


รายงานโครงงานวิศวกรรมไฟฟ้า วิชา 2102499

การวิเคราะห์ภัยพิบัติน้ำท่วม
Flood disaster analysis

นายกฤษฎชัย ชาวเมืองชัย เลขประจำตัว 6230011221
อาจารย์ที่ปรึกษา ผศ.ดร.วิทยากร อัครวิเศษ

ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

ลงชื่ออาจารย์ที่ปรึกษาหลัก  () วันที่ 9 พค 2566	ลงชื่ออาจารย์ที่ปรึกษาร่วม (ถ้ามี) _____ () วันที่ _____	ลงชื่อตัวแทนบริษัท (เฉพาะนิสิตใน โปรแกรมความเชื่อมโยงอุตสาหกรรม) _____ () วันที่ _____
--	--	---

บทคัดย่อ

ในปัจจุบันนี้ Deep learning และ Big data มีการพัฒนาประสิทธิภาพอย่างรวดเร็วและจากปัญหาน้ำท่วมที่อยู่กับคนไทยมาอย่างยาวนานในโครงการนี้นำเสนอการทำนายภาพน้ำท่วมโดยการเก็บภาพดาวเทียม Copernicus จาก Google Earth Engine ตั้งแต่ปี ค.ศ. 2015 ถึงปี ค.ศ. 2022 และจะใช้โมเดล 3D Dimension reducer U-Net (3DDR U-Net) ทำนายภาพน้ำท่วมโดยใช้ภาพดาวเทียมย้อนหลัง 5 ภาพ ซึ่งจากผลลัพธ์ของรูปที่ทำนายได้เทียบกับรูปจริงพบว่าการทำนายน้ำท่วมแบบประเภทน้ำท่วมขังนั้นมีประสิทธิภาพมากกว่าการทำนายน้ำท่วมแบบล้นตลิ่ง ที่ Mean square error (MSE) เท่ากับ 5104, Peak signal-to-noise ratio (PSNR) มีค่าเท่ากับ 11.18 dB และ Structural similarity index measure (SSIM) เท่ากับ $5.00e-3$ สรุปผลจากโครงการนี้พบว่า การเก็บภาพดาวเทียมจะทำได้ดีในภูมิภาคกลางและตะวันออกเฉียงเหนือและการทำนายน้ำท่วมนั้นจะมีประสิทธิภาพขึ้นอยู่กับจำนวนภาพน้ำท่วมที่ใช้เทรน

คำสำคัญ: Google Earth Engine, Copernicus, 3D Dimension Reducer U-Net, Mean square error, Peak signal-to-noise ratio, Structural similarity index measure

Abstract

In recent years, the rapid development of Deep Learning and Big Data has presented opportunities for advancing flood prediction. As a longstanding problem for Thai people, this project uses Copernicus satellite images collected from the Google Earth Engine between 2015 and 2022 to predict flooding. The 3D Dimension Reducer U-Net (3DDR U-Net) model is employed to predict flood images using five time-lagged satellite images. The predicted images were compared with the actual images and revealed that the model was more efficient at predicting groundwater flooding compared to river flooding. The results showed a mean square error (MSE) of 5104, a peak signal-to-noise ratio (PSNR) of 11.18 dB, and a structural similarity index measure (SSIM) of $5.00e-3$. Overall, the study found that satellite imagery works well in central and northeastern Thailand, and flood prediction is effective depending on the number of flood images used in training.

Keywords: Google Earth Engine, Copernicus, 3D Dimension Reducer U-Net, Mean square error, Peak signal-to-noise ratio, Structural similarity index measure

สารบัญ

บทคัดย่อ.....	ก
Abstract	ก
สารบัญ	ข
1. บทนำ	1
1.1 ที่มาและความสำคัญของโครงการ	1
1.2 วัตถุประสงค์ของโครงการ.....	1
1.3 ขอบเขตของโครงการ	1
1.4 ผลลัพธ์ที่คาดหวังจากโครงการ	2
1.5 ขั้นตอนการดำเนินงาน.....	2
2. หลักการและทฤษฎีที่เกี่ยวข้อง.....	2
2.1 ปัญหาน้ำท่วมในประเทศไทย.....	2
2.2 ดาวเทียม Sentinel-1	6
2.3 โครงข่ายประสาทแบบคอนโวลูชัน (Convolution neural network).....	7
2.4 U-Net Model	9
2.5 เครื่องมือในการเปรียบเทียบรูปน้ำท่วมที่ทำนายได้และรูปจริง.....	10
3. ผลลัพธ์ของโครงการและการอภิปรายผล.....	12
3.1 ผลลัพธ์ที่ได้จากการ Visualization บน Google earth engine	12
3.2 ผลลัพธ์ที่ได้จากการ Collect data จาก Google Colab	13
3.3 ผลลัพธ์จากการเทรน 3D Dimension Reducer U-Net model	16
4. บทสรุป.....	20
4.1 สรุปผลการดำเนินการ	20
4.2 ปัญหา อุปสรรค และแนวทางแก้ไข	21
4.3 ข้อเสนอแนะ.....	21
5. กิตติกรรมประกาศ.....	21
6. เอกสารอ้างอิง.....	22
7. ภาคผนวก.....	23
7.1 ภาคผนวก ก. Code ภาษา JavaScript ที่สร้าง Visualization บน Google earth engine	23
7.2 ภาคผนวก ข. Code ที่ Collect data จาก Google earth engine บน Google Colab	23
7.3 ภาคผนวก ค. Code ที่ใช้ในการเทรนและทำนายจากโมเดล 3DDR U-Net.....	24

1. บทนำ

1.1 ที่มาและความสำคัญของโครงการ

ปัจจุบันน้ำท่วมถือเป็นภัยพิบัติธรรมชาติที่อยู่กับประเทศไทยมาอย่างยาวนาน ซึ่งสร้างผลกระทบให้กับเศรษฐกิจและชีวิตของคนในประเทศ ประเทศไทยนั้นประสบกับปัญหาน้ำท่วม สาเหตุส่วนใหญ่มีน้ำท่วมมาจากตำแหน่งที่ตั้ง ซึ่งพื้นที่จะเป็นบริเวณแบบราบแอ่งน้ำและอยู่ในเขตอิทธิพลของมรสุมของภูมิภาคเอเชียตะวันออกเฉียงใต้ โดยความรุนแรงนั้นจะแตกต่างกันไปทุกปี ช่วงปี 1989-2009 พบว่าความเสียหายที่เกิดจากน้ำท่วมนั้นมีมูลค่าสูงถึง 1 แสนล้านบาท และยอดผู้ได้รับความบาดเจ็บและเสียชีวิตมีมากถึง 1 หมื่นคน และปี 2011 เป็นปีที่ประเทศไทยนั้นได้รับความเสียหายจากน้ำท่วมเป็นพื้นที่ถึง 16 ล้านไร่และจากการประเมินของธนาคารโลกสูงถึง 1.44 ล้านล้านบาท

จากเหตุการณ์น้ำท่วมที่ผ่านมา จะเห็นว่าประเทศไทยนั้นประสบปัญหาภัยน้ำท่วมบ่อยรุนแรงและบ่อยครั้งมากขึ้น ดังนั้นการวิเคราะห์สถานการณ์น้ำท่วมล่วงหน้าจึงเป็นสิ่งสำคัญอย่างยิ่งเพื่อที่จะป้องกันลดความเสียหายของน้ำท่วมที่จะเกิดขึ้น เพราะการอพยพคนและทรัพย์สินล่วงหน้า จะทำให้ความเสียหายที่เกิดขึ้นจากน้ำท่วมนั้นบรรเทาลง การคาดการณ์น้ำท่วมนั้นมีหลากหลายวิธี ไม่ว่าจะเป็นการคาดการณ์จากพยากรณ์สภาพอากาศ พายุและมรสุม, การคาดการณ์จากระดับน้ำใกล้เคียง หรือการคาดการณ์จากดาวเทียม

โครงการนี้จึงทำการทำนายการเกิดน้ำท่วมจากดาวเทียมด้วยการเรียนรู้เชิงลึก (Deep learning) ซึ่งการคาดการณ์น้ำท่วมผ่านดาวเทียมเป็นวิธีหนึ่งที่สามารถทำนายน้ำท่วมได้ โดยจากการที่ดาวเทียมนั้นสามารถที่จะส่งสัญญาณและวิเคราะห์พื้นผิวนโลกออกมาเป็นรูปภาพ โดยที่รูปภาพนั้นจะบ่งบอกถึงพื้นดินและน้ำ เมื่อนำ Deep learning มาใช้จากการนำภาพของน้ำท่วมในอดีตมาวิเคราะห์และฝึกฝน จะทำให้การคาดการณ์น้ำท่วมนั้นมีความสะดวกและคล่องตัวสูงขึ้น ตามจำนวนภาพหรือความละเอียดของภาพนั้น

1.2 วัตถุประสงค์ของโครงการ

1. เพื่อสร้างแบบจำลองทำนายพื้นที่ ที่มีโอกาสเกิดน้ำท่วมแบบอัตโนมัติจากการวิเคราะห์สถิติของการเกิดน้ำท่วมในอดีต
2. เพื่อสามารถวิเคราะห์ปัญหาของสภาพภูมิศาสตร์ของพื้นที่ที่เกิดน้ำท่วมบ่อยครั้ง

1.3 ขอบเขตของโครงการ

1. ศึกษา Application program interface (API) ของการเรียกใช้ภาพถ่ายดาวเทียม
2. ศึกษาโปรแกรม Python หรือโปรแกรม MATLAB ในการสร้างแบบจำลองการเรียนรู้เครื่อง (Machine Learning Model)
3. นำแบบจำลองการเรียนรู้เครื่องที่เรียนรู้แล้วมาทำนายน้ำท่วมกับพื้นที่ขนาด 10 กิโลเมตร x 10 กิโลเมตร หรือ Resolution ที่ละเอียดน้อยกว่านั้น

1.4 ผลลัพธ์ที่คาดหวังจากโครงการ

ชุดซอฟต์แวร์ที่รับชุดของภาพดาวเทียม และทำนายบริเวณที่น้ำท่วมจะเกิดขึ้น

1.5 ขั้นตอนการดำเนินงาน

1. ศึกษาและค้นหาโปรแกรมหรือบทความวิจัย (Research) ที่เกี่ยวข้อง
2. เขียนโปรแกรมมิ่ง JavaScript บน Google earth engine เพื่อสังเกตพื้นที่น้ำท่วม
3. เขียนโปรแกรมมิ่ง Python เพื่อเก็บภาพดาวเทียมแต่ละภูมิภาคผ่าน Google earth engine
4. นำชุดภาพดาวเทียมเข้าไปเทรน (Train) ใน 3DDR U-Net model
5. เปรียบเทียบรูปน้ำท่วมที่ทำนายได้เทียบกับรูปน้ำท่วมจริง

2. หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 ปัญหาน้ำท่วมในประเทศไทย

ประเภทของน้ำท่วมในประเทศไทย

1. อุทกภัยจากริมฝั่งแม่น้ำ (น้ำล้นตลิ่ง) การเกิดอุทกภัยชนิดนี้เกิดขึ้นจากการสะสมที่น้ำจากฝนของพายุมรสุมที่เกิดการไหลลงแม่น้ำเป็นจำนวนมากทำให้ปากแม่น้ำนั้นระบายแม่น้ำไม่ทันส่งผลให้ชุมชนบ้านเรือนบริเวณโดยรอบแม่น้ำได้รับความเสียหาย
2. อุทกภัยจากน้ำป่าไหลหลาก การเกิดอุทกภัยชนิดนี้เกิดขึ้นที่บริเวณหุบเขาที่มีความชันลาดชันเมื่อเกิดฝนตกจะมีน้ำไหลลงมา และจะรุนแรงมากขึ้นหากมีการตัดไม้ทำลายป่า โดยอุทกภัยชนิดนี้มีความรุนแรงสูงอาจสร้างภัยดินถล่มต่อมาได้
3. อุทกภัยจากน้ำท่วมขัง การเกิดอุทกภัยชนิดนี้เกิดขึ้นจากการสะสมของปริมาณน้ำจำนวนมากและพื้นดินนั้นมีลักษณะเป็นแอ่งเก็บน้ำ ซึ่งความรุนแรงนั้นจะขึ้นอยู่กับระบบระบายน้ำ อุทกภัยชนิดนี้มักเกิดขึ้นในชุมชนเมืองใหญ่เมื่อเกิดฝนตกหนักติดต่อกันเป็นเวลาหลายวัน [1]



รูปที่ 1 น้ำล้นตลิ่ง, น้ำป่าไหลหลาก และน้ำท่วมขัง ตามลำดับ [1]

ภูมิศาสตร์น้ำท่วมของประเทศไทย

ประเทศไทยแบ่งตามสภาพภูมิประเทศจะแบ่งออกได้เป็น 6 ภูมิภาค ซึ่งภูมิภาคทั้ง 6 นี้จะมีความแตกต่างกันโดยมีเอกลักษณ์ของตนเองในลักษณะของธรรมชาติ ทรัพยากรและภูมิศาสตร์ โดยแต่ละภูมิภาคจะมีเขื่อนและแม่น้ำที่ไหลผ่านดังรูปที่ 2 โดย



รูปที่ 2 รูปแม่น้ำภูมิภาคเหนือ, ภาคตะวันออก และภาคกลาง (ที่มา: sunshine-project.org/แม่น้ำหลัก)

ภาคเหนือ มีลักษณะภูมิประเทศเป็นแบบภูเขาสูงสลับกับหุบเขาและมีพื้นที่ราบสูงเป็นส่วนใหญ่ เป็นภูมิภาคที่ได้รับผลกระทบจากน้ำท่วมไม่มากนัก เมื่อเวลาเกิดน้ำท่วมจะเป็นน้ำท่วมประเภทน้ำป่าไหลหลากซึ่งก่อให้เกิดดินถล่ม ภาคเหนือจะมีแม่น้ำที่สำคัญไหลผ่านคือแม่น้ำปิงและแม่น้ำวัง เมื่อเกิดฝนตกหนักหรือพายุเข้ามวลน้ำของแม่น้ำทั้งสองสายจะไหลระบายลงมาสู่ภาคกลางตอนบน

ภาคตะวันออกเฉียงเหนือหรือภาคอีสาน มีลักษณะภูมิประเทศเป็นแบบเนินเขาเตี้ยผสมกับพื้นที่ราบลุ่มแม่น้ำและมีแม่น้ำสำคัญคือแม่น้ำมูลและแม่น้ำชี เมื่อเกิดพายุฝนตกหนักที่บริเวณภาคอีสานจะก่อให้เกิดน้ำท่วมแบบล้นตลิ่ง ส่งผลกระทบต่อชุมชนรอบบริเวณแม่น้ำเนื่องจากมวลน้ำจากฝนนั่นจะไหลผ่านแม่น้ำมูลและแม่น้ำชีไปยังแม่น้ำโขงที่ติดต่อกับฝั่งประเทศลาวและกัมพูชา

ภาคตะวันตก มีลักษณะภูมิประเทศคล้ายกับภาคเหนือคือมีทิวเขาสูงสลับกับหุบเขาแคบ มีแม่น้ำกลองไหลผ่าน ภาคตะวันตกเป็นที่ตั้งเขื่อนที่สำคัญของประเทศ คือเขื่อนศรีนครินทร์และเป็นภาคที่ได้รับผลกระทบจากน้ำท่วมน้อยที่สุดในทั้ง 6 ภูมิภาคเนื่องจากเป็นภูมิภาคที่ไม่ได้มีแม่น้ำไหลสำคัญเป็นทางผ่านมากนัก ไม่เหมือนกับภาคกลางที่น้ำที่มาจากภาคเหนือจะไหลผ่านเกือบทั้งภูมิภาค

ภาคกลางได้ถูกเรียกว่าเป็นอู่ข้าวอู่น้ำของประเทศไทยเนื่องจากมีลักษณะทางภูมิประเทศเป็นแบบที่ราบลุ่มเหมาะแก่การเพาะปลูกและมีแม่น้ำเจ้าพระยาเป็นแม่น้ำหลัก แต่เนื่องจากภาคกลางมีแม่น้ำสำคัญจากภาคเหนือไหลผ่านหลายแห่ง ภาคกลางจึงเป็นภาคที่ได้รับผลกระทบจากน้ำท่วมเยอะและบ่อยที่สุด ตัวอย่างเช่น จังหวัดพระนครศรีอยุธยาที่อยู่ภาคกลางตอนบนเป็นจังหวัดที่ได้รับผลกระทบจากน้ำท่วมเกือบทุกปีจากท่าเลที่ตั้ง ที่เป็นแหล่งรวมการไหลของแม่น้ำสำคัญของภาคเหนือ

ภาคตะวันออก มีลักษณะทางภูมิประเทศเป็นแบบที่ราบสูงสลับกับภูเขาเตี้ยมาก มีชายทะเลเรียบผ่าน และมีแม่น้ำสำคัญที่ไหลลงสู่อ่าวไทยคือแม่น้ำบางปะกง ภาคตะวันออกนั้นเป็นหนึ่งในภูมิภาคที่ได้รับผลกระทบจากน้ำท่วมบ่อยเนื่องจากอยู่ติดขอบชายทะเลทำให้ระบายน้ำได้ง่ายและรวดเร็ว



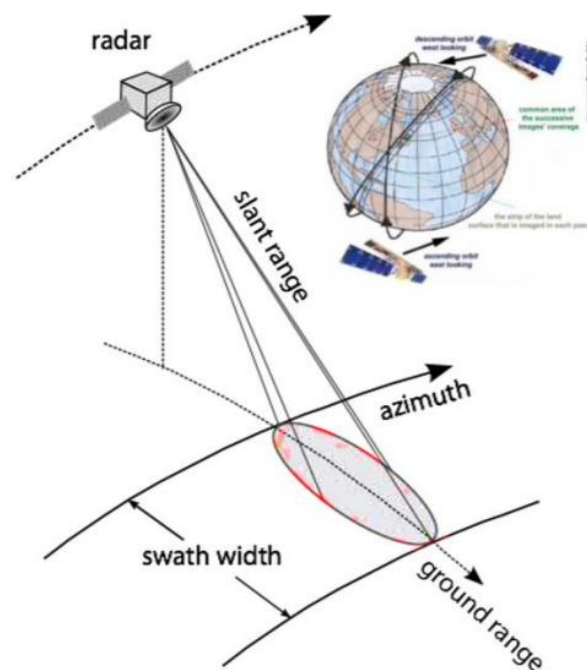
รูปที่ 3 รูปแม่น้ำภูมิภาคเหนือ, ภาคตะวันออก และภาคกลาง (ที่มา: slideplayer.in.th โดย: Satit UP)

ภาคใต้ มีลักษณะเป็นที่ราบและมีทิวเขาย้อมโดยมีทะเลขนานอยู่ 2 ด้านคือ ฝั่งตะวันออกคือทะเลอ่าวไทยและฝั่งตะวันตกคือทะเลอันดามัน ดังรูปที่ 3 ภาคใต้จะเกิดภัยพิบัติน้ำท่วม ในเวลาที่แตกต่างกับภูมิภาคตอนบนของประเทศไทย ปัญหา น้ำท่วมของภาคใต้จะมาจากฝนมรสุมหรือพายุที่พัดมาทางฝั่งทะเลอ่าวไทย เป็นส่วนใหญ่ผสมผสานกับการระบายน้ำที่ยากลำบากในตัวเมืองส่งผลให้ภาคใต้เป็นภูมิภาคที่ได้รับผลกระทบจากน้ำท่วมแม้ว่าจะอยู่ติดขอบชายทะเล

จากตารางที่ 1 เปรียบเทียบน้ำท่วมระหว่างเดือนกันยายนปี 2011 และเดือนกันยายนปี 2021 จะพบว่าภาคตะวันออกเฉียงเหนือและภาคกลางเป็นภาคที่มีพื้นที่น้ำท่วมต่อไร่มากที่สุดในประเทศไทย เนื่องจากเมื่อเกิดพายุมรสุม มวลน้ำฝนจำเป็นต้องระบายไหลออกสู่ทะเลอ่าวไทยและแม่น้ำโขง ส่งผลให้ชุมชนที่อาศัยบริเวณรอบแม่น้ำสำคัญได้รับผลกระทบจากการระบายน้ำที่ไม่ทันเวลา

ตารางที่ 1 ตารางการเทียบน้ำท่วม 2011 กับ 2021 (ที่มา Gistda.or.th)

ภูมิภาค	แม่น้ำสำคัญที่ไหลผ่าน	พื้นที่น้ำท่วม กันยายน 2011 (ไร่)		พื้นที่น้ำท่วม กันยายน 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,แม่น้ำ เจ้าพระยา,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, แม่น้ำหลัง สวน, แม่น้ำตรัง	65,581	0.40%	0	0%
รวม		15,996,150	100%	5,648,252	100%



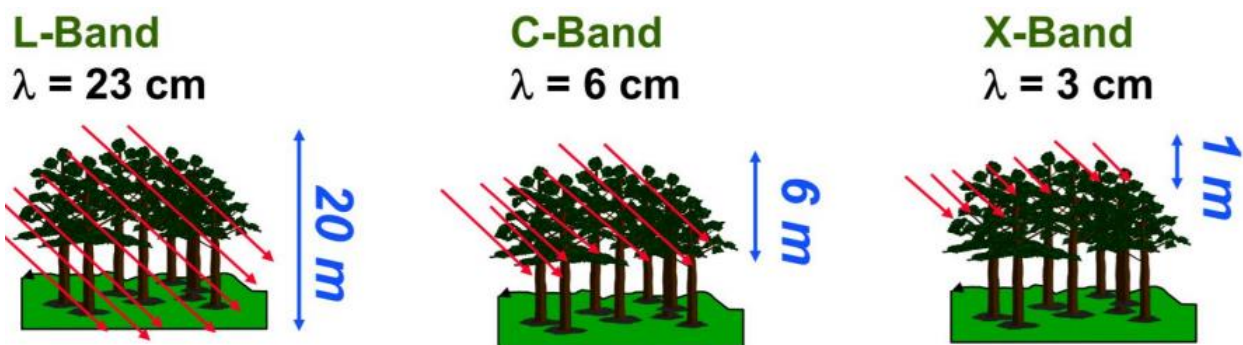
รูปที่ 4 วิธีการเก็บข้อมูลของ SAR [2]

2.2 ดาวเทียม Sentinel-1

ดาวเทียม Sentinel-1 จากโครงการ Copernicus นั้นเป็นดาวเทียมที่มีระบบการสำรวจระยะไกล (remote sensing) รูปแบบการรับส่งสัญญาณนั้นเป็นแบบปล่อยสัญญาณด้วยตนเอง (active remote sensing) ซึ่งข้อมูลที่ได้รับจากดาวเทียมนั้นจะเป็นข้อมูลแบบเรดาร์รับแสงสังเคราะห์หรือภาษาอังกฤษคือ Synthetic aperture radar (SAR) ซึ่ง SAR สามารถที่จะตรวจจับภาพพื้นดินได้ไม่ว่าจะเป็นเวลาเช้าหรือกลางคืน หรือแม้ว่าจะมีเมฆมาก แต่ผลลัพธ์ที่ตามมาคือ ความละเอียด (resolution) นั้นจะต่ำ ตัวอย่างการเก็บข้อมูลของ SAR ดังรูปที่ 4 [2]

แบนด์ (Band) และ โพลาไรเซชัน (Polarization) ของเรดาร์

1. Band คือช่วงความถี่ที่ดาวเทียมนั้นใช้ บ่งบอกถึงความสามารถในการในการทะลุลง โดยเมื่อ band ที่เลือกใช้ มีความยาวคลื่น (wavelength) มากก็ยิ่งทำให้ ความสามารถในการทะลุทะลวงนั้นสูง แต่ก็ก็จะตามมาด้วย ความถี่ (frequency) ที่ต่ำและ การลดทอนสัญญาณที่ต่ำ (free space loss) ซึ่งการเลือกใช้ band ที่ wavelength สูงนั้นเหมาะกับการต้องการตัดปัญหาเรื่อง หิมะหรือน้ำแข็งออก อย่างเช่น L-band ที่เป็นย่านความถี่ที่ใช้กับบริการสัญญาณโทรศัพท์มือถือผ่านทะเล, C-band เป็นย่านความถี่ที่ใช้ในโทรทัศน์และใช้ในการสื่อสารเป็นหลัก ซึ่งไม่ได้รับผลกระทบจากฝน



รูปที่ 5 ความสามารถของแบนด์ในการทะลุทะลวง [2]

2. Polarization คือทิศทางการแผ่กระจายของสนามแม่เหล็กไฟฟ้าของคลื่นแม่เหล็กไฟฟ้า ซึ่งจะมีการกระจายอยู่สองแบบคือ แนวตั้ง (vertical) และแนวนอน (horizontal) โดยระบบเรดาร์ของดาวเทียมนั้นสามารถที่จะส่งหรือรับสัญญาณของคลื่นแม่เหล็กไฟฟ้าได้แบ่งเป็น 4 รูปแบบแต่ละแบบนี้จะมีคุณสมบัติ และประโยชน์ที่แตกต่างกัน โดยแบ่งตามการส่งสัญญาณ และการรับสัญญาณคือ

- HH เมื่อส่งคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวนอน และรับคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวนอน
- VV เมื่อส่งคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวตั้ง และรับคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวตั้ง

- HV เมื่อส่งคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวนอน และรับคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวตั้ง
- VH เมื่อส่งคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวตั้ง และรับคลื่นแม่เหล็กในทิศทางกระจายผ่านแนวนอน [3]

การที่จะเลือกใช้ Polarization จำเป็นต้องเลือก Polarization ให้เหมาะสมกับการใช้งานโดยวัดจาก ค่า Backscatter คือค่าสะท้อนกลับของ Polarization โดยจากรูปที่ 6 จะเห็นว่า HV กับ VH นั้นไม่มีความต่างในการตรวจหาพื้นที่น้ำท่วม ส่วน VV และ HH นั้นมีศักยภาพที่เหมาะสมกว่าที่จะใช้ในการตรวจหาพื้นที่น้ำท่วม เนื่องจาก VV และ HH นั้นมีค่า Backscatter มาก ซึ่งเหมาะแก่การทะลุทะลวงน้ำ โดย VV นั้นมีช่วง Backscatter กว้าง เหมาะสำหรับการตรวจหาวัตถุที่จมอยู่ภายใต้น้ำ และ HH ที่จำกัดช่วง Backscatter น้อยกว่า เหมาะสำหรับการแยกระหว่างพื้นดินกับน้ำ [4]

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

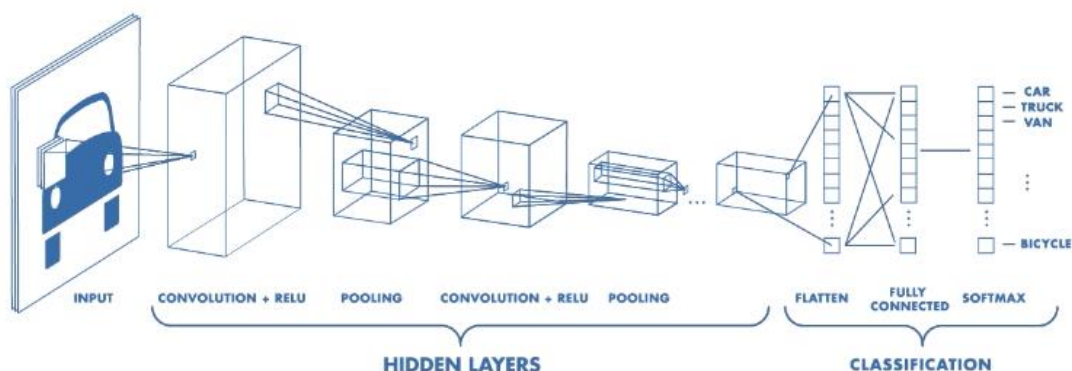
รูปที่ 6 ค่า Backscatter ของแต่ละ Polarization กับพื้นที่บนน้ำต่าง ๆ [3]

2.3 โครงข่ายประสาทแบบคอนโวลูชัน (Convolution neural network)

Convolution neural network (CNN) เป็นหนึ่งในประเภทของการเรียนรู้เชิงลึก (Deep learning) ซึ่งข้อมูลที่จะนำไปฝึก (train) คือข้อมูลแบบ Image data ข้อมูลจะเป็นรูปภาพนั้นจะประกอบด้วย 2 มิติ โดยหลักการจะจำลองการมองเห็นของมนุษย์โดยการประมวลผลไปที่ละพื้นที่ย่อย ๆ และขยับการประมวลผลไปเรื่อย ๆ เพื่อให้ครบทั้งรูป ตัวอย่างดังรูปที่ 7 โดยโครงสร้างพื้นฐานของตัว CNN นั้นมีหลายเลเยอร์ (layer) ซึ่งประกอบไปด้วย

1. ชั้นคอนโวลูชัน (Convolution layer) เป็น layer ที่อยู่ชั้นเริ่มแรกต้นต่อหลังจากชั้น input ชั้นที่ประกอบไปด้วยข้อมูลจากภาพที่นำเข้า และมี kernel หรือที่เรียกว่า filter โดยชั้นนี้จะทำการ convolution ระหว่าง filter กับ input โดยจุดประสงค์คือนำ feature ออกมาจากข้อมูล input ดังรูปที่ 7

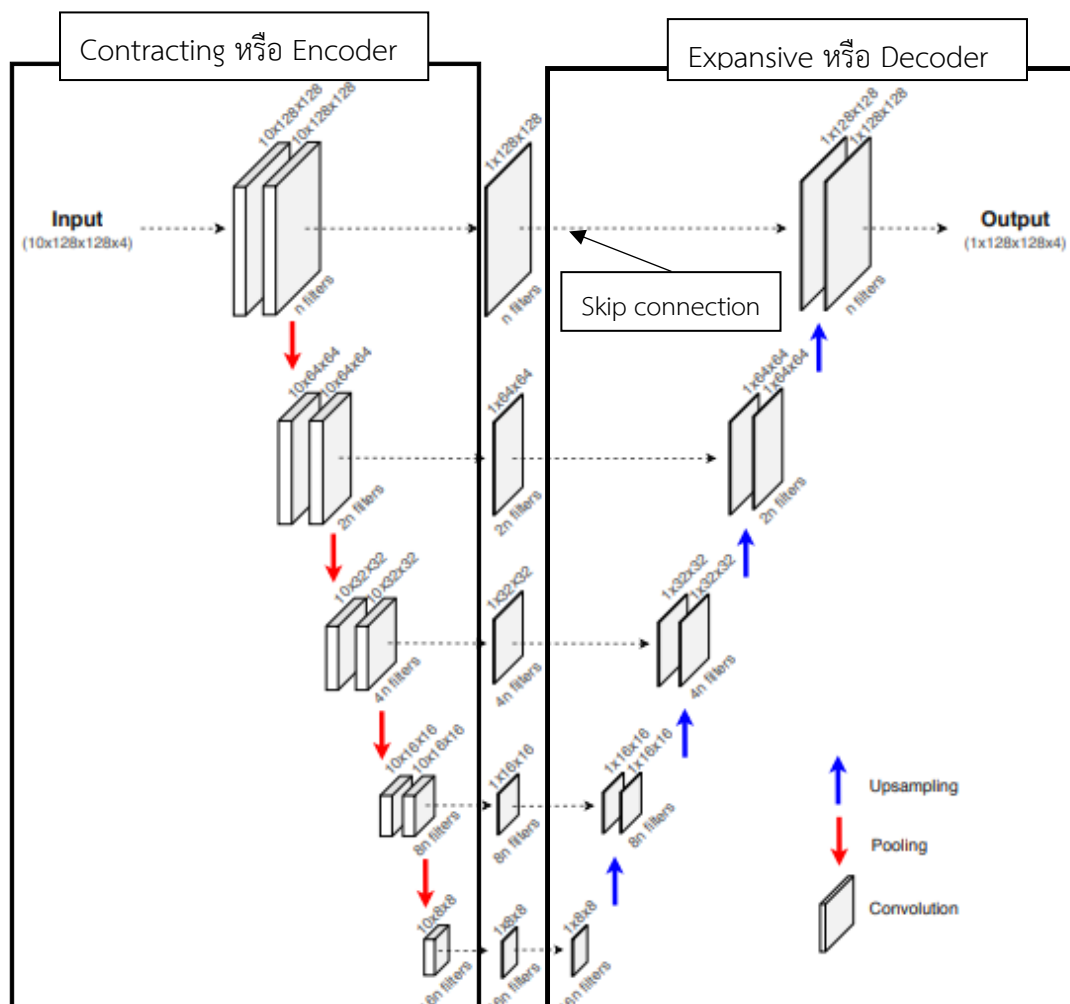
2. ชั้นกระตุ้น (Activation layer) เป็นชั้นที่เปรียบเสมือนตัวแปลงค่า (transform) โดยก่อนที่ input จะถูกส่งไปยัง โหนด node อื่น ๆ จะต้องถูก transform ให้เหมาะสมก่อน หน้าที่นั้นเป็น ประตุทางผ่านสำหรับชั้นก่อนหน้าและขัดถัดไป โดยฟังก์ชันกระตุ้นนั้นมีหลายรูปแบบอยู่ที่ใช้ กำหนด เช่น ฟังก์ชัน Rectified Linear Unit (ReLU) หรือ ฟังก์ชัน Soft max
3. ชั้นพูลลิ่ง (Pooling or Sub-sampling layer) เป็นชั้นที่มีวัตถุประสงค์คือลดขนาดของ feature จากการสุ่มตัวอย่าง (sampling) เพื่อลดจำนวนพารามิเตอร์ (parameter) และการคำนวณใน โครงข่ายลง โดยการ pooling นั้นยกตัวอย่างเช่น max pooling คือ การสุ่มตัวอย่างจากค่าที่ มากสุดในขอบเขตของ filter หรือ average pooling คือการสุ่มค่าตัวอย่างแบบเฉลี่ยในขอบเขต ของ filter
4. ชั้นเชื่อมโยงสมบูรณ์ (Fully connected layer) เป็นชั้นดำเนินการขั้นสุดท้ายของโครงข่าย มี การเชื่อมต่อกับ Feature กับ output แบบเต็มระบบกับ ฟังก์ชันจาก activation layer
5. ชั้นถอดรหัส (Decoder หรือ Up-sampling layer) เป็นชั้นที่ใช้การถอดรหัส Deconvolution เพื่อขยายข้อมูลผลลัพธ์จากการดำเนินการ convolution จากตอนแรกให้มีขนาดของภาพเท่ากับ ภาพที่นำเข้ามาจากขั้นตอนแรก
6. ชั้นปรับปรุงโครงสร้าง (Batch normalization layer) เป็นชั้นที่มีการดำเนินการทางคณิตศาสตร์ เพื่อเร่งความเร็วในการฝึกสอนของ neural network ชั้นนี้ช่วยให้โมเดลลูเข้าสู่จุดที่ทำงานได้ เหมาะสมเร็วขึ้น
7. ชั้นรวบรวม (Merging layer) เป็นชั้นที่ผสมผสานผลลัพธ์ของสองชั้นหรือมากกว่าด้วยกัน โดยรับ ข้อมูลมาเป็นแบบภาพชนะ (tensor) ที่มีรูปร่างทั้งหมดเหมือนกัน แล้วส่งผลลัพธ์เป็น tensor ตัว เดียว [4]



รูปที่ 7 การทำงานของ Convolution Neural Network (ที่มา: medium.com/CNN โดย: Natthawat)

2.4 U-Net Model

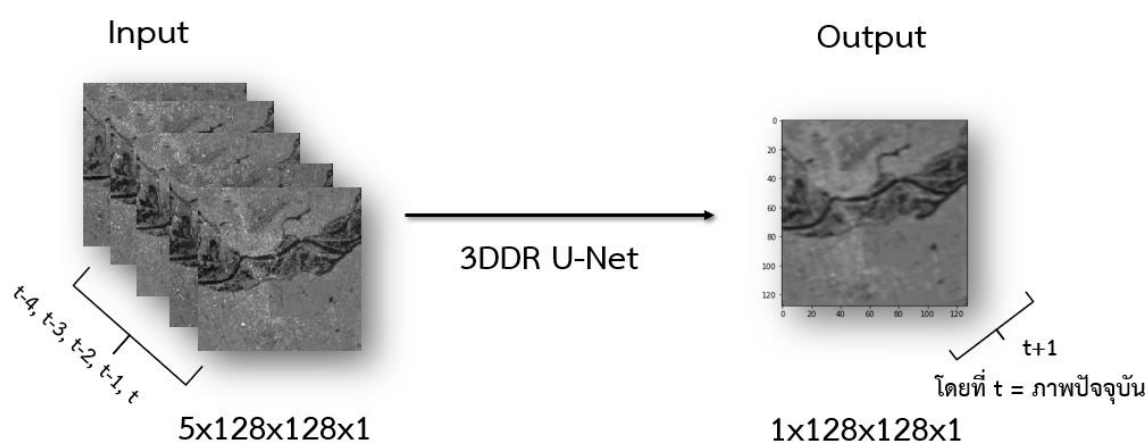
โมเดล U-net นั้นมีพื้นฐานมาจากสถาปัตยกรรม Convolutional Neural Networks (CNN) ซึ่งถูกปรับปรุงและพัฒนาให้เหมาะสมกับงานการแบ่งส่วนภาพ (Segmentation) โมเดล U-net สามารถทำงานได้ดีแม้จะมีจำนวนข้อมูลภาพที่น้อยลง และยังคงให้ผลลัพธ์ที่แม่นยำสำหรับงานด้าน Segmentation สถาปัตยกรรม U-net จะประกอบด้วยส่วนหดตัว (Contracting หรือ Encoder) และส่วนขยาย (Expansive หรือ Decoder) ประกอบกันเป็นรูปตัว U โดยส่วน Contracting ประกอบด้วยการประยุกต์ใช้ชั้น Convolution ขนาด 3x3 สองแผ่นแต่ละแผ่นตามด้วยฟังก์ชันกระตุ้น Rectified Linear Unit (ReLU) จากนั้นจะเริ่มหดขนาด Size รูปโดยการ Max pooling พร้อมกับ Stride และเพิ่มจำนวนของ Feature channels เป็น 2 เท่า ส่วน Expansive ของ U-Net มีโครงสร้างคล้ายกับส่วน Contracting แต่เปลี่ยนการหดขนาดเป็นการขยายขนาดแทนเรียกว่า Up sampling และสถาปัตยกรรม U-net นั้นจะมีขั้นตอนการ Skip connection ในตรงกลางโมเดลเพื่อช่วยในการรักษาข้อมูลและรายละเอียดของภาพที่ทำนายดังรูปที่ 8 [5]



รูปที่ 8 สถาปัตยกรรม 3DDR U-Net [5]

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

3DDR U-Net นั้นพัฒนามาจาก U-Net แบบธรรมดาโดยจะมีความแตกต่างกันที่ขนาด Size ของ Input คือเป็นแบบเมทริกซ์แบบ 4 Dimension $L \times H \times W \times V$ (รูป 3 มิติ) โดยที่ L นั้นหมายถึงขนาดของ Time lag, H และ W หมายถึงขนาด size ของรูป และ V คือจำนวน Channel โดยตัวโมเดลจะทำการแกลน Feature พร้อม ๆ กันทั้งก่อน 3 มิติ รวมถึงทำการหดขนาด (Down sampling หรือ Pooling) และขยายขนาด (Up sampling) คล้าย ๆ กับ U-Net ธรรมดาแบบ 2 มิติ แต่ตัว 3D Dimension reducer จะความพิเศษโดยที่จะทำการลดขนาด Output จาก 3 มิติ (จากโครงงาน $5 \times 128 \times 128 \times 1$) เป็นเพียงแค่ Output 2 มิติ ($1 \times 128 \times 128 \times 1$) ตรงขั้นตอนของการขยายตัว และ ตรงขั้นตอนของ Skip connection ซึ่งการทำงานลักษณะนี้จะสามารถทำให้โมเดลสามารถสร้างรูปที่ทำนายโดยเป็น Size แตกต่างกับ Input ได้ ตัวอย่างดังรูปที่ 9 [5]



รูปที่ 9 ตัวอย่างรูป Input Lag = 5 และ output ของ 3DDR U-Net model

2.5 เครื่องมือในการเปรียบเทียบรูปน้ำท่วมที่ทำนายได้และรูปจริง

Mean square error (MSE)

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

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 เป็นเครื่องมือที่ใช้ในการวัดประสิทธิภาพของการปรับปรุงคุณภาพของรูปภาพ โดยหลักการคิดจะมาจากการนำพิกเซลรูปภาพที่ทำนายได้ไปเทียบกับค่าจริงจาก MSE และนำไปใส่ค่าลอการิทึมโดยค่า PSNR ที่ได้ควรจะอยู่ในช่วง 1-100 decibel (dB) ยิ่ง PSNR มีค่ามากจะบ่งบอกว่าภาพที่ทำนายได้นั้นมีความรบกวนเทียบกับภาพต้นฉบับนั้นน้อย [6]

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

โดยที่

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

MSE คือค่า Mean Square error ดังสมการที่ (1)

Structural similarity index measure (SSIM)

SSIM เป็นดัชนีวัดความคล้ายคลึงของภาพต้นฉบับและภาพที่ทำนายได้ โดยดัชนีจะวัดเทคนิคทางด้านภาพ 3 เรื่อง คือ แสง (Luminance), ความสว่าง(Contrast) และโครงสร้างของภาพ (Structure) โดยที่ค่าที่คำนวณได้จะมีค่าระหว่าง 0-1 ซึ่งเข้าใกล้ 1 หมายความว่าภาพต้นฉบับนั้นมีใกล้เคียงกับภาพที่ทำนายได้ [6]

$$SSIM(A, B) = \frac{(2\mu_A\mu_B + C_1)(2\sigma_{AB} + C_2)}{(\mu_A^2 + \mu_B^2 + C_1)(\sigma_A^2 + \sigma_B^2 + C_2)} \dots (3)$$

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

C_1 คือ $(0.01L)^2$ โดยที่ L นั้นหมายถึงค่า MAX_I ดังสมการที่ (2)

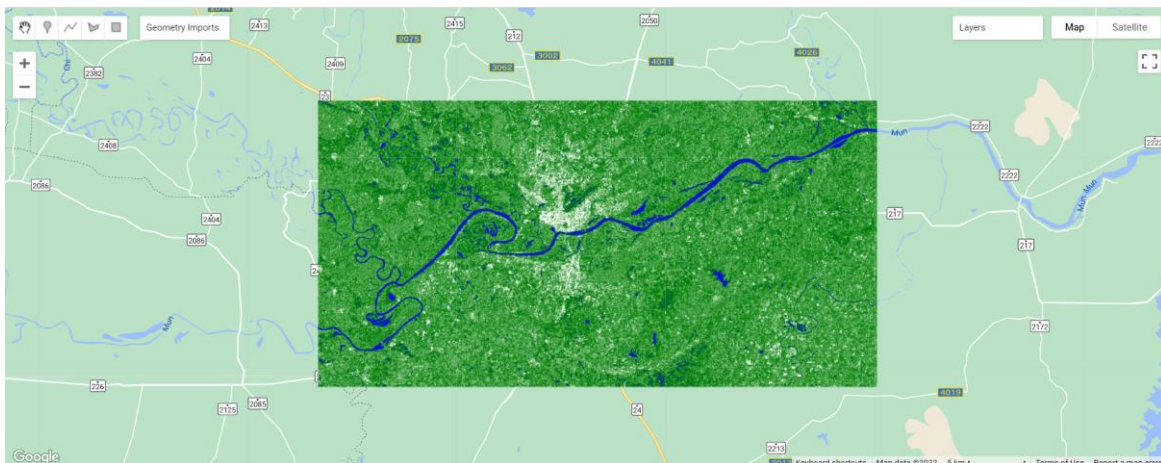
C_2 คือ $(0.03L)^2$ โดยที่ L นั้นหมายถึงค่า MAX_I ดังสมการที่ (2)

3. ผลลัพธ์ของโครงการและการอภิปรายผล

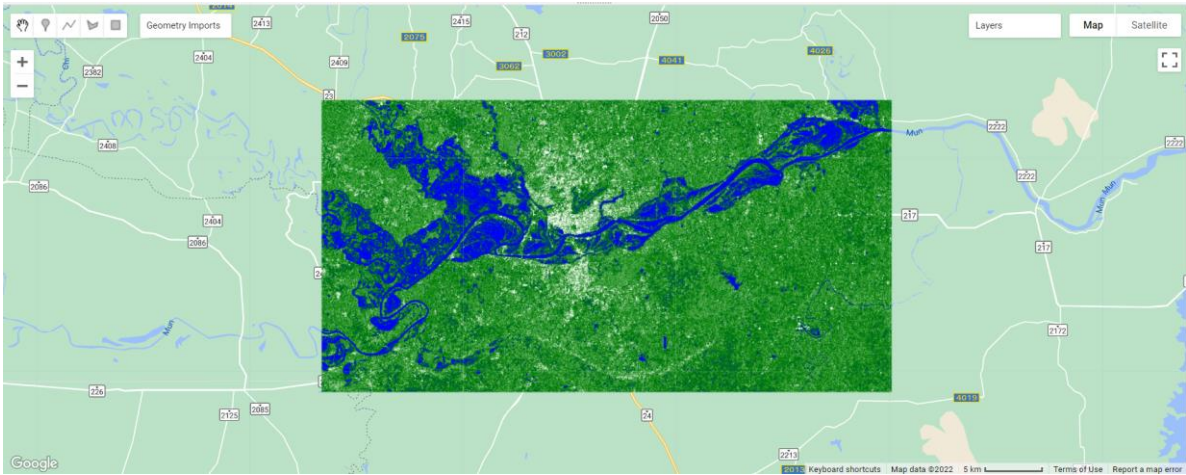
จากการเก็บข้อมูลภาพถ่ายดาวเทียมจากตัวโปรแกรม Google earth engine และ Google Colab โดยเลือกใช้ข้อมูลดาวเทียม Sentinel-1 จาก Copernicus ทำให้สามารถได้ข้อมูล (Data) ภาพของน้ำท่วม ในบริเวณพื้นที่และช่วงระยะเวลาที่ต้องการได้ โดยได้นำข้อมูลที่เป็นรูปภาพถ่ายดาวเทียมเหล่านั้นมาสอน (Train) ให้กับตัวโมเดล 3D Dimension Reducer U-Net (3DDR U-Net) โดยตัวข้อมูลจะแบ่งเป็นพื้นที่ราบลุ่มที่มีน้ำอยู่แล้วกับพื้นที่ราบลุ่มธรรมดา ผลลัพธ์ที่ได้จากการสอนโมเดลจะนำมาเทียบกับภาพจริงโดยใช้ตัวเปรียบเทียบเป็น MSE, PSNR และ SSIM

3.1 ผลลัพธ์ที่ได้จากการ Visualization บน Google earth engine

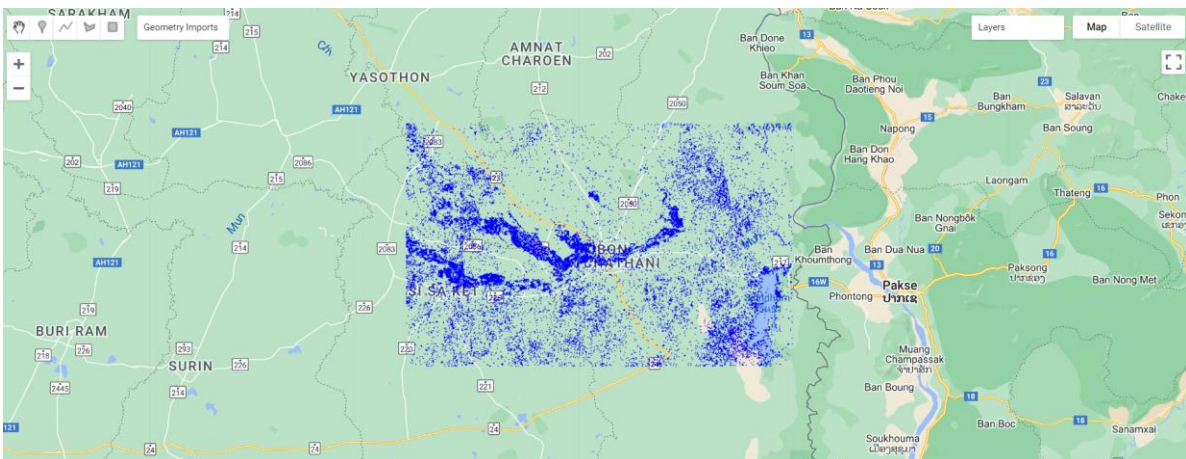
จากการเขียนโปรแกรมมิ่งโดยภาษาจาวา (JavaScript) บนโปรแกรม Google earth engine นั้นโดยเลือกข้อมูลจากดาวเทียมเป็น Copernicus/S1_GRD และเลือก Polarization เป็น VV จากนั้นได้สร้างตัวแปรเพื่อเลือกวันที่โดยแบ่งเป็นสองตัวแปรคือ before (ก่อนเกิดน้ำท่วม) และ after (หลังเกิดน้ำท่วม) จากนั้นเลือกพื้นที่โดยใช้คำสั่ง geometry เพื่อเลือกจังหวัดอุบลราชธานีที่ลองจิจูด (longitude) ที่ 105 องศาและละติจูด (latitude) 13.5 องศา จากนั้นใช้คำสั่ง add layer เพื่อให้แสดงผลเป็น layer จากข้อมูลดาวเทียม โดยได้ใช้คำสั่ง palette เพื่อให้มองเห็นรูปได้ง่ายขึ้น พื้นที่น้ำจะแสดงเป็นสีน้ำเงิน ส่วนพื้นที่ดินจะแสดงเป็นสีเขียวดังรูปที่ 10 และ 11 [7]



รูปที่ 10 วันที่ 5 พฤษภาคม 2022 ก่อนเกิดน้ำท่วม จังหวัดอุบลราชธานี (Ubon Before flood)



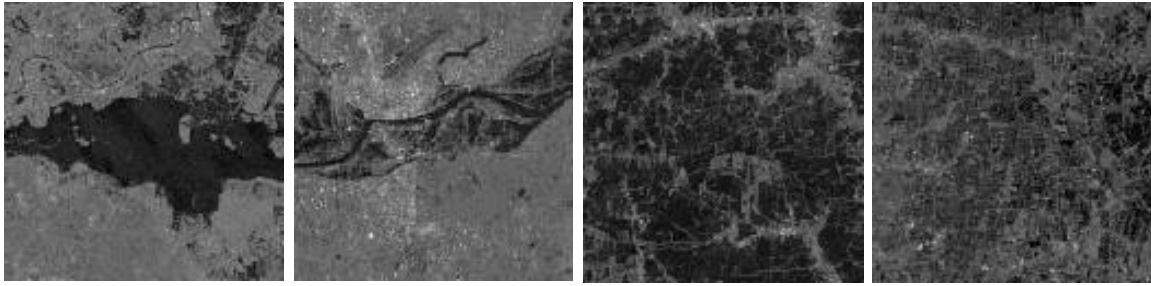
รูปที่ 11 วันที่ 20 ตุลาคม 2022 หลังเกิดน้ำท่วม จังหวัดอุบลราชธานี (Ubon during Flood)
จากการนำใช้ฟังก์ชันลบ (subtract) ระหว่างภาพที่ได้จากก่อนน้ำท่วม (Ubon Before flood) และภาพที่ได้หลังน้ำท่วม (Ubon during Flood) พร้อมกับการใช้คำสั่งโฟกัส (focal_median) เป็นจุดเล็ก ๆ โดยที่ตั้งค่าความละเอียดอยู่ที่ 100 เมตร (Smooth radius) โดยจะได้ผลลัพธ์ดังรูปที่ 12



รูปที่ 12 พื้นที่ที่มีโอกาสที่จะเกิดน้ำท่วม จังหวัดอุบลราชธานี (flood area)

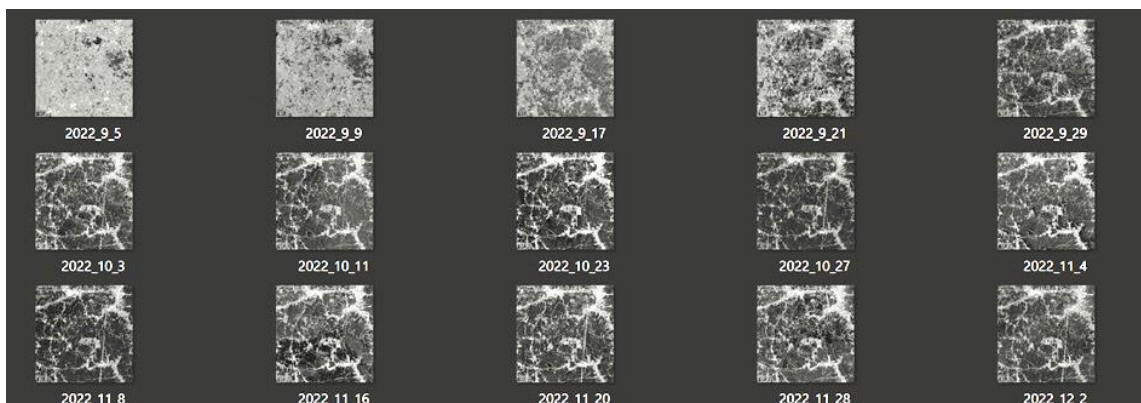
3.2 ผลลัพธ์ที่ได้จากการ Collect data จาก Google Colab

จากการเขียนโปรแกรมมิ่งโดยใช้ภาษา Python บนโปรแกรม Google Colab โดยเริ่มจากการใช้คำสั่ง ImageCollection เพื่อรับข้อมูลจาก Copernicus/S1_GRD โดยเลือก Polarization เป็น VV เลือก geometry เป็นจุดลองติจูด (Longitudes) และ ละติจูด (Latitude) ที่ต้องการ พร้อมกับใช้คำสั่ง Export Image to drive เพื่อนำรูปที่ได้ไปยัง Google drive โดยเลือกโฟลเดอร์ที่ต้องการและค่าความละเอียดต่อ Unit pixel (Scale) เป็น 100 หลังจากนั้นใช้คำสั่ง For loop เพื่อให้ดาวนโหลดข้อมูลจนครบเป็นเวลาตามที่ต้องการ โดยไฟล์ที่ได้นั้นจะเป็นไฟล์ tiff และระยะห่างวันของภาพที่ถ่ายนั้นขึ้นอยู่กับการโคจรของดาวเทียม ซึ่งจะประมาณได้ว่า 2-12 วัน รูปตัวอย่างดังรูปที่ 13

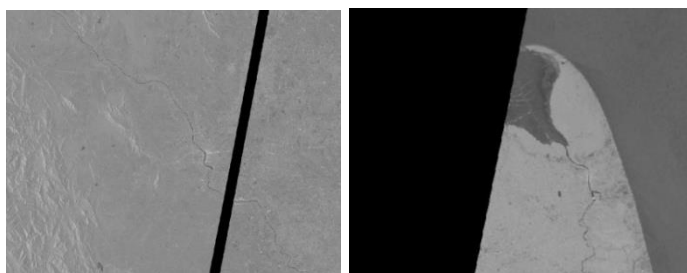


รูปที่ 13 ตัวอย่างภาพขนาด 128x128 จังหวัด นครสวรรค์, อุบลราชธานี, พระนครศรีอยุธยา, นครปฐม ตามลำดับ

จากรูปที่ 13 เมื่อตรวจสอบวันที่และพิกัดตามแผนที่ประเทศไทยจะได้ว่าเป็นวันที่ 17 กันยายน ปี 2022 พื้นที่ของจังหวัดนครสวรรค์คือบึงบอระเพ็ด อำเภอเมืองนครสวรรค์, วันที่ 29 ตุลาคม ปี 2022 พื้นที่จังหวัดอุบลราชธานีคือแม่น้ำมูล อำเภอเมืองอุบลราชธานี, วันที่ 29 กันยายน 2022 พื้นที่จังหวัดพระนครศรีอยุธยา อำเภอผักไห่ และวันที่ 3 ตุลาคม ปี 2022 พื้นที่จังหวัดนครปฐม อำเภอกำแพงแสน โดยทั้ง 4 พื้นที่จะมีระยะเวลาการท่วมเฉลี่ยรวมกันอยู่ที่ 12 ภาพหรือประมาณ 20-62 วัน



รูปที่ 14 ภาพน้ำท่วมเดือนกันยายน 2022 อำเภอผักไห่ จังหวัดพระนครศรีอยุธยา



รูปที่ 15 รูปภาพที่ใช้งานไม่ได้ของภาคเหนือ จังหวัดสุโขทัย และภาคใต้ จังหวัดนครศรีธรรมราช

การเก็บภาพดาวเทียมจะขึ้นอยู่กับวงโคจรของดาวเทียม ณ ขณะนั้นจะสามารถเก็บภาพได้ครอบคลุมทั้ง พิกัดหรือไม่ จากการตรวจสอบพบว่าที่ภาคเหนือและภาคใต้ของประเทศไทยตัวดาวเทียมนั้นถ่ายรูปไม่ ครอบคลุมทั้งพื้นที่เป็นจำนวนหลายรูป ดังรูปที่ 15 และสามารถสรุปจำนวนรูปที่เก็บได้ตามตารางที่ 2

ตารางที่ 2 จำนวนรูปภาพที่เก็บได้จากดาวเทียม Copernicus แบ่งตามภูมิภาค

ภูมิภาค รูปภาพ	ภาคเหนือ*	ภาค ตะวันออกเฉียงเหนือ	ภาคกลาง			ภาคใต้*
		อุบลราชธานี	พระนครศรีอยุธยา	นครสวรรค์	นครปฐม	นครศรีธรรมราช
รูปดาวเทียมที่เก็บได้ ปี 2015 ถึง 2022	126	558	499	483	497	262
รูปภาพที่ใช้งานได้	56	558	499	481	492	82
จำนวนรูปที่มี เหตุการณ์น้ำท่วม	4	40	104	81	56	8
จำนวนครั้งที่ท่วม (2015-2022)	-	3	7	5	4	-
จำนวนภาพที่ใช้เทรน โมเดล	-	456	373	481	369	-

หมายเหตุ * ภาคเหนือและภาคใต้ผู้จัดทำเก็บภาพดาวเทียมเพียงแค่ปี 2018 ถึงปี 2022

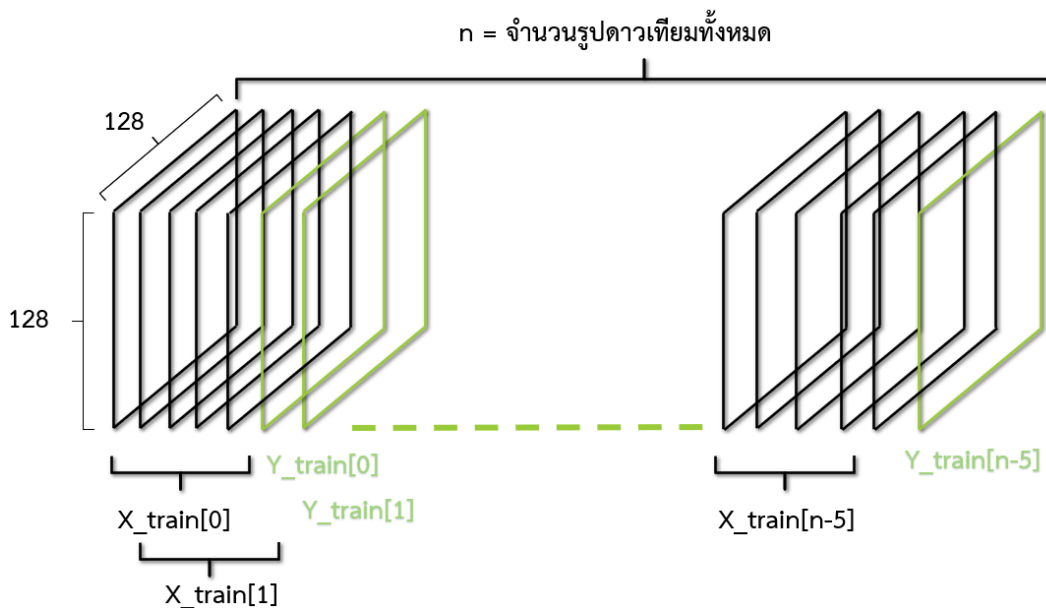
จากตารางที่ 2 จะเห็นว่าภาคใต้และภาคเหนือมีดาวเทียม Copernicus มีจำนวนภาพถ่ายที่ใช้งานได้มีจำนวนน้อยมากเมื่อเทียบกับภาคกลางและภาคตะวันออกเฉียงเหนือ ในขั้นตอนการเทรน Model ทางผู้จัดทำจึงเลือกเทรนเฉพาะภาพดาวเทียมจากภาคตะวันออกเฉียงเหนือและภาคกลาง และเนื่องจากทั้ง 2 ภาคนี้เป็นพื้นที่ราบลุ่มคล้ายคลึงกันผู้จัดทำจึงแบ่ง Data เป็นประเภทน้ำท่วมแบบล้นตลิ่งและน้ำท่วมขัง

แบ่งการเทรนเป็น 4 ครั้งดังนี้

1. ประเภทน้ำท่วมล้นตลิ่งที่ บึงบอระเพ็ด อำเภอเมืองนครสวรรค์ จังหวัดนครสวรรค์
2. ประเภทน้ำท่วมล้นตลิ่งที่ แม่น้ำมูล อำเภอเมืองอุบลราชธานี จังหวัดอุบลราชธานี
3. ประเภทน้ำท่วมขังที่ พื้นที่อำเภอดักไถ่ จังหวัดพระนครศรีอยุธยา
4. ประเภทน้ำท่วมขังที่ พื้นที่อำเภอกำแพงแสน จังหวัดนครปฐม

3.3 ผลลัพธ์จากการเทรน 3D Dimension Reducer U-Net model

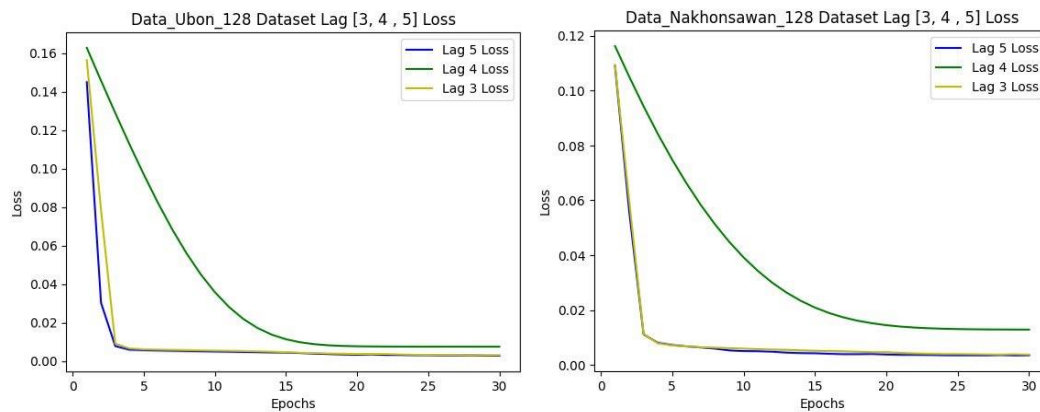
จากการเทรน Model ด้วยชุดภาพดาวเทียมโดยแบ่งเป็น 4 จังหวัดตามหัวข้อที่ 3.2 โดยจะใช้ภาพดาวเทียมทั้งหมด 5 ปีคือ ปี 2018 ถึง ปี 2022 โดยจะจัด Data เป็น X_train และ Y_train ดังรูปที่ 16 และจัด Data X_test เป็นปี 2015 ถึงปี 2017 โดยจะเลือกเฉพาะช่วงที่เป็นน้ำท่วมในแต่ละจังหวัดนั้น จากผลลัพธ์การ train โดยใช้ภาพย้อนหลังเท่ากับ 3, 4 และ 5 จะพบว่าที่ การใช้ภาพดาวเทียมย้อนหลัง 5 ภาพ (ค่า Lag ใน model เท่ากับ 5) นั้นตัว Model จะมี Loss ที่จุดอิมิตัวต่ำที่สุด ทั้ง 4 จังหวัด ดังรูปที่ 17 และ 18



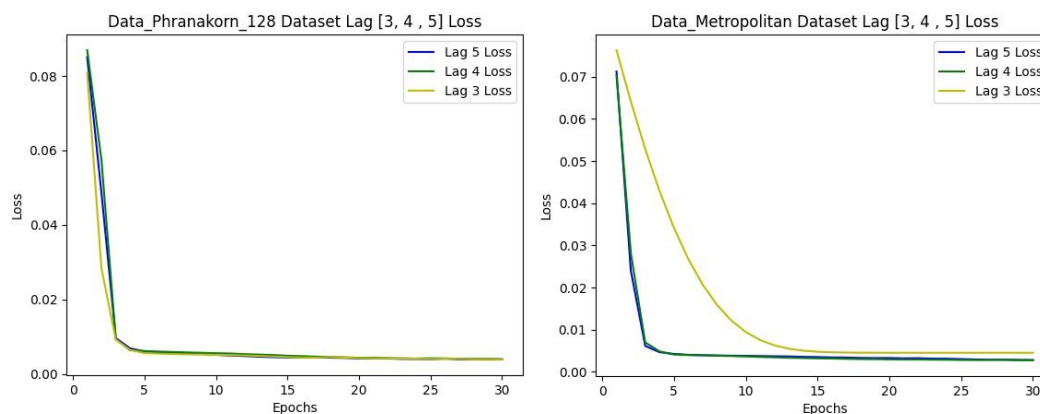
รูปที่ 16 การจัด Data เป็น X_train และ Y_train เพื่อให้ Model train

จากความต้องการทราบว่าตัว Model นั้นจะทำนายน้ำท่วมได้ภายในระยะเวลากี่วัน ผู้จัดทำจึงได้เทียบ ผลลัพธ์ของรูปที่ทำนายกับรูปจริงจากข้อมูลที่น่าไปเทรนแล้วใน Model (Seen Data) โดยเทียบว่า Input จำนวนภาพน้ำท่วมมาแล้ว 1, 2, 3, 4 และ 5 ภาพที่ภาพน้ำท่วมมาแล้วก่อนหน้านี้เท่าใดนั้นจะทำให้ Model ทำนายได้แม่นยำที่สุด พบว่าเมื่อ Input มีภาพน้ำท่วมก่อนหน้านี้ตั้งแต่ 4 ภาพจะทำให้ Model ทำนายภาพน้ำท่วมผลลัพธ์ได้ดีที่สุด โดยผลลัพธ์ของ MSE จะใกล้เคียงกับ 5 ภาพ, PSNR จะมากขึ้นอย่างมีนัยสำคัญเมื่อเทียบกับ 3 ภาพ และ SSIM มีค่าต่ำใกล้เคียงกับ 5 ภาพเช่นกัน ดังรูปที่ 20 ดังนั้นจึงบอกได้ว่าตัว Model นั้นจะสามารถทำนายน้ำท่วมก่อนหน้านี้ได้ดีที่สุดเมื่อมีน้ำท่วมมาแล้ว 4 ภาพ ดังรูปตัวอย่าง น้ำท่วมล้นลิ่งจังหวัด อุบลราชธานี รูปที่ 19 และ 21

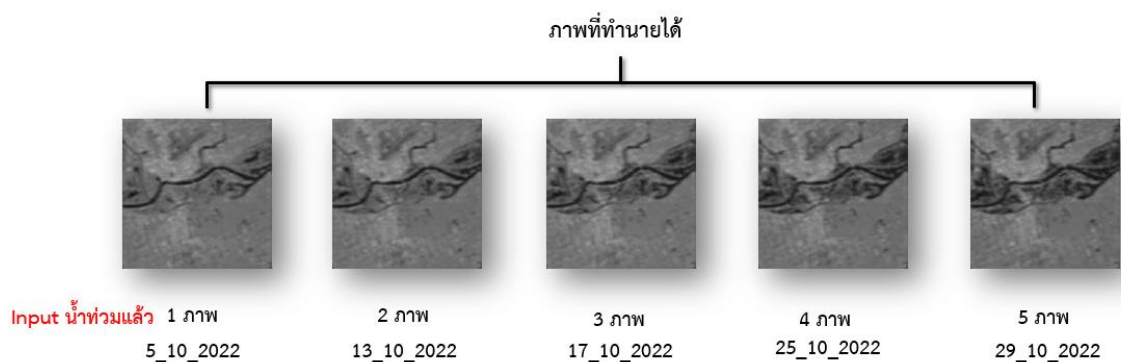
และผลลัพธ์เมื่อเทียบภาพที่ได้จากการทำนายและภาพจริง ของน้ำท่วมทั้ง 2 ประเภทคือแบบน้ำท่วมล้นตลิ่งและน้ำท่วมขังโดยใช้ X_test Data คือ Data ปี 2015 ถึง ปี 2017 ที่ Model ยังไม่เคยได้รับการ train (Unseen data) ให้ Input มีภาพน้ำท่วมมาแล้ว 4 วัน เมื่อเทียบค่า MSE, PSNR, SSIM จากการทำนายแล้วจะพบว่า น้ำท่วมแบบขังนั้นมีค่าการทำนายที่แม่นยำมากกว่า น้ำท่วมล้นตลิ่ง เฉลี่ยทั้ง 2 จังหวัด ที่น้ำล้นตลิ่งเท่ากับ 9520, 8.52 dB, 5.00e-3 ตามลำดับ และน้ำท่วมขังเท่ากับ 5104, 11.175 dB, 5.26e-3 ดังตารางที่ 3 และตัวอย่างรูปของการทำนายจาก Unseen data ของแต่ละจังหวัดดังรูปที่ 22, 23, 24 และ 25



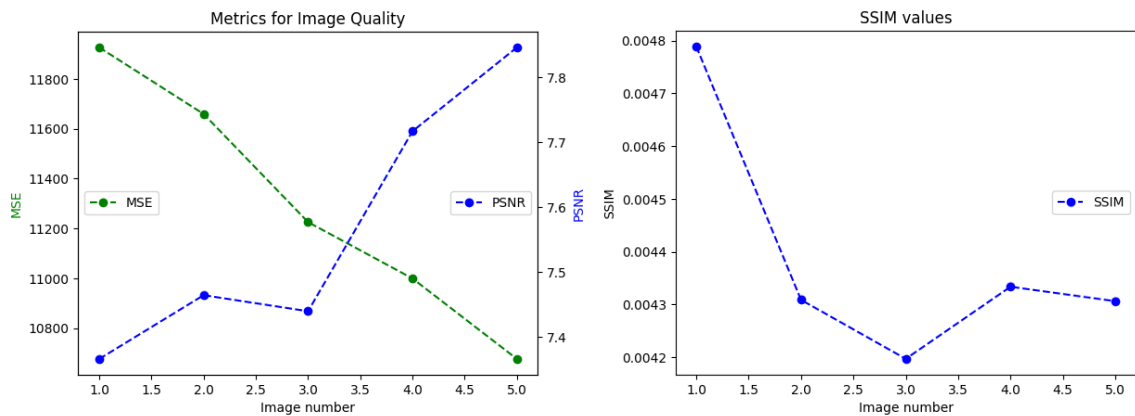
รูปที่ 17 เปรียบเทียบ Loss ของจังหวัดอุบลราชธานีและจังหวัดนครสวรรค์



รูปที่ 18 เปรียบเทียบ Loss ของจังหวัดพระนครศรีอยุธยาและจังหวัดนครปฐม

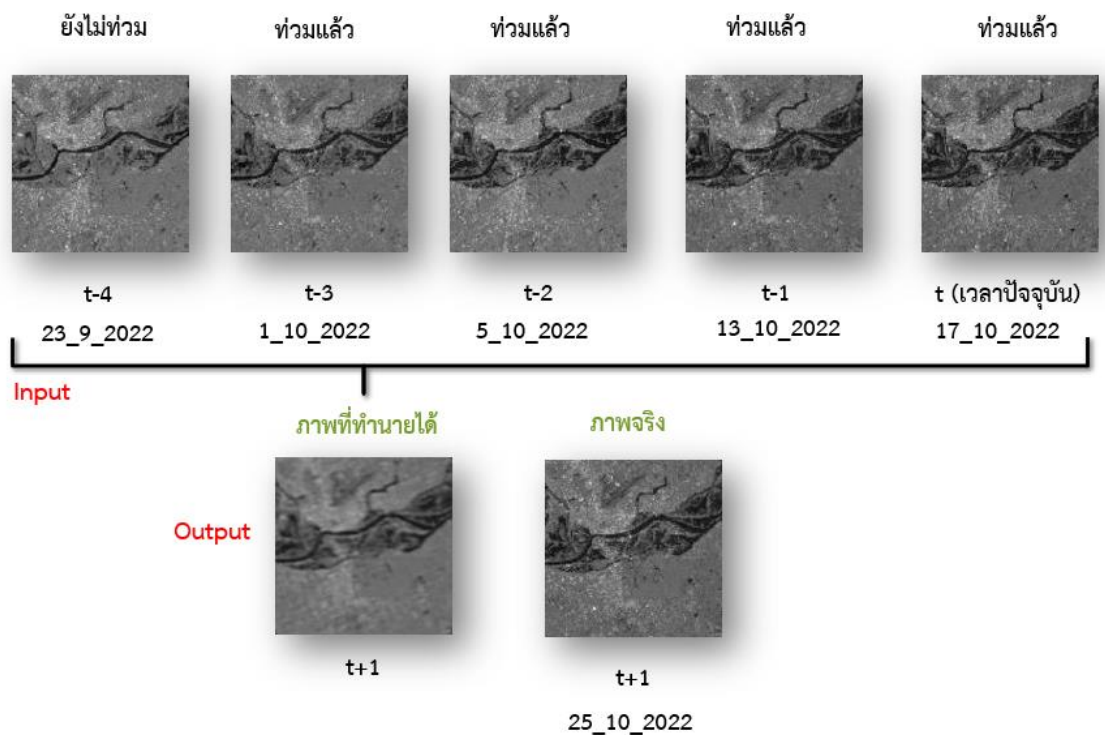


รูปที่ 19 รูปผลลัพธ์ที่ทำนายได้ตามจำนวนการใส่ภาพน้ำท่วมใน Input ที่จังหวัดอุบลราชธานี



รูปที่ 20 PSNR, MSE และ SSIM ของจังหวัดอุบล จำนวนภาพน้ำท่วมมาแล้ว 1, 2, 3, 4, 5 ภาพใน Input

จากการวิเคราะห์ผลลัพธ์ที่ได้จากตารางที่ 3 จะเห็นว่ารูปที่ทำนายจากจังหวัดพระนครศรีอยุธยา นั้นมีความใกล้เคียงกับรูปน้ำท่วมจริงมากที่สุด ซึ่งสมเหตุสมผลเนื่องจากจังหวัดพระนครศรีอยุธยาเป็นจังหวัดที่มีรูปภาพน้ำท่วมเยอะที่สุดเมื่อเทียบทั้ง 4 จังหวัด และรองลงมาจะเป็นจังหวัดนครปฐมและจังหวัดนครสวรรค์ที่มีค่าใกล้เคียงกัน และจังหวัดสุดท้ายคือจังหวัดอุบลราชธานี ซึ่งเป็นจังหวัดที่ไม่ได้มีน้ำท่วมบ่อยมากนัก แต่เมื่อท่วมแล้วจะกินระยะเวลาและรุนแรง



รูปที่ 21 ตัวอย่างจังหวัดอุบล มีภาพน้ำท่วมมาแล้ว 4 ภาพเป็นต้นไปมีความแม่นยำในการทำนายสูง

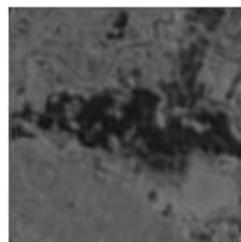
ตารางที่ 3 ผลลัพธ์จากการทำนาย Unseen data ของแต่ละจังหวัด

ผลจากการวัดค่าแต่ละจังหวัด	MSE	PSNR (dB)	SSIM
จังหวัดอุบลราชธานี	12180	7.27	4.54e-3
จังหวัดนครสวรรค์	6861	9.77	5.46e-3
จังหวัดพระนครศรีอยุธยา	4991	11.14	5.38e-3
จังหวัดนครปฐม	5217	10.95	5.14e-3

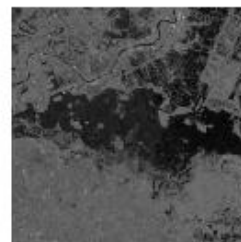
หมายเหตุ พื้นหลังสีเขียวหมายถึง น้ำท่วมแบบล้นตลิ่ง

พื้นหลังสีน้ำเงินหมายถึง น้ำท่วมแบบขัง

Prediction



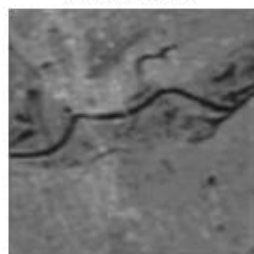
Real



2017_9_1

รูปที่ 22 รูปน้ำท่วมที่ทำนายได้จาก Unseen Data และรูปน้ำท่วมจริง จังหวัดนครสวรรค์

Prediction

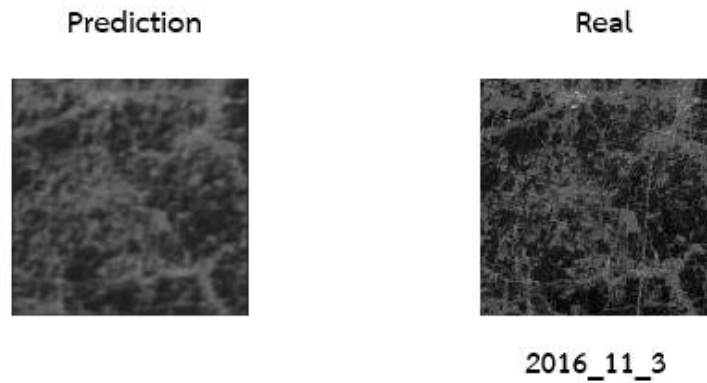


Real

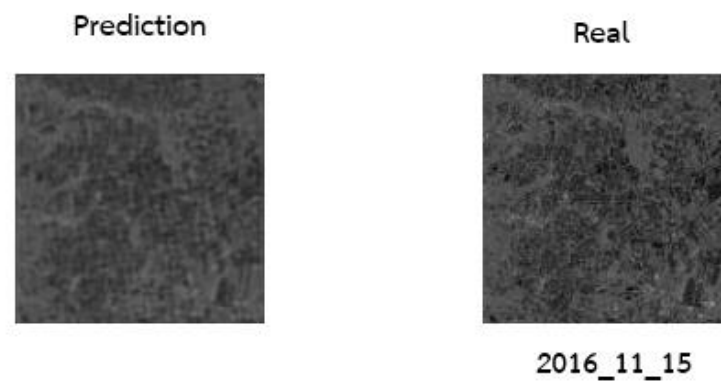


2017_8_23

รูปที่ 23 รูปน้ำท่วมที่ทำนายได้จาก Unseen Data และรูปน้ำท่วมจริง จังหวัดอุบลราชธานี



รูปที่ 24 รูปน้ำท่วมที่ทำนายได้จาก Unseen Data และรูปน้ำท่วมจริง จังหวัดพระนครศรีอยุธยา



รูปที่ 25 รูปน้ำท่วมที่ทำนายได้จาก Unseen Data และรูปน้ำท่วมจริง จังหวัดนครปฐม

4. บทสรุป

4.1 สรุปผลการดำเนินการ

โครงการนี้นำเสนอการทำนายภาพน้ำท่วมจากดาวเทียม โดยการเก็บข้อมูลภาพดาวเทียมผ่าน Google earth engine ตั้งแต่ปี 2015 ถึง 2022 และใช้ดาวเทียม Copernicus เลือกพิกัดเป็น ภูมิภาคเหนือ, ภาคตะวันออกเฉียงเหนือ, ภาคกลาง และภาคใต้ของประเทศไทย แต่ละภูมิภาคจะได้จำนวนของภาพและความละเอียดของภาพดาวเทียมแต่ละจังหวัดในช่วงน้ำท่วมแตกต่างกัน ขึ้นอยู่กับวงโคจรของดาวเทียม ซึ่งผลลัพธ์ของการเก็บภาพดาวเทียมจากโครงการจะพบว่าเมื่อพิกัดเป็นจังหวัดในภูมิภาคกลางและภาคตะวันออกเฉียงเหนือ จำนวนภาพที่ใช้งานได้จะมีจำนวนมากกว่าภูมิภาคใต้และภูมิภาคเหนือ จากนั้นเมื่อนำภาพดาวเทียมที่ได้เข้าไปฝึกสอนใน 3D Dimension Reducer U-Net โดยจะใช้ภาพดาวเทียมย้อนหลัง 5 ภาพ และแบ่งชุดภาพดาวเทียมเป็น 2 ชุดคือ ชุดภาพดาวเทียมน้ำท่วมแบบล้นตลิ่ง (River flooding) และ น้ำท่วมแบบน้ำซัง (Groundwater flooding) จะพบว่าค่า MSE, PSNR, และค่า SSIM ในการทำนายภาพน้ำท่วมแบบน้ำท่วมซังนั้นจะมีประสิทธิภาพมากกว่าการทำนายน้ำท่วมล้นตลิ่ง

4.2 ปัญหา อุปสรรค และแนวทางแก้ไข

ในช่วงขั้นตอนต้นตอนที่ทำการฝึกหัดเพื่อเขียนโปรแกรมมิงบน Google Earth Engine นั้น เนื่องจากภาษาที่ใช้ในการเขียนโปรแกรมมิงนั้นเป็นภาษา Java และจำเป็นต้องเลือกใช้ดาวเทียมและค่า Polarization ให้เหมาะสม แนวทางที่ได้แก้ไขจึงได้ปรึกษาอาจารย์ภาควิชาสำรวจเพื่อทำความเข้าใจเกี่ยวกับระบบดาวเทียมและโปรแกรม Google Earth Engine

ขั้นตอนการเก็บข้อมูลภาพถ่ายดาวเทียมจาก Google Earth Engine นั้นใช้เวลาจำนวนมากในการเก็บแต่ละภาพดาวเทียม โดยสามารถประมาณได้ว่าการเก็บข้อมูลภาพทั้งหมด 5 ปี ของแต่ละจังหวัดนั้นต้องใช้เวลาในการดำเนินการบน Google Colab เป็นระยะเวลาประมาณ 2 วันอย่างต่อเนื่องและเนื่องจากในประเทศไทยนั้นยังไม่มีมีการเก็บข้อมูลภาพถ่ายดาวเทียมแบบเรดาร์ (Radar) มากนักส่งผลให้ตัวเลือกในการเลือกดาวเทียมที่จะใช้ในการเก็บภาพนั้นมีไม่มาก

งานวิจัยใน U-Net นั้นส่วนมากจะใช้งานของการแยกรูปภาพ (Segmentation) โดยจะยังไม่มีในส่วนของการทำนาย (Prediction) มากนัก โดยในโครงงานนี้จะโมเดล (Model) ต้องรับภาพเป็นระยะเวลา (Time series) เพื่อทำนายภาพน้ำท่วมวันถัดไปซึ่งจะมีความแตกต่างจากโมเดล U-Net แบบทั่วไปที่ใช้งานเพียงแค่แบ่งแยกรูปภาพ 2D และ 3D โดยผู้จัดทำได้ใช้เวลาในการหางานวิจัย (Research) ที่เกี่ยวข้องจนพบเจองานวิจัยที่ต้องการผลลัพธ์ใกล้เคียงกัน

4.3 ข้อเสนอแนะ

เนื่องจากปัจจุบันความห่างของระยะเวลาภาพถ่ายดาวเทียมในโครงงานนี้นั้นมีค่าเป็นช่วง (4-16 วัน) การดำเนินงานในอนาคตของการทำนายชุดภาพถ่ายดาวเทียมน้ำท่วมจะมีประสิทธิภาพมากขึ้นเมื่อจำนวนภาพของดาวเทียมนั้นมีระยะห่างของระยะเวลาน้อยลงเป็นหลักชั่วโมงหรือหลักรายวัน ซึ่งในปัจจุบันนี้จำเป็นต้องใช้ค่าใช้จ่ายจำนวนมากในการเก็บภาพถ่ายดาวเทียมที่มีความละเอียดขนาดชั่วโมงและการพัฒนาชุดตัว U-Net model ให้มีประสิทธิภาพเหมาะสมกับตัวข้อมูลมากขึ้น เป็นการเพิ่มความแม่นยำของการทำนาย

5. กิตติกรรมประกาศ

โครงงานฉบับนี้สามารถสำเร็จอย่างลุล่วงได้สมบูรณ์ ด้วยความกรุณาช่วยเหลือเป็นหลักของ ผศ.ดร. วิทยากร อัครวิเศษ ที่สละเวลาอันมีค่าแก่ผู้จัดทำ เพื่อแนะนำให้คำปรึกษาอันเป็นประโยชน์ต่อโครงงานทั้งช่วยเหลือแก้ไขปัญหาที่เกิดขึ้นตลอดการดำเนินงาน ด้วยความเอาใจใส่อย่างยิ่ง

และขอขอบคุณ รศ.ดร.ไพศาล สันติธรรมนนท์ และ นายเทพชัย ศรีน้อย ที่สละเวลาให้คำปรึกษาเกี่ยวกับการเก็บข้อมูลภาพถ่ายดาวเทียมผ่าน Google earth engine และความรู้เกี่ยวกับดาวเทียม Copernicus และขอบคุณ อ.ดร. นฤมล ประทานวนิช ที่ให้คำปรึกษาเกี่ยวกับการทำ Machine learning

สุดท้ายนี้ขอขอบคุณครอบครัว และเพื่อน ๆ ของผู้จัดทำที่ให้ความช่วยเหลือและเป็นกำลังใจในการจัดทำโครงงานฉบับนี้เสมอมา

6. เอกสารอ้างอิง

- [1] กรมการส่งเสริมการปกครองท้องถิ่น, มาตรฐานการป้องกันและบรรเทาสาธารณภัยที่เกิดจากอุทกภัยและโคลนถล่ม, กระทรวงมหาดไทย, หน้า 1-2.
- [2] รองศาสตราจารย์ ดร.ไพศาล สันติธรรมนนท์, สไลด์การเรียนการสอน, “การสำรวจระยะไกลแบบแอคทีฟและเข้าถึงได้แบบเปิด”, คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย.
- [3] Int. J. Disaster Risk Sci, "Optimization of Threshold Ranges for Rapid Flood Inundation Mapping by Evaluating Backscatter Profiles of High Incidence Angle SAR Images", vol. 3, no. 2, Page 120- 122, 2012.
- [4] รัฐพงษ์ นันทาแพร่, “การเรียนรู้เชิงลึกโดยโครงข่ายประสาทเทียมคอนโวลูชันสำหรับกำหนดขอบเขตอัตโนมัติบนอวัยวะที่มีความเสี่ยงของมะเร็งศีรษะและลำคอ, วิทยานิพนธ์ปริญญามหาบัณฑิต”, มหาวิทยาลัยนเรศวร, 2563.
- [5] J. G. Fernandez, I. Alaoui and S. Mehrkanoon, "Deep coastal sea elements forecasting using U-Net based models", Page 3-4, November 2021
- [6] อัฐพร กิ่งบุ และ ปฎิมากร จริยฐิติพงศ์, "การปรับปรุงคุณภาพของภาพจากการขยายด้วยเทคนิค IBE-T", วารสารวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยอุบลราชธานี, 2563
- [7] Simon Gascoin, “Mapping flooded areas using Sentinel-1 in Google Earth Engine”, [Online] Available: <https://labo.obs-mip.fr/multitemp/mapping-flooded-areas-using-sentinel-1-in-google-earth-engine/>. [Accessed : Nov. 25, 2022].

7. ภาคผนวก

7.1 ภาคผนวก ก. Code ภาษา JavaScript ที่สร้าง Visualization บน Google earth engine สามารถศึกษาเพิ่มเติมได้ที่ <https://github.com/jackitchai/Flood-disaster.git> ที่ GGE Visualization

```
Imports (1 entry)
var geometry: Polygon, 4 vertices

1 var imgVV = ee.ImageCollection('COPERNICUS/S1_GRD')
2   .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VV'))
3   .select('VV')
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);
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);
12
13 //เลือกว่าวันที่และmosaic รูปภาพพร้อมกันด้วยภาพเพื่อให้อยู่ในกรอบของ geometry
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
20 Map.addLayer(diff_thresholded.updateMask(diff_thresholded),{min:0,max:1,palette:"blue"},"flooded areas - blue",1);
21 var col = ee.ImageCollection('COPERNICUS/S1_GRD')
22   .filterBounds(ee.Geometry.Point(105, 13.5))
23   .filterDate('2020-01-01', '2020-01-31');
24 print('Number of images', col.size());
25 // Get the number of images in the collection.
26
27
```

7.2 ภาคผนวก ข. Code ที่ Collect data จาก Google earth engine บน Google Colab สามารถศึกษาเพิ่มเติมได้ที่ <https://github.com/jackitchai/Flood-disaster.git> ที่ GGE Collect Data

```
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)

```

7.3 ภาคผนวก ค. Code ที่ใช้ในการเทรนและทำนายจากโมเดล 3DDR U-Net

สามารถศึกษาเพิ่มเติมได้ที่ <https://github.com/jackitchai/Flood-disaster.git> ที่ 3DDR U-Net

```

import tensorflow as tf
import os
import numpy as np
from tqdm import tqdm
from skimage.io import imread, imshow
from natsort import natsorted
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))
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
import matplotlib.pyplot as plt
X_train = X_train[:len(train_ids)-LAGS]
#check shape
print(X_train.shape)
print(Y_train.shape)
#check image
imshow(X_train[350][3]) #450
plt.show()
imshow(Y_train[200][0])
plt.show()

import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Add, Dropout, concatenate,
BatchNormalization, Activation
from tensorflow.keras.layers import Conv3D, MaxPool3D, UpSampling3D
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)

#--- 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)

#--- 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)

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)
model.save(filepath)
TEST_PATH = 'X_test/test_Phranakorn/Flood_1/x_test/'

test_ids = natsorted(os.listdir(TEST_PATH))
# ytest_ids = os.listdir('X_test/6/y_train/')
X_test = np.zeros((1,5, IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
# print(ytest_ids)
for n, id_ in tqdm(enumerate(test_ids), total=len(test_ids)):
    path = TEST_PATH + id_
    print(path)
    img = imread(path,as_gray= True)
    img = np.expand_dims(img, axis=-1)
    X_test[0][n] = img
# model.predict()
img_predict = model.predict(X_test)

import matplotlib.pyplot as plt
print(np.squeeze(img_predict).shape)
# plt.imshow(img_predict[0, :, :,0], cmap='gray')
# (img_predict > 0.5).astype(np.uint8)
predict_fig = np.squeeze(img_predict)
imshow(predict_fig)
# plt.imshow(img_predict)
print(Y_train.shape)
print(X_train.shape)

```

```

print(img_predict.shape)
from skimage.metrics import peak_signal_noise_ratio
from skimage.metrics import structural_similarity as ssim
from skimage import io
img_orig = imread("Predict/%s/%s_%s"%(province,province,date),as_gray= True)
# img_orig = imread('Predict/Ubon/Ubon_%s'%(date),as_gray= True)
# img_orig = imread('test8.jpg',as_gray= True)
img_compressed =
imread('X_test/test_Phranakorn/Flood_1/key_x_test/%s'%(date),as_gray= True)
# img_compressed =
imread('Seen_x_test/Ubon/Flood_5/key_x_test/%s'%(date),as_gray= True)
# img_compressed = imread('X_test/1/x_train/2019_9_5.jpg',as_gray=True)
print(img_orig.shape)
print(img_compressed.shape)
imshow(img_orig)
plt.show()
imshow(img_compressed)
plt.show()

psnr = peak_signal_noise_ratio(img_orig, img_compressed)
mse = np.mean((img_orig - img_compressed) ** 2)
ssim_score = ssim(img_orig, img_compressed, multichannel=True)
print(ssim_score)
print(mse)
print(psnr)

```