LANDSLIDE DETECTION

Cam Le & Lam Pham & Jasmin & Matthias (DSAI - AIT)

Vienna, October 2023

AGENDA

AGENDA

I-A. Overview

- The main scope of this research is to develop a system that can automatically detect landslide regions.
- There are 2 approaches we will focus on:
  - Detect landslides of a region using features in the form of tabular data (binary classification task).
  - Detect landslides of a region using remote sensing images taken by the satellites in the Sentinel-2 mission (segmentation + classification task).

⇒ In general, we would like to **combine** ML models from the 1st approach and DL models from the 2nd approach to form a robust landslide-detecting system. (For example, we retrieve tabular data from sensors and images from Sentinel-hub based on latitude and longitude of a suspicious region and feed them into our models to make predictions).

- Regarding the traditional machine learning approach, we visualized lots of figures about the basic information and data distribution of each feature of the provided landslide dataset. We also made comments and suggestions on the next step of developing ML landslide-detection models (on the other slide) . Recently, Matthias has already developed a baseline for the ML approach and he pointed out the importance of each feature contributing to the overall result (on the other slide).
- Because the labels for the tabular data have just available recently, so in the past 2 months, we mainly focus on detecting landslides from remote sensing images.

I-B. Remote Sensing Image

- Remote sensing images: images captured by remote sensors that represent a specific location on the Earth's surface as seen from space [1].
- A common representation of remote sensing images are using multispectral imagery (multiple bands of the electromagnetic spectrum beyond just visible spectrum) .
- Environmental monitoring applications: change detection [2], urban planning [3], [4], land use classification [5], landslide detection, etc.

=> Increase in demand for accurate and efficient methods to extract useful information from remote sensing images.

⇒> This presentation will focus on segmenting landslide regions from multispectral remote sensing images.

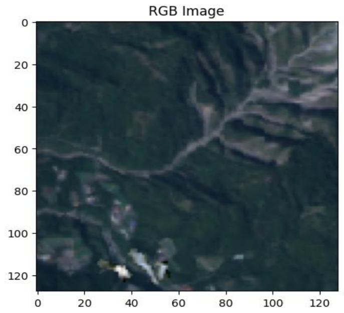


Figure: A remote sensing image with landsliding regions (rgb bands only)

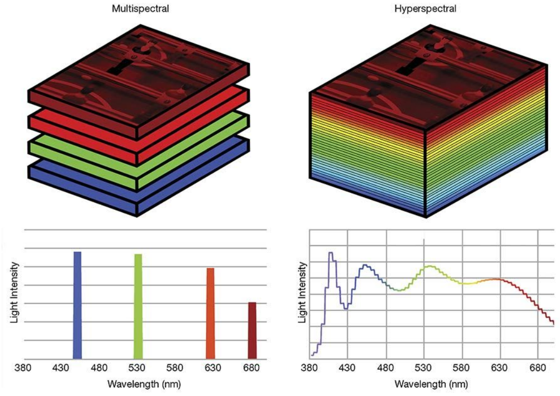


Figure: Multispectral vs Hyperspectral [6]

I-C. Data

- Use computer vision/deep learning approach applying on multi-spectral remote sensing images because of the robustness of the neural network and the availability of images from SENTINEL-2 [8] mission (which means our system can be tested easily and used in the long term).

- Dataset used for landslide segmentation: LandSlide4Sense (2022) [7] which consists of 3799, 245, and 800 images (resolution of ~10m per pixel) for the training, validation, and test set respectively.

- Each image is a composite of 14 bands including: multispectral data from Sentinel-2 (B1, ... , B12), slope data (B13) and digital elevation model (DEM) (B14) from ALOS PALSAR .

- Due to the unavailability of the server for validating and testing, we split the training set into a 80/20 train/test ratio (3040 images for training and 759 images for testing).

- Challenges:
    - Input is not RGB images as usual.
    - Class imbalance.
        - 58.72 % images of the training set have landslide.
        - The landslide regions account for only 2.31 % of the total number of pixels of the training set.
        - Maximum landslide area of an image: 47.53 %
        - Minimum landslide area of an image: 0.0061 % (1 pixel out of 128x128 pixels ~ 100 m²)

- Main evaluation metrics: F1 score (authors proposed) and Mean IOU (self evaluation).

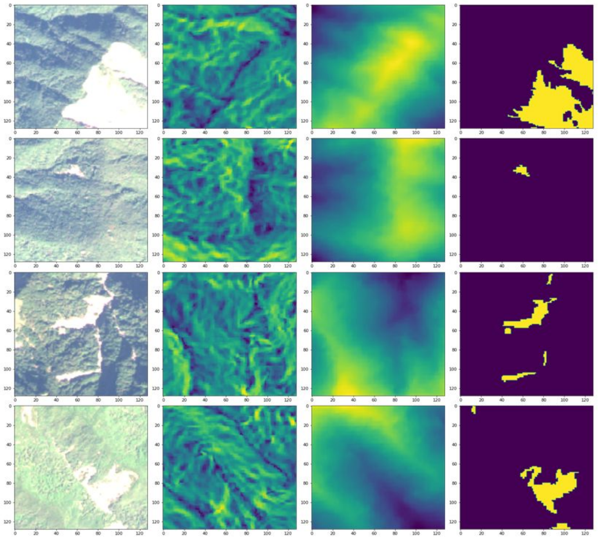
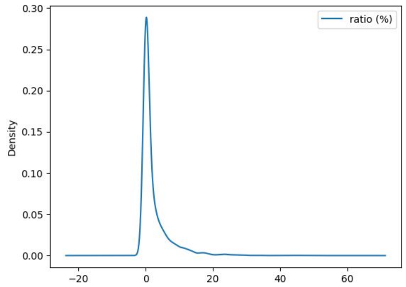


Figure: Remote sensing images and relative landslide mask [7]



I-D. Development Process

- Throughout this research: Examine and find out factors that contribute to the development of the most efficient landslide segmentation model.

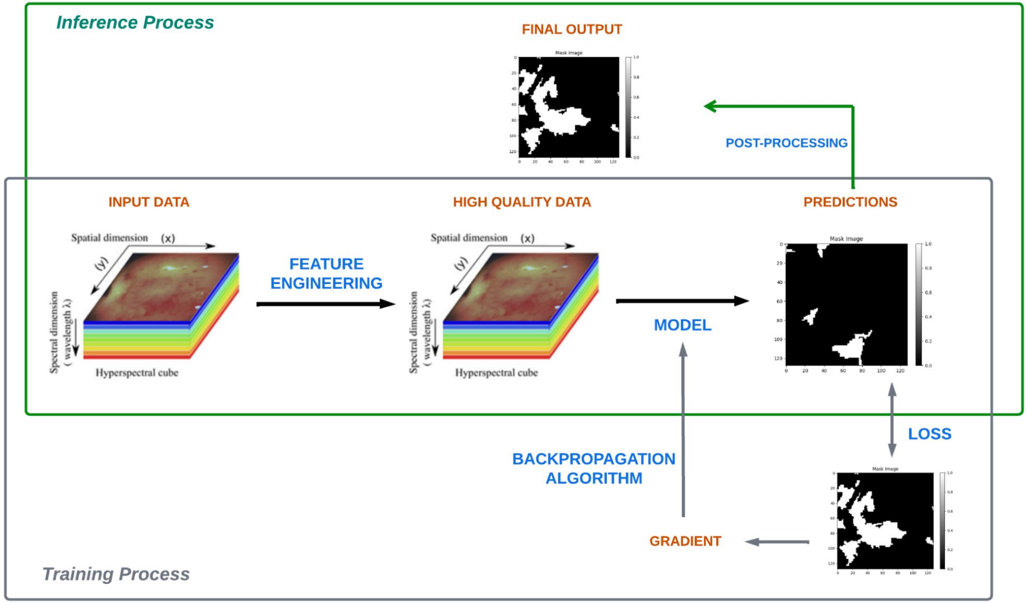


Figure: Common development process of a deep learning model

I-D. Baseline System

- Data is normalized by dividing by its mean value.
- Feature Engineering: None ⇒ input shape: (B, 128,128, 14)
- Data Augmentation: only random rotation + cutmix
- Model: traditional U-Net
- Loss: Cross Entropy (pixel-wise)
- Optimizer: Adam with a custom learning rate scheduler
- Post-processing: None

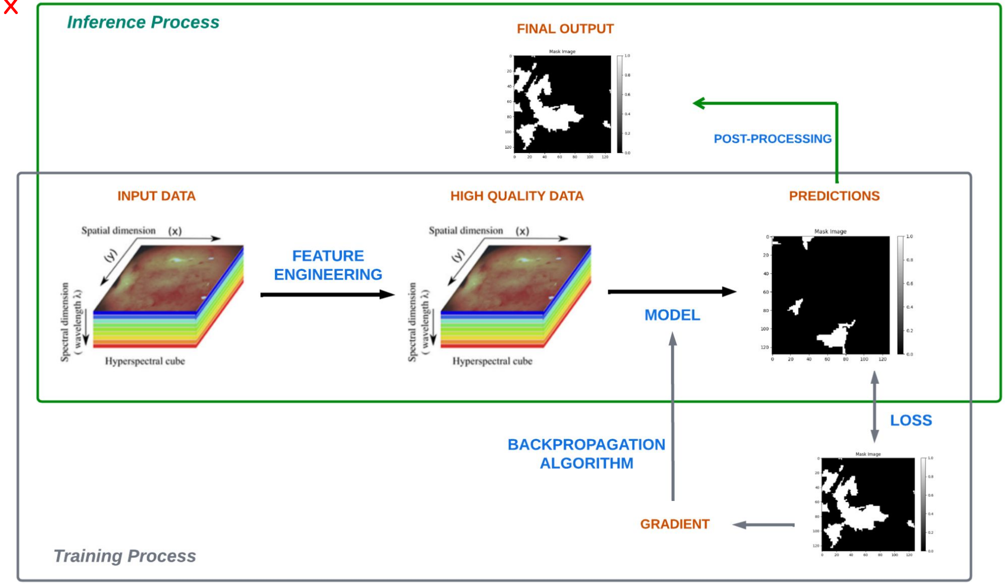


Figure: Common development process of a deep learning model

I-D. Baseline System

- Data is normalized (each band is divided by its mean value).
- Feature Engineering: None ⇒ input shape: (B, 128,128, 14)
- Data Augmentation: only random rotation + cutmix
- Model: U-Net
- Loss: Cross Entropy (pixel-wise)
- Backpropagation Algorithm: Adam with custom learning rate scheduler
- Post-processing: None

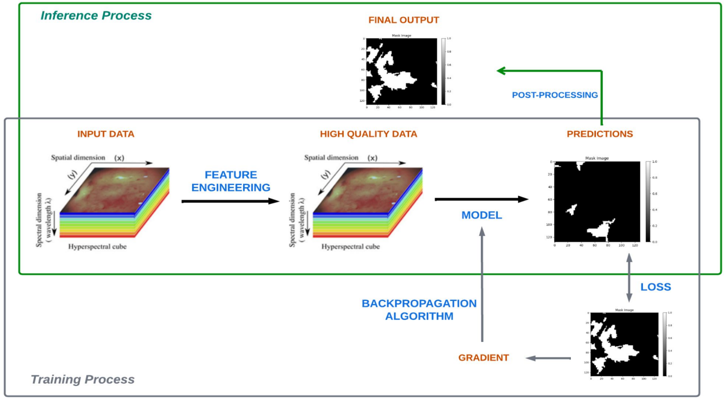


Table: Performance of baseline model

Model	Loss used	Input shape	Post process	Param	MIOU	F1	Time for 1 epoch
U-Net	CE	(B,128,128,14)	no	31.4 M	0.6001	0.6783	≈ 2m37s (gpu T4)

I-D. Baseline System

- Data is already normalized (each band is divided by its mean value).
- Feature Engineering: None ⇒ input shape: (B, 128,128, 14)
- Data Augmentation: only random rotation + cutmix
- Model: U-Net
- Loss: Cross Entropy (pixel-wise)
- Backpropagation Algorithm: Adam with custom learning rate scheduler
- Post-processing: None

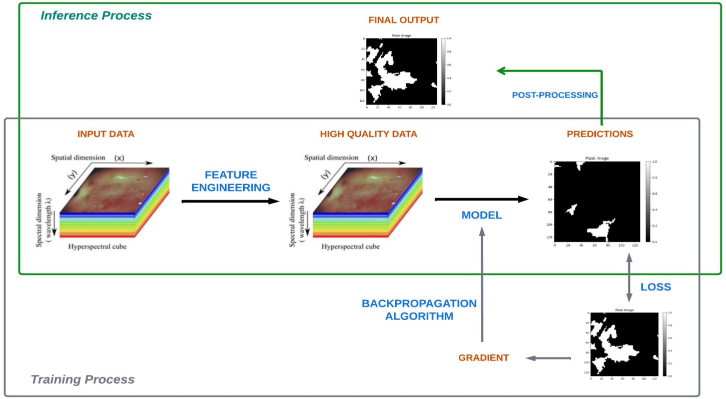


Table: Performance of baseline model

Model	Loss used	Input shape	Post process	Param	MIOU	F1	Time for 1 epoch
U-Net	CE	(B,128,128,14)	no	31.4 M	0.6001	0.6783	≈ 2m37s (gpu T4)

☐ Improve F1 score and Mean IOU ?
☐ Low parameters ?

AGENDA

I. Introduction

II. Loss Examination

III. Feature Engineering

IV. Model Architecture

V. Optimization Algorithm

VI. Post Processing

VII. Conclusion

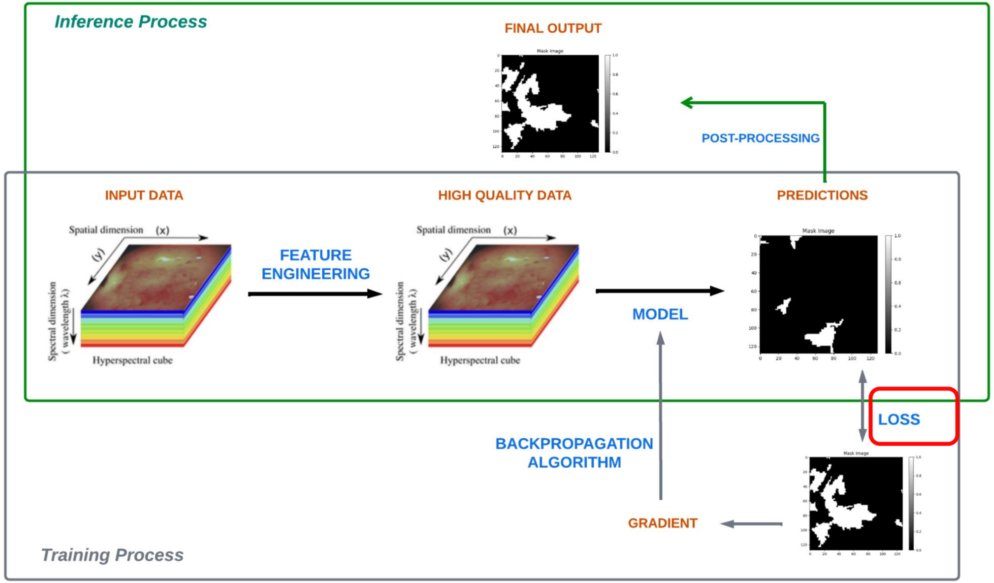


Figure: Common development process of a deep learning model

II. Losses Examination

- Test segmentation losses individually:
    - Distribution-based:
        - Cross Entropy (CE)
        - Focal loss
        - Log-Cosh loss
    - Region-based:
        - IOU loss:
        - Tversky loss:
        - Lovasz loss:
    - Others:
        - **Center loss:** gives extra weights on the pixels near the center of a landsliding region.
        - **Boundary loss**: gives extra weights on the pixels near the boundary of a landsliding region.

II. Losses Examination

- Data is already normalized (each band is divided by its mean value).
- Feature Engineering: None ⇒ input shape: (B, 128,128, 14)
- Data Augmentation: only rotation + cutmix
- Model: U-Net
- Backpropagation Algorithm: Adam with custom learning rate scheduler
- Post-processing: None

Note: all of the settings stay the same (U-Net, Adam optimizer, basic data augmentation, no post-processing, no feature engineering).

Table: Performance of the system with individual losses

Loss	Mean IOU	F1	1 epoch time
Cross Entropy	0.6001	0.6783	≈2m37s (gpu T4)
Focal	**0.6037**	**0.6828**	**≈2m40s (gpu T4)**
Log-Cosh	05995	0.6773	≈2m35s (gpu T4)
IOU	**0.6023**	**0.6820**	**≈2m40s (gpu T4)**
Tversky	0.5854	0.6621	≈2m40s (gpu T4)
Lovasz	0.5737	0.6461	≈2m45s (gpu T4)
Boundary	0.5492	0.6013	≈3m0s (gpu T4)
Center	0.5821	0.6561	≈3m0s (gpu T4)

II. Losses Examination

- Data is already normalized (each band is divided by its mean value).
- Feature Engineering: None ⇒ input shape: (B, 128,128, 14)
- Data Augmentation: only rotation + cutmix
- Model: U-Net
- Backpropagation Algorithm: Adam with custom learning rate scheduler
- Post-processing: None

Note: all of the settings stay the same (U-Net, Adam optimizer, basic data augmentation, no post-processing, no feature engineering).

⇒ Why not combine the best losses ?

⇒> Loss used for our system: α * Focal_loss + (1-α) * IOU_loss where α is a hyper-parameter in the range of [0,1]

Table: Performance of the system using combined loss.

Loss	mIOU	F1	1 epoch time
IOU	0.6023	0.6820	≈2m40s (gpu T4)
Focal	0.6037	0.6828	≈2m43s (gpu T4)
Focal-IOU	**0.6114**	**0.6905**	**≈2m55s (gpu T4)**
Entropy-IOU	0.6052	0.6861	≈2m51s (gpu T4)
Focal-IOU-Center	0.6045	0.6842	≈3m10s (gpu T4)

AGENDA

I. Introduction

II. Loss Examination

III. Feature Engineering

IV. Model Architecture

V. Optimization Algorithm

VI. Post Processing

VII. Conclusion

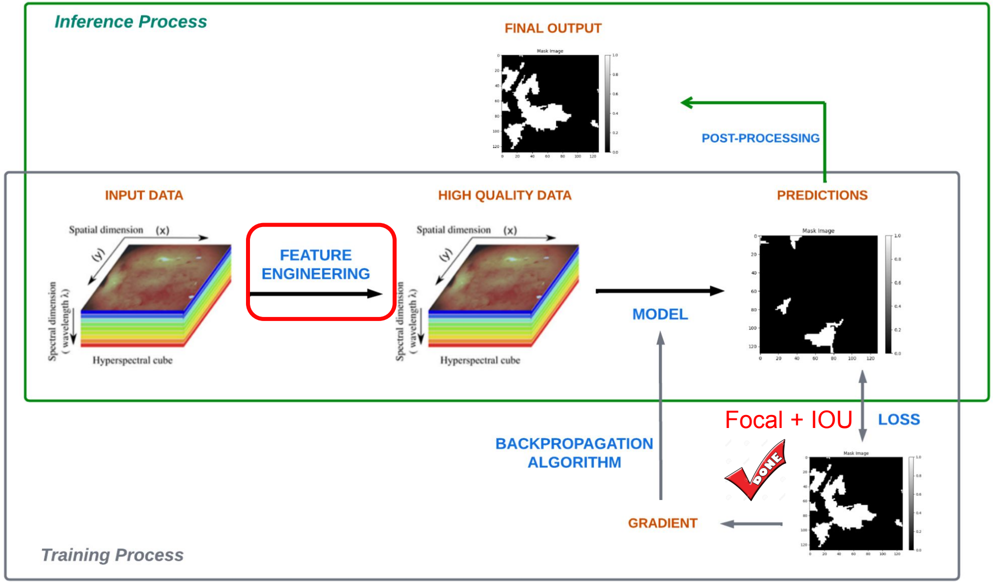


Figure: Common development process of a deep learning model

III. Feature Engineering

\# 14 original bands. (12 bands + slope data + digital elevation model)

- RGB normalization (+3 channels):
  - local normalization: (x - x_img_min) / (x_img_max - x_img_min)
  - global normalization : (x - x_dataset_min) / (x_dataset_max - x_dataset_min)

- Add Normalized Difference Vegetation Index (+1 channel):
  - NVDI = (NIR - RED) / (NIR + RED) = (B8 - B4) / (B8 + B4)

- Add Vegetation Index(+1 channel):
  - VI = (B8 - B11) / (B8 + B11)

- Add Normalized Burn Ratio (+1 channel):
  - NBR = (NIR-SWIR) / (NIR+SWIR) = (B8 - B12) / (B8 + B12)

- Add a grayscale image (+1 channel):
  - Gray = (R + B +G) / 3 = (B2 + B3 + B4) / 3

- Add 2 blurred image (+2 channels):
  - apply Gausian and Median filter (k=10)

- Add an image of edges(+1 channel):
  - apply Canny edge detector ⇒ an image with only edges

- Add images of gradient (+2 channels):
  - calculate image gradient across x and y axis (use Sobel kernel)

III. Band Selection

- 14 original bands.
- RGB local normalization (+3 channels):
- Add new features (+4 channels):
  - NVDI = (NIR - RED) / (NIR + RED)
  - VI = (B8 - B11) / (B8 + B11)
  - NBR = (NIR-SWIR) / (NIR+SWIR)
  - Gray = (R + B +G) / 3
- Add 2 blurred image (+2 channels):
  - apply Gausian and Median filter (k=10)
- Add an image of edges (+1 channel):
  - apply Canny edge detector
- Add images of gradient (+2 channels):
  - calculate image gradient across x and y axis

Note: all of the configuration stay the same (U-Net, Focal + IOU loss, Adam optimizer, basic data augmentation, no post-processing)

⇒ The input shape will be (B,128,128,23) instead of (B,128,128,14) in previous systems.

Number of input channels	mIOU	F1
9 (3 rgb + 6 norm rgb)	0.5671	0.6357
14 (original bands)	0.6114	0.6905
17 (14 og bands + 3 local rgb)	0.6122	0.6939
21 (17 + 4 new features)	0.6097	0.6983
23 (21 + 2 blur channels)	**0.6176**	**0.6996**
25 (23 + 2 gradient channels)	0.6164	0.6991
26 (25 + 1 edge image)	0.6065	0.6854
22 (17 + 2 blur + grad + 1 edge)	0.5761	0.6463

Table: Performance of the system with different channel combinations.

AGENDA

I. Introduction

II. Loss Examination

III. Feature Engineering

IV. Model Architecture

V. Optimization Algorithm

VI. Post Processing

VII. Conclusion

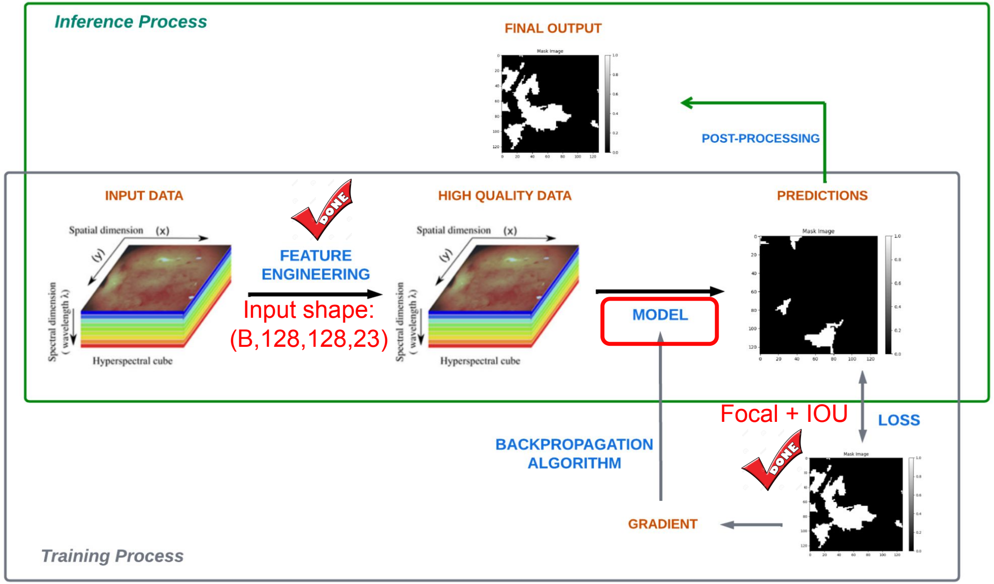


Figure: Common development process of a deep learning model

AGENDA

I. Introduction

II. Loss Examination

III. Feature Engineering

IV. Model Architecture

+ Model Head

+ Model Backbone

+ Improve The Backbone

. Architecture

. Low complexity model

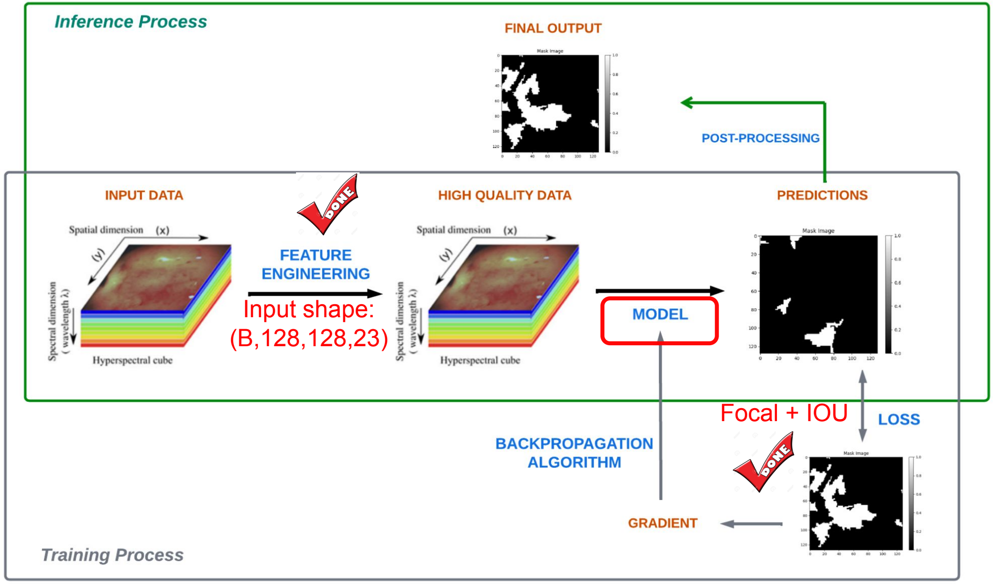
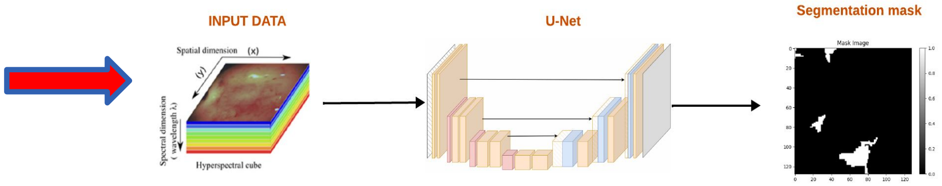


Figure: Common development process of a deep learning model

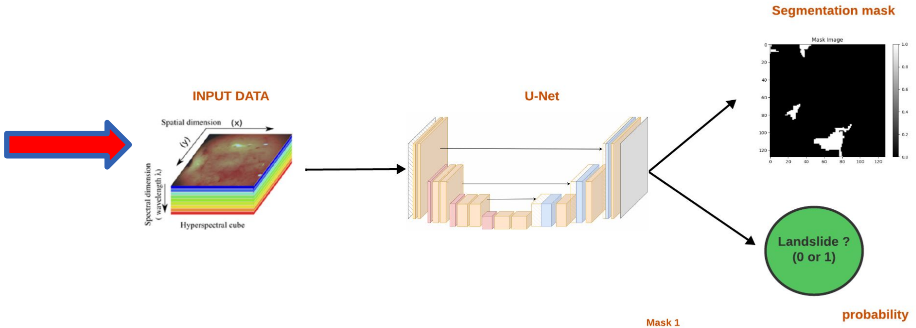
IV-A. Model's Head

- **Naive approach**: output **single segmentation mask** (normal mono branch U-Net).
  ⇒ calculate segmentation loss on a single mask



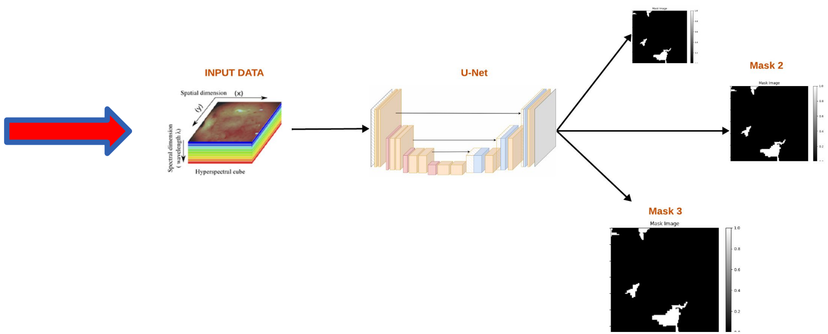
More complicated

- **Dual-head approach**: output **a segmentation mask** and **the probability of whether the image contains landsliding regions**.
  ⇒ calculate segmentation + classification loss



More complicated

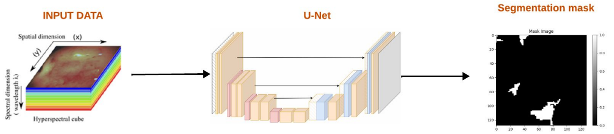
- **Multi-resolution approach** (our proposed method): output **several segmentation masks with different resolution** (ex: 128x128x2, 256x256x2, 64x64x2).
  ⇒ calculate segmentation loss based on multi-resolution masks of a same input image

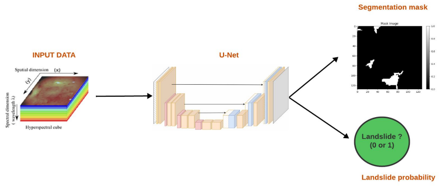


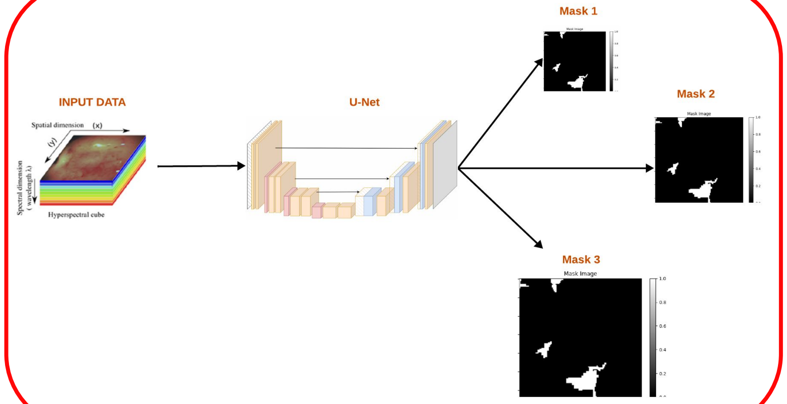
IV-A. Model’s Head

Table: Performance of the system using different model’s heads.

Architecture	Losses	mIOU	F1
Dual-head	focal + iou + cross entropy	0.5911	0.6651
Mono-branch	focal + iou	0.6176	0.6996
Multi-resolution	**focal + iou**	**0.6219**	**0.7045**







IV-B. Model's Backbone

- Drawbacks of traditional U-Net:
    - Has concatenations but still suffer vanish gradient problem because of no skip connections between several continuous convolutions.
    - Outdated (no attention mechanism, not use Convolution Transpose for upsampling, etc).
    - Not well-designed as modern architectures.
    - A large number of parameters.

⇒ Try different Backbone Architectures used for segmentation task like Deeplab, MobileNet or EfficientNet based.

Table: Performance of the system using well-known backbones

Architecture	mIOU	F1
U-Net	**0.6219**	**0.7045**
DeepLab v3 based	0.6049	0.6830
MobileNetV3 based	0.5472	0.6026
EfficientNetV2 based	0.5727	0.6451

 U-Net like architecture is still efficient for segmentation task

IV-B. Improve Model's Backbone

- To leverage U-Net like architecture and address the drawbacks of traditional U-Net, we try 2 ways:
  - (1) Add attention module to the traditional U-Net
  - (2) Replace layers at each level of traditional U-Net blocks by our blocks

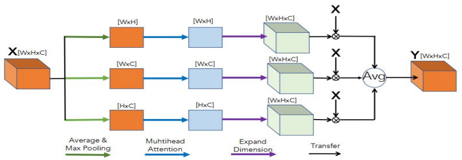


Figure: Our proposed attention block (in a previous paper)

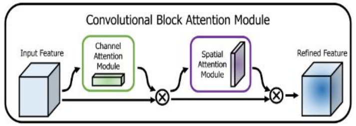


Figure: CBAM [12]

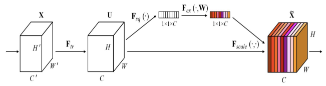


Figure: Squeeze and Exciting (SE) [13]

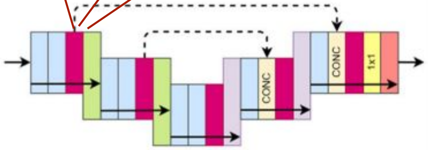


Figure: (1) Add attention module [9]

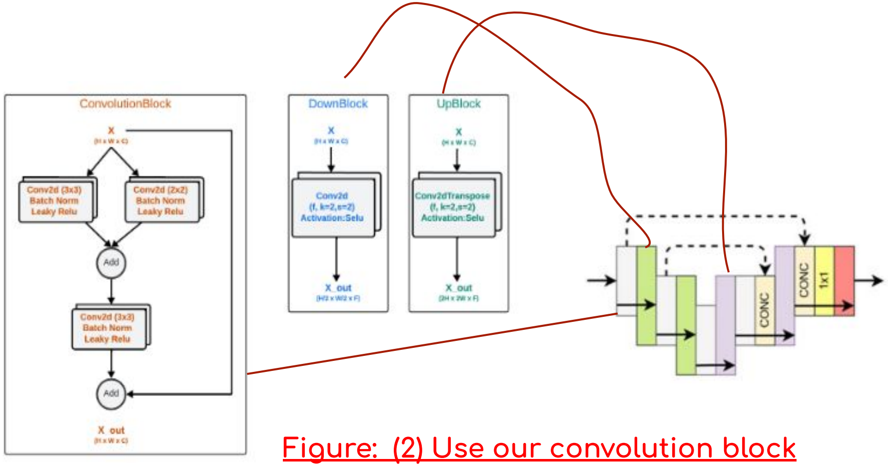


Figure: (2) Use our convolution block

IV-B. Improve Model's Backbone

Note: All of the other settings stay the same (input shape: (128,128,23), Focal + IOU loss, Adam optimizer, basic data augmentation, no post-processing).

Table: Performance of the system using improved U-Net

Architecture	Param	mIOU	F1	1 epoch time
Traditional U-Net	31.4M	0.6219	0.7045	≈3m05 (gpu Titan)
U-Net (add CBAM attention)	31.9M	6253	0.7082	≈3m38 (gpu Titan)
U-Net (add SE attention)	31.6M	0.6286	0.7126	≈3m44 (gpu Titan)
U-Net (use our proposed attention)	**31.8M**	**0.6305**	**0.7145**	**≈3m50 (gpu Titan)**
U-Net (use our convolution blocks)	**24.4M**	**0.6345**	**0.7207**	**≈3m15 (gpu titan)**

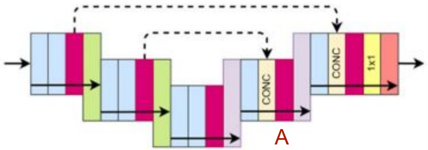


Figure: Add attention module

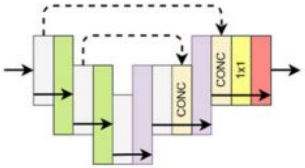


Figure: Use our ConvolutionBlock

IV-B. Model's Backbone

Note: All of the other settings stay the same (input shape: (128,128,23), Focal + IOU loss, Adam optimizer, basic data augmentation, no post-processing).

⇒ why not combine our convolution blocks vs attention module ?

Table: Performance of our combination: attention + conv block

Architecture	Param	mIOU	F1	1 epoch time
U-Net (use our proposed attention)	31.8M	0.6305	0.7145	≈3m50 (gpu Titan)
U-Net (use our convolution blocks)	24.4M	0.6345	0.7207	≈3m15 (gpu titan)
RAU-Net 1 (our combination 1)	**24.8M**	**0.6388**	**0.7242**	**≈4m40 (gpu Titan)**
RAU-Net 2 (our combination 2)	24.8M	0.6348	0.7211	≈5m (gpu Titan)

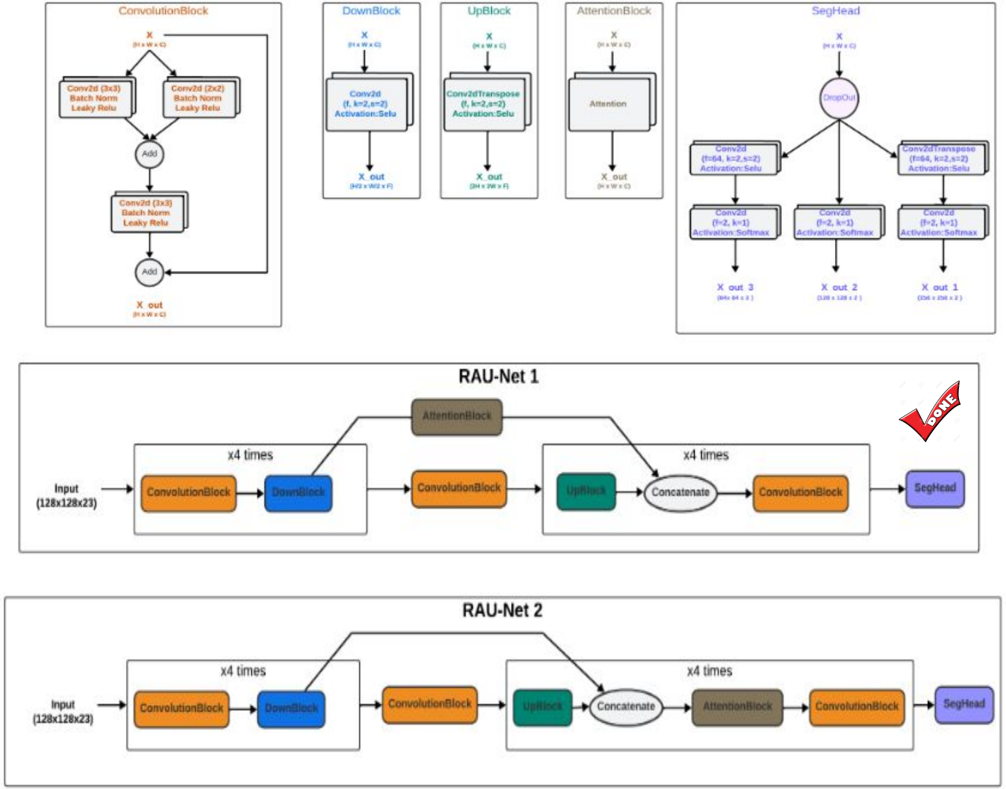


Figure: Two combinations of new Residual U-Net vs attention mechanism

IV-C. Small Models

- Our proposed RAU-Net1 still requires 24.8M parameters and need a lot of training/inference time using gpu Titan which is really hard to be deployed on edge device.

⇒ We would like to provide a small segmentation model which is around 10 times smaller and need less time to run even in a weaker device.

Note: All of the other settings stay the same (input shape: (128,128,23), Focal + IOU loss, Adam optimizer, basic data augmentation, no post-processing).

Table: Performance of different low-complexity models

Architecture	param	mIOU	F1	1 epoch time
V-Net	2.6M	0.5893	0.6662	≈4m (gpu 2080)
Squeeze U-Net	2.6M	0.5967	0.6744	≈2m40 (gpu 2080)
MobileNetV3 based	2.6M	0.5472	0.6026	≈2m32 (gpu 2080)
DeepLabv3 based	2.6M	0.5817	0.6524	≈2m47 (gpu 2080)
Residual U-Net based	2.6M	0.6043	0.6830	≈2m43 (gpu 2080)
Small RAU Net 1	**2.6M**	**0.6076**	**0.6877**	**≈2m42 (gpu 2080)**

AGENDA

I. Introduction

II. Loss Examination

III. Feature Engineering

IV. Model Architecture

V. Optimization Algorithm

VI. Post Processing

VII. Conclusion

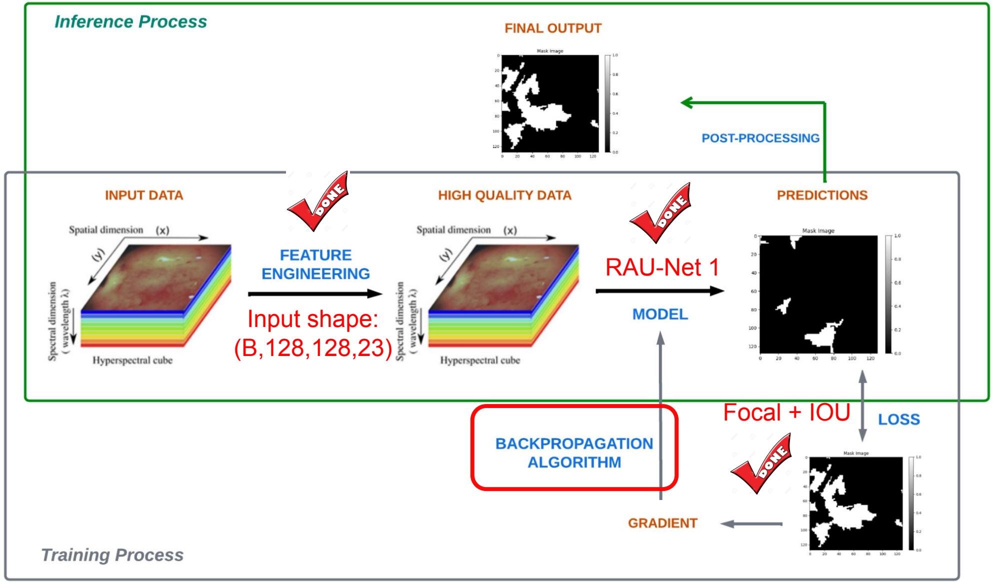


Figure: Common development process of a deep learning model

V.Backpropagation Optimizers

- Try out different optimization algorithms to examine the model's performance so that it will be easier to finetune in case of deploying into production.

Note: All of the other settings stay the same (input shape: (128,128,23), Focal + IOU loss, RAU-Net, basic data augmentation, no post-processing).

Optimizer	mIOU	F1	1 epoch time
SGD	0.5180	0.5622	≈4m30 (gpu Titan)
Adagrad	0.5283	0.5779	≈4m20 (gpu Titan)
RMSprop	0.6286	0.7125	≈4m37 (gpu Titan)
Adam	**0.6388**	**0.7242**	**≈4m40 (gpu Titan)**
AdamW	0.6340	0.7203	≈4m40 (gpu Titan)
Nadam	0.6119	0.6960	≈5m00 (gpu Titan)

Table: Performance of RAU-Net with different optimizers

AGENDA

I. Introduction

II. Loss Examination

III. Feature Engineering

IV. Model Architecture

V. Optimization Algorithm

VI. Post Processing

VII. Conclusion

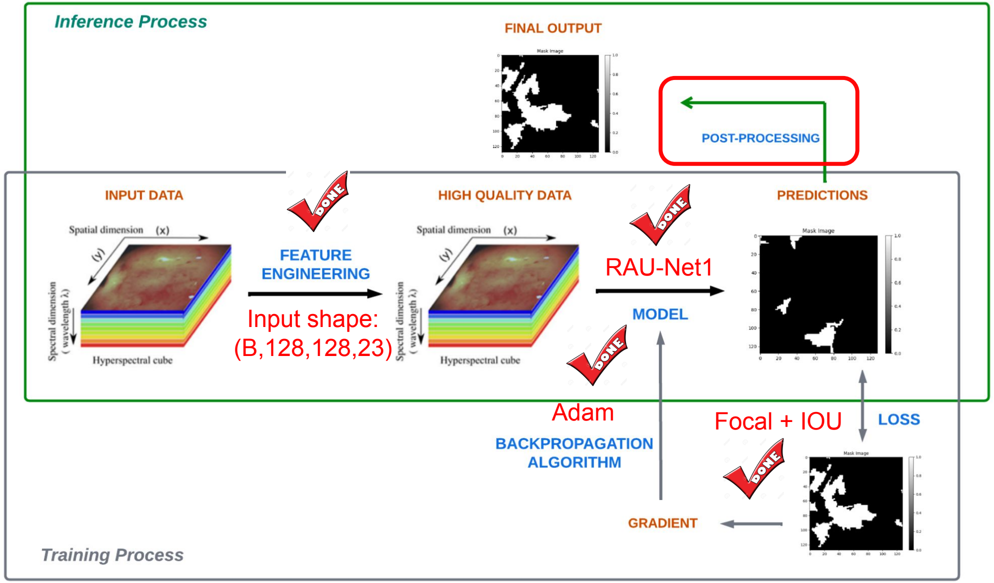


Figure: Common development process of a deep learning model

VI. Post-Processing Techniques

- List of post-processing techniques used
  - Morphology operation (traditional remove noise techniques used in computer vision).
  - Voting masks (use average of 3 resolution masks).
  - Multi-angles (make prediction of the same image with different angles than take average).
  - Thresholding (x > threshold ? 1 else 0)
  - Combine good techniques

\# Note: All of the other settings stay the same (input shape: (128,128,23), Focal + IOU loss, RAU-Net, basic data augmentation, Adam optimizer).

Techniques	mIOU	F1
None	0.6377	0.7220
Morphology opening (salt noise) closing (pepper noise)	 0.5120 0.5025	 0.5561 0.4959
Voting masks	0.6363	0.7215
Multi-angles	0.6381	0.7234
Thresholding 0.4 0.5 0.6 0.75 0.85 0.9 **0.95** 0.99	 0.6329 0.6388 0.6402 0.6443 0.6469 0.6507 **0.6597** 0.6468	 0.7178 0.7242 0.7260 0.7302 0.7313 0.7368 **0.7463** 0.7311
thresholding + multi-angles	0.6492	0.7344

Table: Performance of RAU-Net with different post-processing techniques

VI. Post-Processing (k-fold)

- Thresholding (x > threshold ? 1 else 0) on 5-fold cross validation.
- All of previous results were obtained from validation process of fold number 1.

\# Note: All of the other settings stay the same (input shape: (128,128,23), Focal + IOU loss, RAU-Net, basic data augmentation, Adam optimizer).

⇒

Thresholding ?	Mean F1	STD of F1
No	0.8408	0.0584
Yes	**0.8453**	**0.0497**

Table: Mean and standard deviation after apply 5-fold cross validation

Fold-Number	Threshold	mIOU	F1
1	t = 0.5 **t = 0.95**	0.6388 **0.6597**	0.7242 **0.7463**
2	t = 0.5 t = 0.5	0.7887 **0.7887**	0.8676 **0.8676**
3	t = 0.5 **t = 0.45**	0.7996 **0.7998**	0.8768 **0.8770**
4	t = 0.5 **t = 0.4**	0.7942 **0.7946**	0.8721 **0.8725**
5	t = 0.5 t **= 0.5**	0.7823 **0.7823**	0.8631 **0.8631**

Table: Performance of RAU-Net using k-fold cross validation

AGENDA

I. Introduction

II. Loss Examination

III. Feature Engineering

IV. Model Architecture

V. Optimization Algorithm

VI. Post Processing

VII. Conclusion

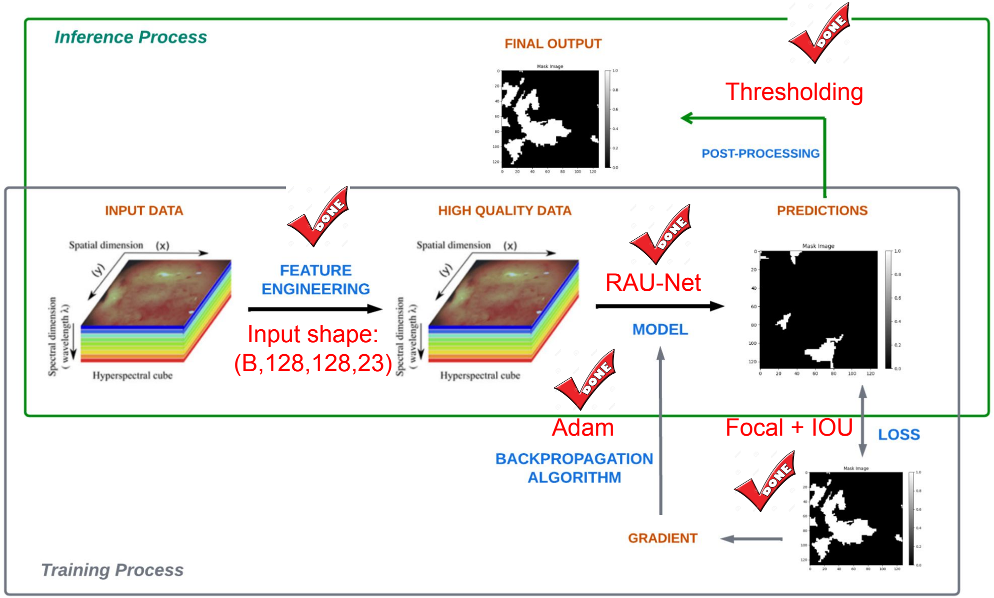


Figure: Common development process of a deep learning model

Compare with SOTA systems

- Our system's F1 score: 0.7463 (1st fold) and 0.8453 (avg of 5 folds)
  - split the development set into 80% for training and 20% of images for testing
  - apply 5-fold cross validation
- SOTA system's F1 score: 0.7365
  - 3799 images (whole development set) for training and 800 images (test set) for testing

 The challenge was done, we cannot send the results on the Test set for evaluation

Pos.	Team name	Submission	Score
1	Dense U-net	dense-u-net more details	0.73648522342449
2	Rahul Siripurapu	test-submission-4 more details	0.63255471755272
3	CDUT_D	batch1500 more details	0.63152339450232
4	CDUT_D	no_valid-2 more details	0.62633524991363
5	spppe1211	resnet50-deeplabv3_plus more details	0.62039377381962
6	virylon	test5 more details	0.61970553956848
7	CDUT_D	non-base more details	0.61933581988945
8	abc	sub5 more details	0.61518270757736
9	spppe1211	shufflenetv2-deeplabv3plus more details	0.6120815484854
10	spppe1211	resnet-deeplabv3_plus_lr more details	0.61117558022199

< 1 2 ... 6 >

VII-A. Small Demo

DEMO

https://huggingface.co/spaces/Cam-Le/landslide

VII-A. Small Demo

- Besides our small demo, we found out that we can automatically download remote sensing images of Sentinel-2 mission using API provided by https://apps.sentinel-hub.com/ (just run the code below to automatically download a remote sensing image of a specific region on a specific day) .

```python
sb_coords_wgs84 = (16.303089,48.180581,16.316371,48.186604) #  (longitude and latitude coordinates of lower left and upper right corners)
resolution = 10
sb_bbox = BBox(bbox=sb_coords_wgs84, crs=CRS.WGS84)
sb_size = (400,400)
```

```python
evalscript_all_bands = """
    //VERSION=3
    function setup() {
        return {
            input: [{
                bands: ["B01","B02","B03","B04","B05","B06","B07","B08","B8A","B09","B10","B11","B12"],
                units: "DN"
            }],
            output: {
                bands: 12,
                sampleType: "INT16"
            }
        };
    }

    function evaluatePixel(sample) {
        return [sample.B01,
                sample.B02,
                sample.B03,
                sample.B04,
                sample.B05,
                sample.B06,
                sample.B07,
                sample.B08,
                sample.B8A,
                sample.B09,
                sample.B10,
                sample.B11,
                sample.B12];
    }
"""

request_all_bands = SentinelHubRequest(
    evalscript=evalscript_all_bands,
    input_data=[
        SentinelHubRequest.input_data(
            data_collection=DataCollection.SENTINEL2_L1C,
            time_interval=("2023-09-21", "2023-09-27"),
            mosaicking_order=MosaickingOrder.LEAST_CC,
        )
    ],
    responses=[SentinelHubRequest.output_response("default", MimeType.TIFF)],
    bbox=sb_bbox,
    size=sb_size,
    config=config,
)

all_bands_response = request_all_bands.get_data()
```

DEMO

https://huggingface.co/spaces/Cam-Le/landslide

VII-B. Conclusion

- In this research, we built segmentation models from scratch and did intensive experiments on LandSlide4Sense dataset to develop a high performance landslide segmentation model and its small version.
- To achieve the best F1 score of 0.7463 (1st fold) and 0.8453 (avg of 5 folds), we use following techniques:
  - Feature engineering: adding 4 new features, add blurred/grayscale images and add 3 channels for rgb local normalization.
  - Losses: combination of Focal loss and IOU loss
  - Model: proposed RAU-Net 1 (24.8M parameters) with a multi-resolution prediction head
  - Optimizer: Adam
  - In the inference/test process we apply thresholding to further boost the model's performance.
- We also provide a small landslide segmentation model (2.6M parameters) which has a F1 score of 0.6877 (before applying thresholding) and 0.7045 (after applying thresholding) on the 1st fold and is potential to be deployed on edge devices.
- Future work:
  - Continue performing EDA on tabular data related landslide regions in the gAia project.
  - We may develop and combine ML approach and DL approach together (multi-models) or use a model to double check with the other.
  - Deploy on server with an accessible API (if this direction can go further).

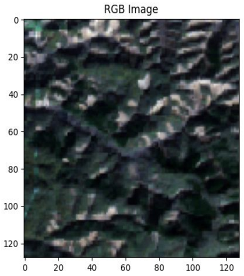


Figure: A remote sensing image with landslide

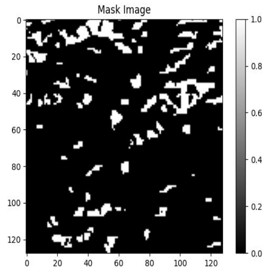


Figure: Ground truth mask

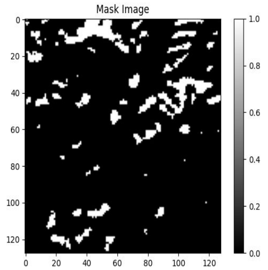


Figure: Predicted mask from our model (1)

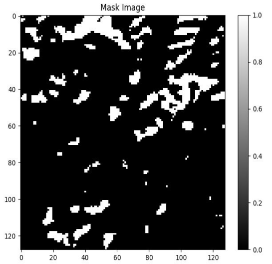


Figure: Predicted mask from our model (2)

AUSTRIAN INSTITUTE OF TECHNOLOGY

Thank You For Your Attention

Vienna, October 2023

Support Slide: Cutmix

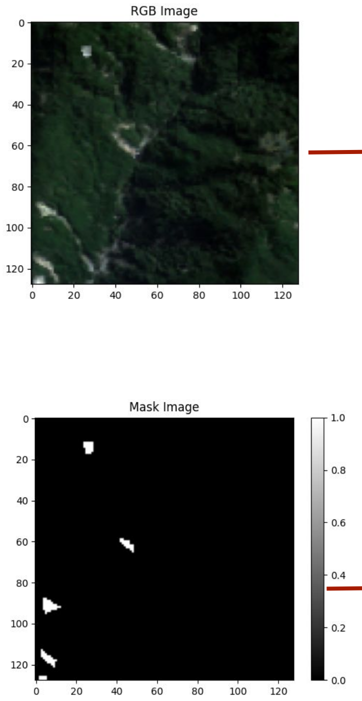


Figure: Original image/mask

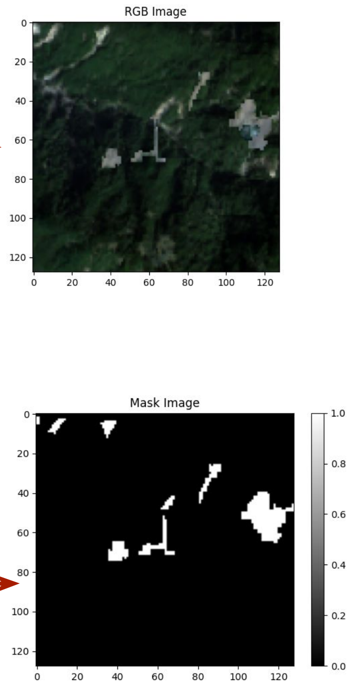


Figure: After cutmix + rotate image/mask

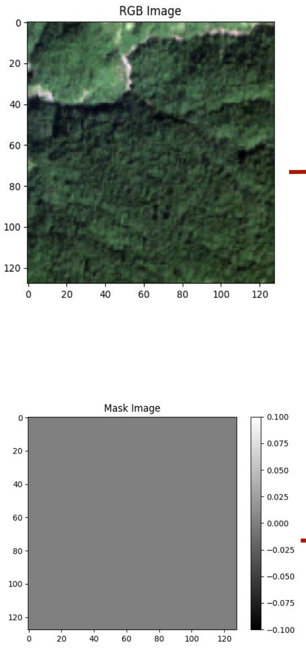


Figure: Original image/mask

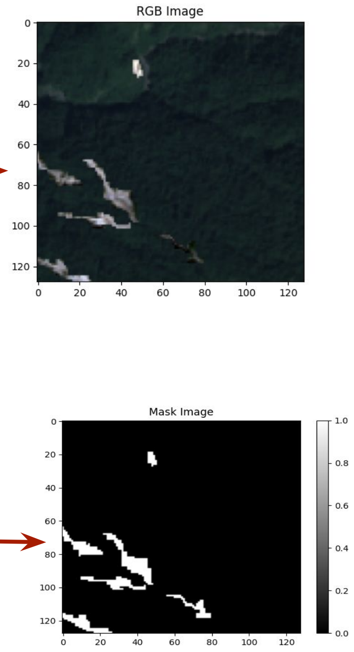


Figure: After cutmix image/mask

Support Slide: Sentinel only

Sentinel-only			
Fold number	input shape	mIOU	F1
k=1	(128,128,21)	60.03	68.02
k=2	(128,128,21)	75.91	84.33
k=3	(128,128,21)	77.08	85.44
k=4	(128,128,21)	76.61	84.98
k=5	(128,128,21)	74.93	83.62

With POLSAR			
Fold number	input shape	mIOU	F1
k=1	(128,128,23)	63.88 65.97	72.42 74.63
k=2	(128,128,23)	78.87 78.87	86.76 86.76
k=3	(128,128,23)	79.96 79.98	87.68 87.70
k=4	(128,128,23)	79.42 79.46	87.21 87.25
k=5	(128,128,23)	78.23 78.23	86.31 86.31

Support Slide: Losses

$$-\log(P_t)$$

Cross Entropy loss

$$\mathrm{TI} = \frac{\mathrm{TP}}{\mathrm{TP} + \alpha\mathrm{FN} + \beta\mathrm{FP}}$$

Tversky loss (log (1 - TI))

$$\log(\cosh(y_i^p - y_i))$$

Log cosh loss

$$\frac{|B \cap B^{gt}|}{|B \cup B^{gt}|}$$

IOU loss (log(1- iou))

$$-\alpha_t(1 - p_t)^\gamma \log(p_t)$$

Focal loss

References

1. What is Remote Sensing? The Definitive Guide - GIS Geography (last access: 10/9/2023)

2. Maik Netzband, William L Stefanov, and Charles Redman. Applied remote sensing for urban planning, governance and sustainability. Springer Science & Business Media, 2007

3. Rajesh Bahadur Thapa and Yuji Murayama. “Urban mapping, accuracy, & image classification: A comparison of multiple approaches in Tsukuba City, Japan”. In: Applied geography 29.1 (2009), pp.135–144.

4. Dimitris Poursanidis and Nektarios Chrysoulakis. “Remote Sensing, natural hazards and the contribution of ESA Sentinels missions”. In: Remote Sensing Applications: Society and Environment 6 (2017), pp.25–38.

5.Maryam Mehmood et al. “Remote sensing image classification: A comprehensive review and applications”. In: Mathematical Problems in Engineering 2022

6. https://www.photonics.com/Articles/Multispectral_Lighting_A_Practical_Option_for/a66251 (last access: 10/9/2023)

7. https://www.iarai.ac.at/landslide4sense/challenge/ (last access: 10/9/2023)

8. https://sentinel.esa.int/web/sentinel/missions/sentinel-2 (last access: 10/9/2023)

9. Kugelman, J., Allman, J., Read, S.A. *et al.* A comparison of deep learning U-Net architectures for posterior segment OCT retinal layer segmentation. *Sci Rep* 12, 14888 (2022). https://doi.org/10.1038/s41598-022-18646-2

10. Liang-Chieh Chen et al. Rethinking Atrous Convolution for Semantic Image Segmentation. 2017

11. Howard, Andrew, et al. "Searching for mobilenetv3." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.

12. Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

13. Woo, Sanghyun, et al. "Cbam: Convolutional block attention module." *Proceedings of the European conference on computer vision (ECCV)*. 2018.

References

14. Cheng Peng et al. “RSBNet: One-shot neural architecture search for a backbone network in remote sensing image recognition”. In: Neurocomputing (2023)

15. [Machine learning best practices: combining lots of models - The SAS Data Science Blog] (last access: 19/8/2023)

16. Zhichuang Sun et al. “Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps”. In: 30th USENIX Security Symposium (USENIX Security 21). 2021, pp. 1955–1972

17. [Supervised Machine learning - Javatpoint] (last access: 20/8/2023)

18. Qi Zhao et al. “Embedded Self-Distillation in Compact Multibranch Ensemble Network for Remote Sensing Scene Classification”. In: IEEE Transactions on Geoscience and Remote Sensing 60 (2022), pp. 1–15. doi: 10.1109/tgrs.2021.3126770. url: https://doi.org/10.1109%2Ftgrs.2021.3126770

19. Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 7132–7141.

20. Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: International conference on machine learning. PMLR. 2020, pp. 1597–1607

21. Trisla Chaurasia. Network Pruning Comprehensively. 2021. url: https://chaurasiat.medium.com/network-pruning-comprehensively-78d4274087d3.

22. [TensorFlow Lite Delegates]

23. [How to accelerate and compress neural networks with quantization - Mathematics of machine learning (tivadardanka.com)] (last access: 2/9/2023)

24. Nguyen Thanh Tuan. Deep Learning co ban. 2021. url: https://nttuan8.com/. (last access: 1/9/2023)

25. Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki, and Ramakrishna Nemani. 2015. Deepsat: a learning framework for satellite imagery. In Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems. 1–10

26. Cam Le et al. “A Robust and Low Complexity Deep Learning Model for Remote Sensing Image Classification”. In: 2023 8th International Conference on Intelligent Information Technology (ICIIT 2023). 2023.