขยาย Filter เป็นครั้ง 2 เท่า
- เมื่อถึงครบ Convolution 9 จะเข้าสู่ Final layer โดย Conv10 จะยุบ layer รวมกัน

```
#--- Expanding part / decoder ---#
up6 = Conv3D(8*filters, 2, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(UpSampling3D(size = (1,2,2))(drop5))
compresslags = Conv3D(8*filters, (lags,1,1),activation = 'relu', padding = 'valid')(drop4)
merge6 = concatenate([compresslags,up6], axis = -1)
conv6 = Conv3D(8*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(merge6)
conv6 = Conv3D(8*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv6)

up7 = Conv3D(4*filters, 2, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(UpSampling3D(size = ([1,2,2]))(conv6))
compresslags = Conv3D(4*filters, (lags,1,1),activation = 'relu', padding = 'valid')(conv3)
merge7 = concatenate([compresslags,up7], axis = -1)
conv7 = Conv3D(4*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(merge7)
conv7 = Conv3D(4*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv7)

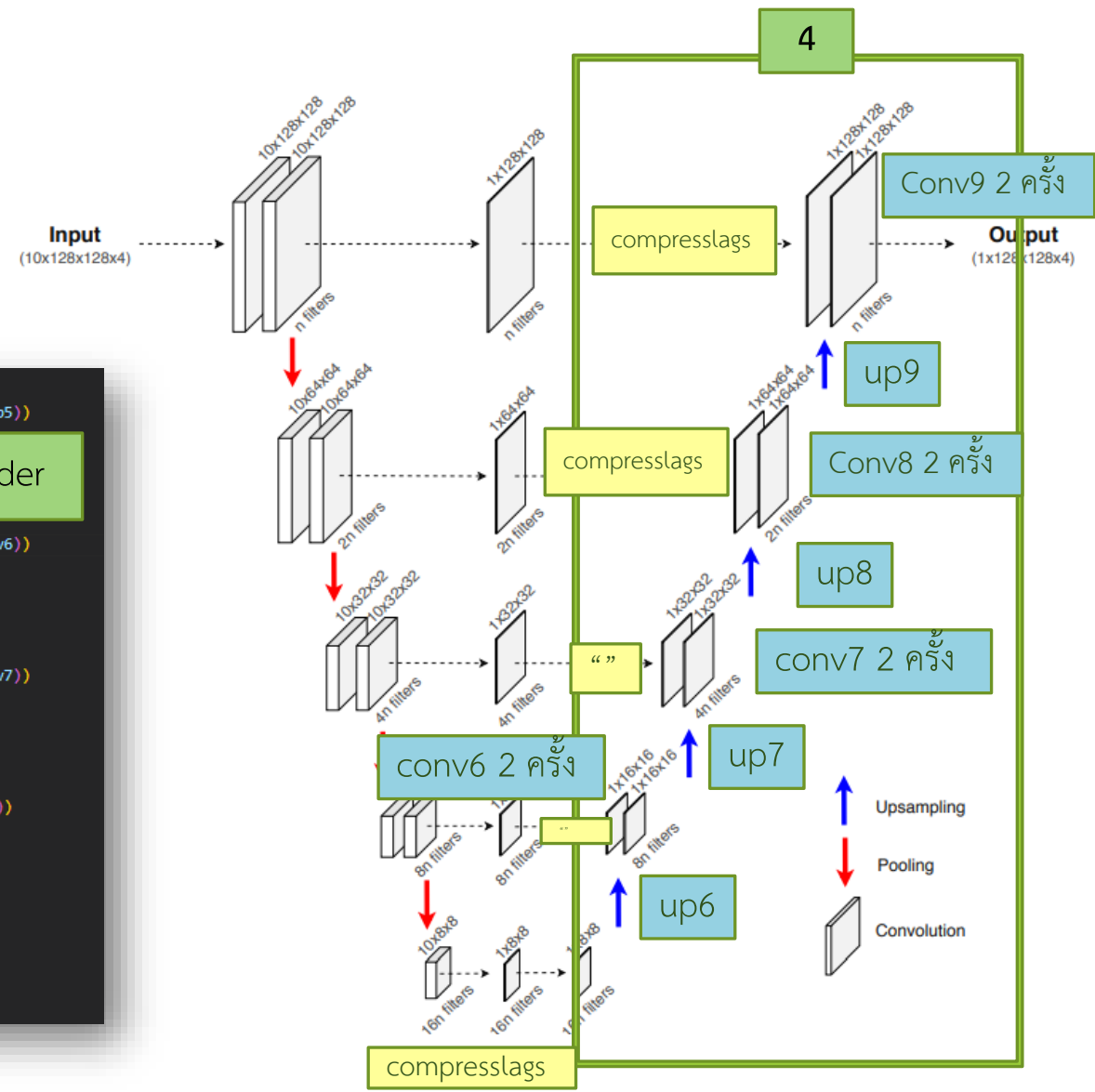
up8 = Conv3D(2*filters, 2, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(UpSampling3D(size = (1,2,2))(conv7))
compresslags = Conv3D(2*filters, (lags,1,1),activation = 'relu', padding = 'valid')(conv2)
merge8 = concatenate([compresslags,up8], axis = -1)
conv8 = Conv3D(2*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(merge8)
conv8 = Conv3D(2*filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv8)

up9 = Conv3D(filters, 2, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(UpSampling3D(size = (1,2,2))(conv8))
compresslags = Conv3D(filters, (lags,1,1),activation = 'relu', padding = 'valid')(conv1)
merge9 = concatenate([compresslags,up9], axis = -1)
conv9 = Conv3D(filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(merge9)
conv9 = Conv3D(filters, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv9)
conv9 = Conv3D(2*features_output, 3, activation = 'relu', padding = 'same', kernel_initializer = kernel_init)(conv9)

conv10 = Conv3D(features_output, 1, activation = 'relu')(conv9) #Reduce last dimension

return Model(inputs = inputs, outputs = conv10)
```

1.4 Decoder



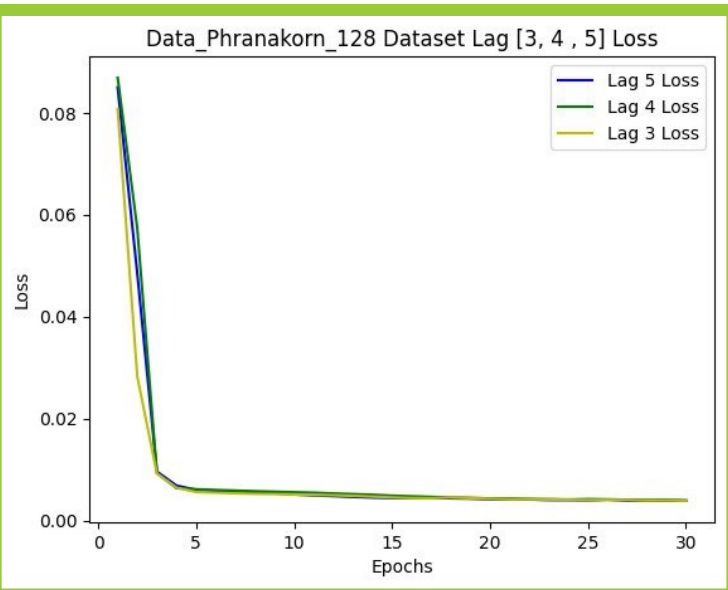
Code ที่ใช้ในการ train model

1.5 Train model

```
from tensorflow.keras.models import load_model
from tensorflow.keras.callbacks import ModelCheckpoint
model = UNet_3DDR()
filepath="model_%s_5_deno.hdf5"%(province)
# model = load_model(filepath, compile=False)
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, save_weights_only=False, mode='min')
callbacks = [checkpoint]
model.summary()
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
history1 = model.fit(X_train, Y_train, epochs=30)
# history2 = model.fit(X_train, Y_train, epochs=30)
# history3 = model.fit(X_train, Y_train, epochs=30)
# history4 = model.fit(X_train, Y_train, epochs=30)
```

```
=====
Total params: 380,833
Trainable params: 380,833
Non-trainable params: 0
=====
```

```
Epoch 1/30
12/12 [=====] - 62s 5s/step - loss: 0.0741
Epoch 2/30
12/12 [=====] - 60s 5s/step - loss: 0.0515
Epoch 3/30
12/12 [=====] - 60s 5s/step - loss: 0.0111
Epoch 4/30
```

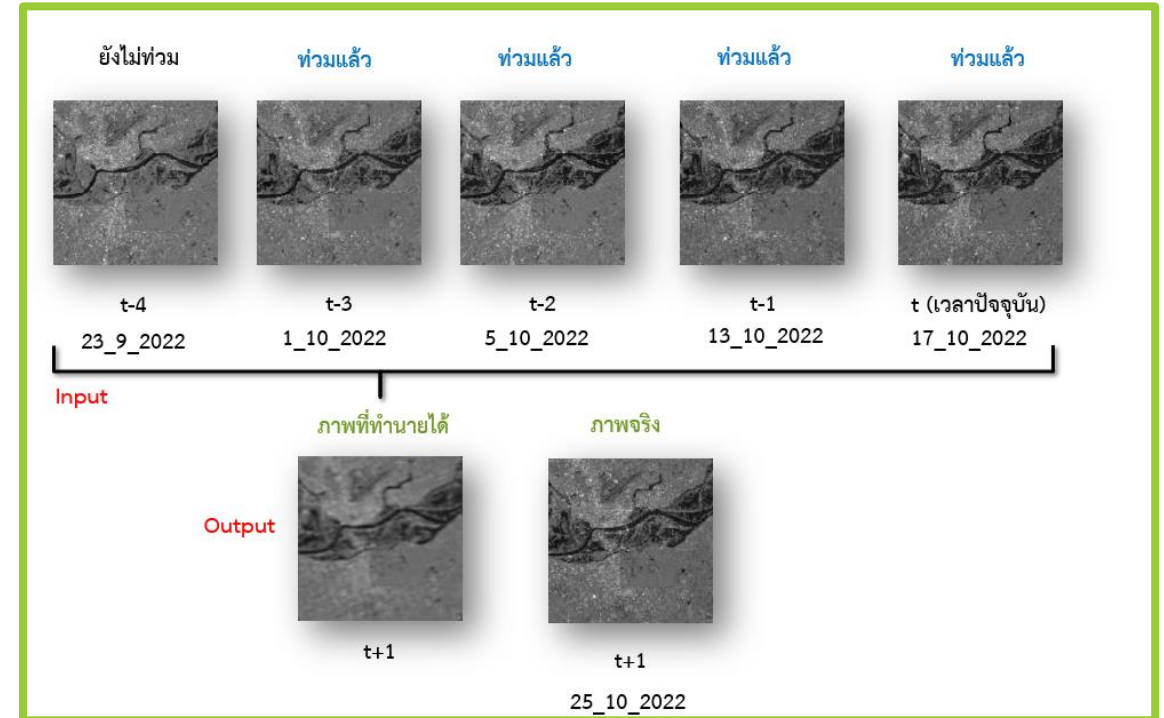
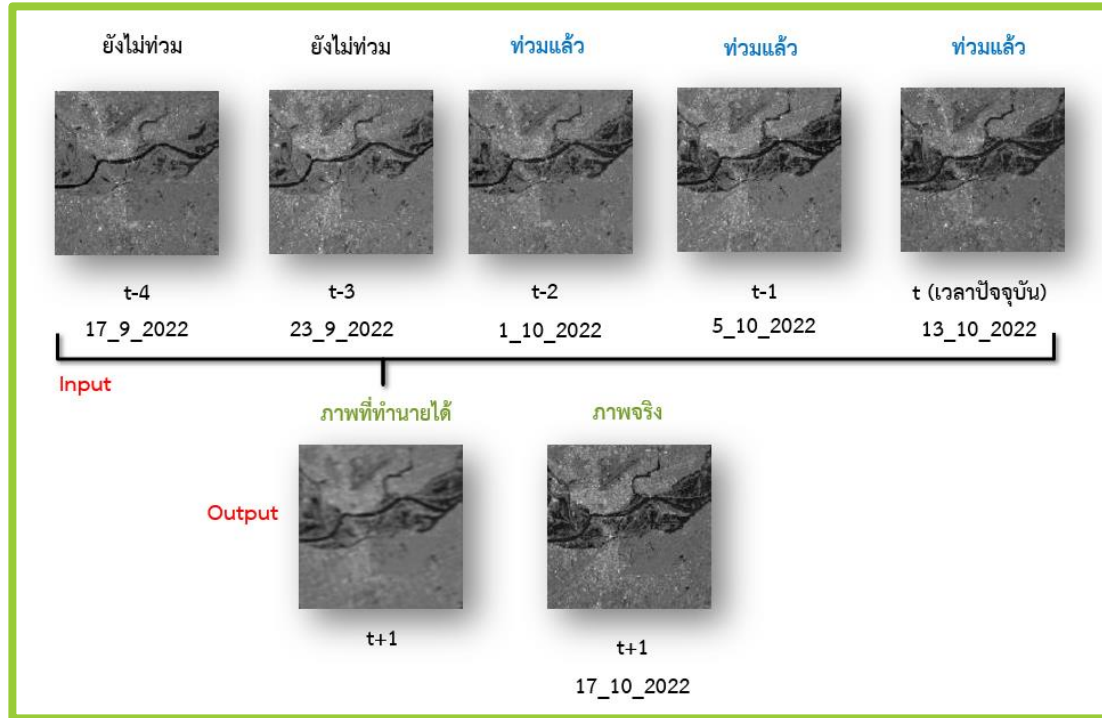


- กราฟ Training loss ของจังหวัดพระนครศรีอยุธยา จะพบว่า loss จะใกล้ล้นตัวที่ epoch ประมาณ 25 และที่Lag 3 , Lag 4 และ Lag 5 เมื่อถึงจุดล้นตัวนั้นมีการเรียนรู้ของ model มีค่าใกล้เคียงกัน

```
from skimage.metrics import peak_signal_noise_ratio
from skimage.metrics import structural_similarity as ssim
psnr = peak_signal_noise_ratio(img_orig, img_predict)
mse = np.mean((img_orig - img_predict) ** 2)
ssim_score = ssim(img_orig, img_predict, multichannel=True)
print(ssim_score)
print(mse)
print(psnr)
```

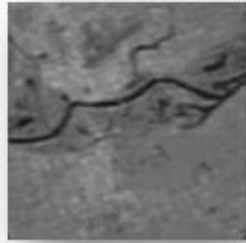
1.6 เครื่องมือวัดรูปทำนายและรูปจริง

2. ความสามารถในการทำนายของโมเดล 3DDR U-Net

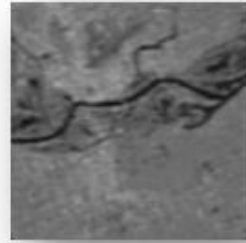


- ตัวอย่างการ Input ภาพน้ำท่วมแล้ว 3 และ 4 ภาพเข้า Input โดยใช้ Lag เท่ากับ 5 ภาพของจังหวัดอุบลราชธานี
- พบว่าการใส่ Input แล้ว 3 ภาพนั้นเมื่อเทียบกับน้ำท่วมจริงแล้วแบบ 4 ภาพนั้นมีความใกล้เคียงมากกว่า

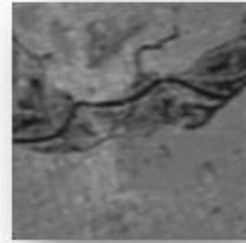
ภาพที่ทำนายได้



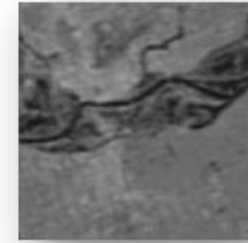
Input น้ำท่วมแล้ว 1 ภาพ



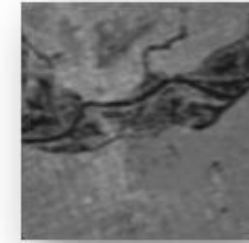
2 ภาพ



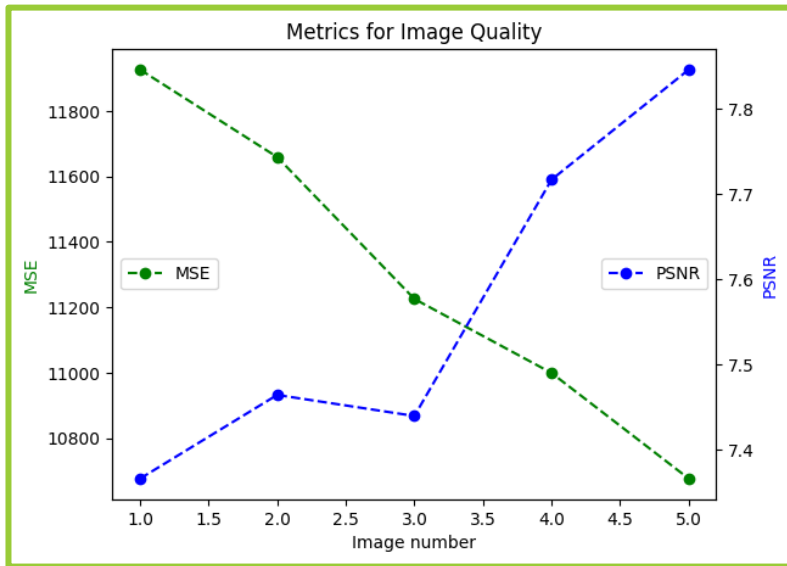
3 ภาพ



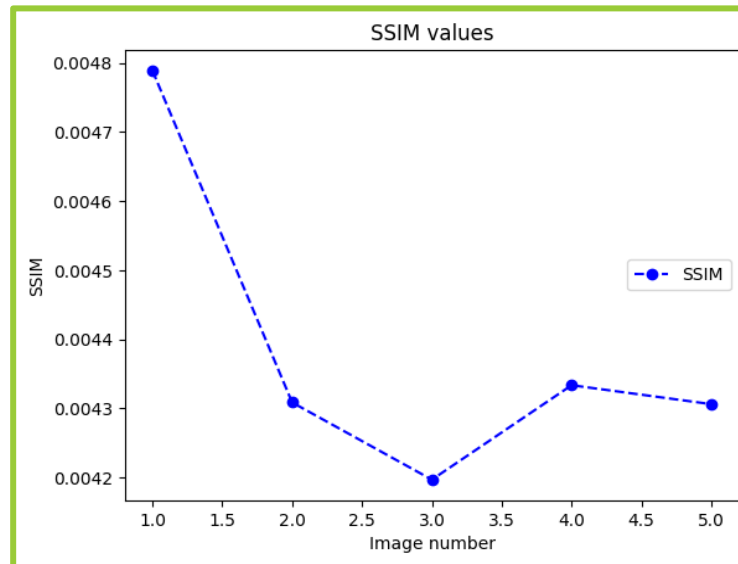
4 ภาพ



5 ภาพ



กราฟ MSE และ PSNR



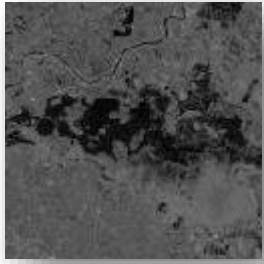
กราฟ MSE และ PSNR

- กราฟ MSE, PSNR และ SSIM ของการใส่ Input น้ำท่วมแต่ละภาพ โดยพบว่าที่ 5 ภาพนั้น ตัวโมเดลมีความแม่นยำมากที่สุดจาก MSE น้อยสุด, PSNR มากสุด และ SSIM ต่ำเกือบที่สุด

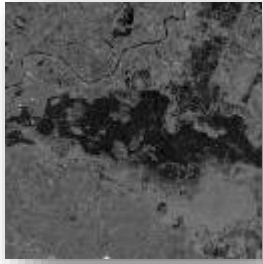
3. การทำนายน้ำท่วมจาก Unseen Data

- น้ำล้นตลิ่ง
- น้ำท่วมขัง

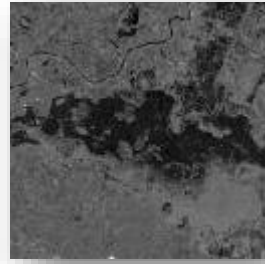
• บึงบอระเพ็ด จังหวัดนครสวรรค์



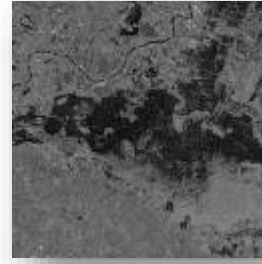
24 กรกฎาคม 2017



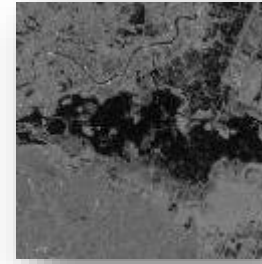
1 สิงหาคม 2017



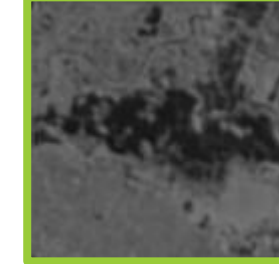
5 สิงหาคม 2017



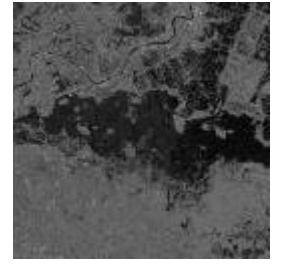
11 สิงหาคม 2017



20 สิงหาคม 2017



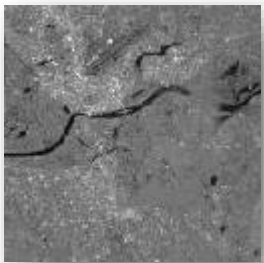
รูปที่ทำนายได้



รูปน้ำท่วมจริง

1 กันยายน 2017

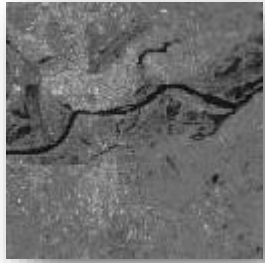
• แม่น้ำมูล จังหวัดอุบลราชธานี



22 กรกฎาคม 2017



1 สิงหาคม 2017



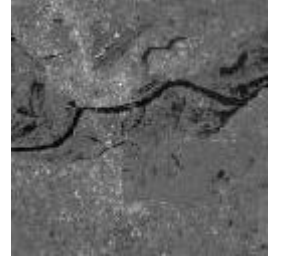
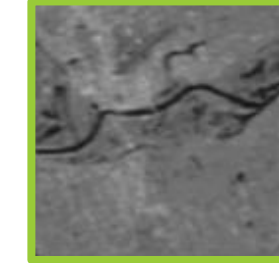
6 สิงหาคม 2017



13 สิงหาคม 2017



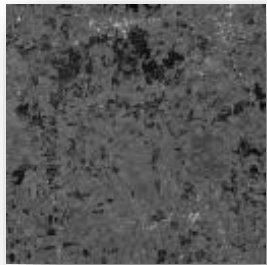
20 สิงหาคม 2017



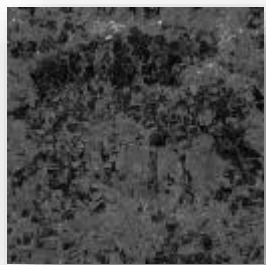
23 สิงหาคม 2017

ผลจากการวัดค่าแต่ละจังหวัด	MSE	PSNR (dB)	SSIM
จังหวัดนครสวรรค์	6861	9.77	5.46e-3
จังหวัดอุบลราชธานี	12180	7.27	4.54e-3

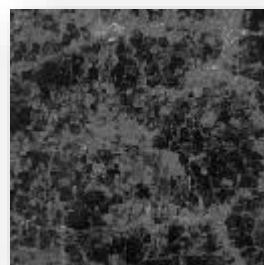
อำเภอผักไห่ จังหวัดพระนครศรีอยุธยา



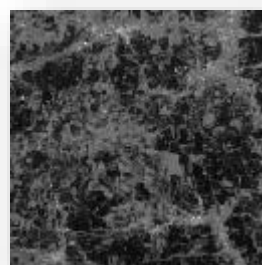
24 กันยายน 2016



28 กันยายน 2016



10 ตุลาคม 2016



18 ตุลาคม 2016



22 ตุลาคม 2016



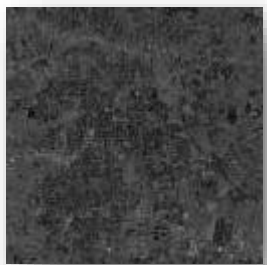
รูปที่ทำนายได้



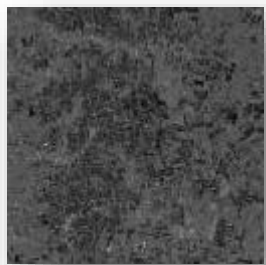
รูปน้ำท่วมจริง

3 พฤศจิกายน 2016

อำเภอกำแพงแสน จังหวัดนครปฐม



10 ตุลาคม 2016



18 ตุลาคม 2016



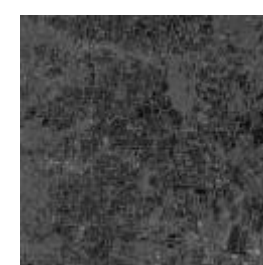
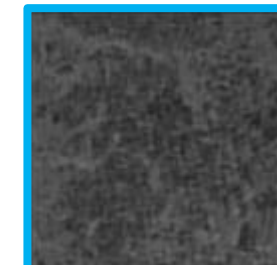
22 ตุลาคม 2016



3 พฤศจิกายน 2016



12 พฤศจิกายน 2016



15 พฤศจิกายน 2016

ผลจากการวัดค่าแต่ละจังหวัด	MSE	PSNR (dB)	SSIM
จังหวัดพระนครศรีอยุธยา	4991	11.14	5.38e-3
จังหวัดนครปฐม	5217	10.95	3.68e-3

วิเคราะห์ผลจากการดำเนินงาน

ผลการวัดค่าแต่ละจังหวัด	MSE	PSNR (dB)	SSIM
จังหวัดนครสวรรค์	6861	9.77	5.46e-3
จังหวัดอุบลราชธานี	12180	7.27	4.54e-3
จังหวัดพระนครศรีอยุธยา	4991	11.14	5.38e-3
จังหวัดนครปฐม	5217	10.95	3.68e-3

น้ำล้นตลิ่ง

น้ำท่วมขัง

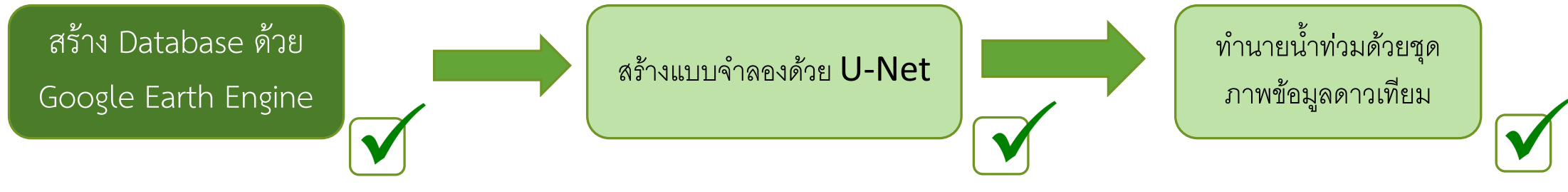
แม่น้ำน้อยที่สุด

แม่น้ำมากที่สุด

- จากตาราง Heatmap พบว่าจังหวัดนครศรีอยุธยาเป็นจังหวัดที่โมเดลมีการทำนายได้แม่นยำมากที่สุดจาก Unseen data ซึ่งสอดคล้องกับจำนวนข้อมูลของน้ำท่วมที่จังหวัดพระนครศรีอยุธยานั้นเป็นจังหวัดที่ได้รับผลกระทบจากน้ำท่วมมากที่สุดในไทย
- การทำนายแบบประเภทน้ำท่วมขังและน้ำล้นตลิ่งจากตาราง Heatmap จะพบว่า การทำนายน้ำท่วมแบบประเภทน้ำขังนั้น มีความแม่นยำมากกว่าการท่วมแบบล้นตลิ่งอย่างเห็นได้ชัด
- เครื่องมือวัดการทำนายจาก SSIM มีค่าไม่สอดคล้องกับ MSE และ PSNR เนื่องจาก SSIM นั้นวัดจากค่าความเข้มแสงของ Pixel จากการสังเกตพบว่าจังหวัดที่มีน้ำน้อยที่สุดจะเป็นจังหวัดที่ SSIM มีค่าความแม่นยำมากที่สุด เช่นจังหวัดนครปฐมและจังหวัดอุบล ที่รูปการทำนายนั้นมีน้ำท่วมไม่เยอะมาก

สรุปผลและข้อเสนอแนะ (Conclusion)

สรุปผลการดำเนินงานของโครงการ



ข้อเสนอแนะ

- โมเดลจะสามารถ tren ได้อย่างแม่นยำมากขึ้น เมื่อเทคโนโลยีของดาวเทียมมีการเจริญก้าวหน้ามากขึ้น ปัจจุบัน วงโคจรของดาวเทียม Sentinel-1 นั้นมีช่วงระยะเวลาอยู่ที่ 4-16 วันและยังถ่ายภาพได้ไม่ครบทั้งประเทศไทย
- ปรับปรุงแก้ไขโมเดล U-Net เพื่อพัฒนาตัวโมเดลให้มีประสิทธิภาพเหมาะสมเข้ากับตัวข้อมูลมากขึ้น