Adaptive-Labeling for Enhancing Remote Sensing Cloud Understanding

[1]Gala Jay, [2]Nag Sauradip, [3]Huang Huichou, [4]Liu Ruirui, and [2]Zhu Xiatian

[1]NMIMS University, [2]University of Surrey, [3]Brunel University London, [4]City University of Hong Kong

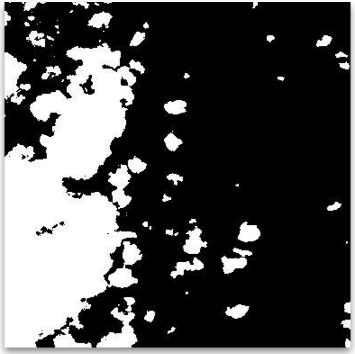
Presented at the TCCML Workshop at NeurIPS 2023

Problem Statement

Given a satellite image *I*, cloud detection predicts a pixel-level binary classification of whether each pixel is a cloud, outputting a binary mask *M*

Formally, a training dataset D_{train} contains image-mask pairs *(I, M)* used to train a cloud detection model, which is then evaluated on a testing split D_{test}

Image



Mask

Noise in Cloud Annotations

- Cloud annotation is a challenging task
- Previous methods assume the availability of reliable segmentation annotations without considering the noise in the dataset

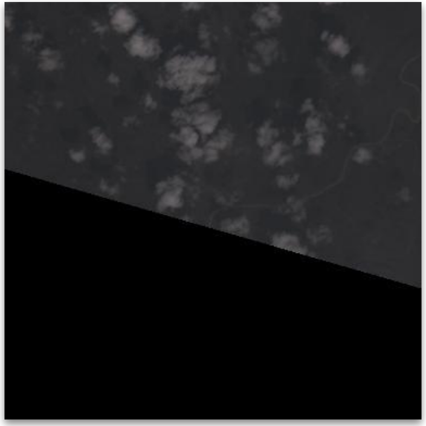


Image 1

Mask 1

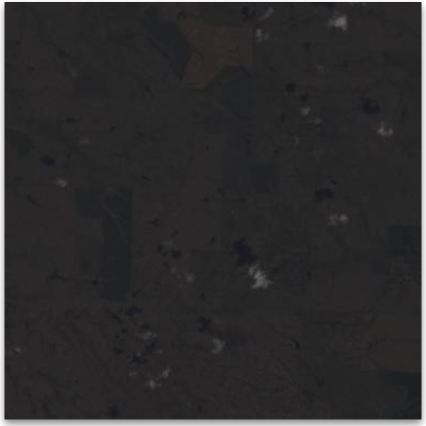
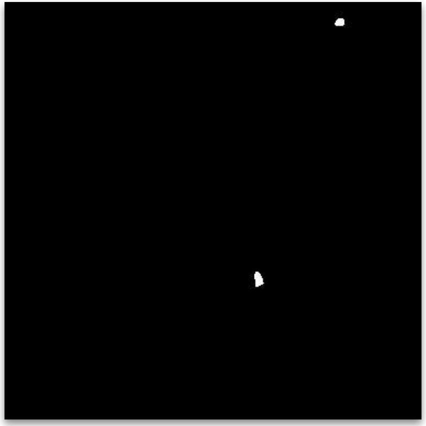


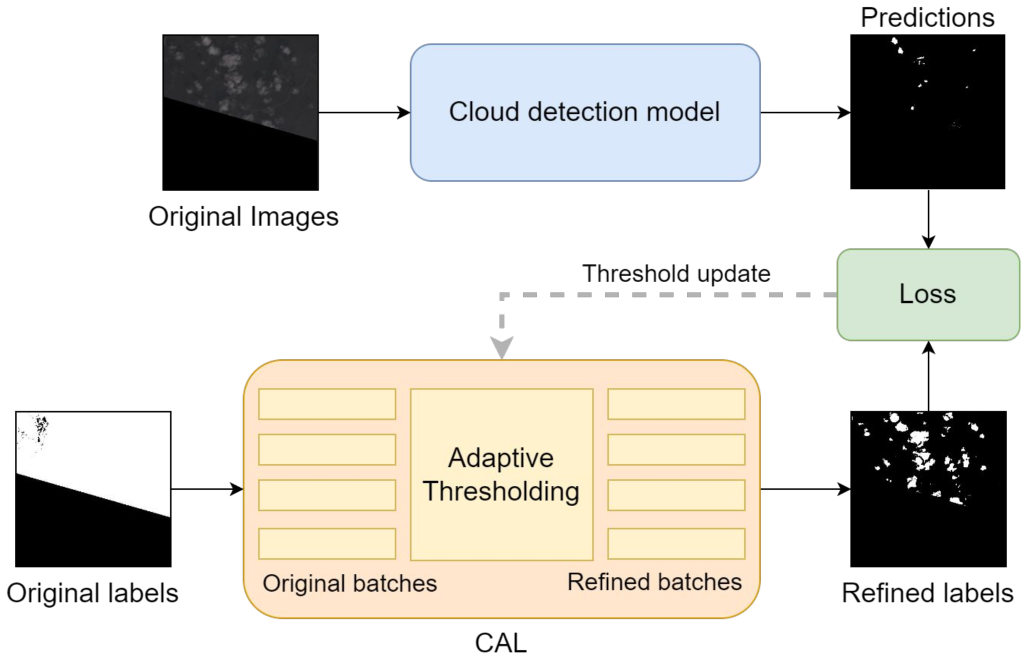
Image 2



Mask 2

CAL: Cloud Adaptive Labeling

- To tackle this noise, we propose **Cloud Adaptive-Labeling (CAL).**
- An efficient module that enhances existing cloud detection approaches.
- Plugged in as a post-processing block to **relabel** existing noisy cloud labels into a refined version.



CAL Algorithm

- **CAL** generates refined mask labels by using the **binarization** operation.
- We incorporate a trainable **pixel intensity threshold** to adaptively label the cloud masks
- New labels are used for fine-tuning the model.
- The threshold is dynamically adjusted based on the training loss

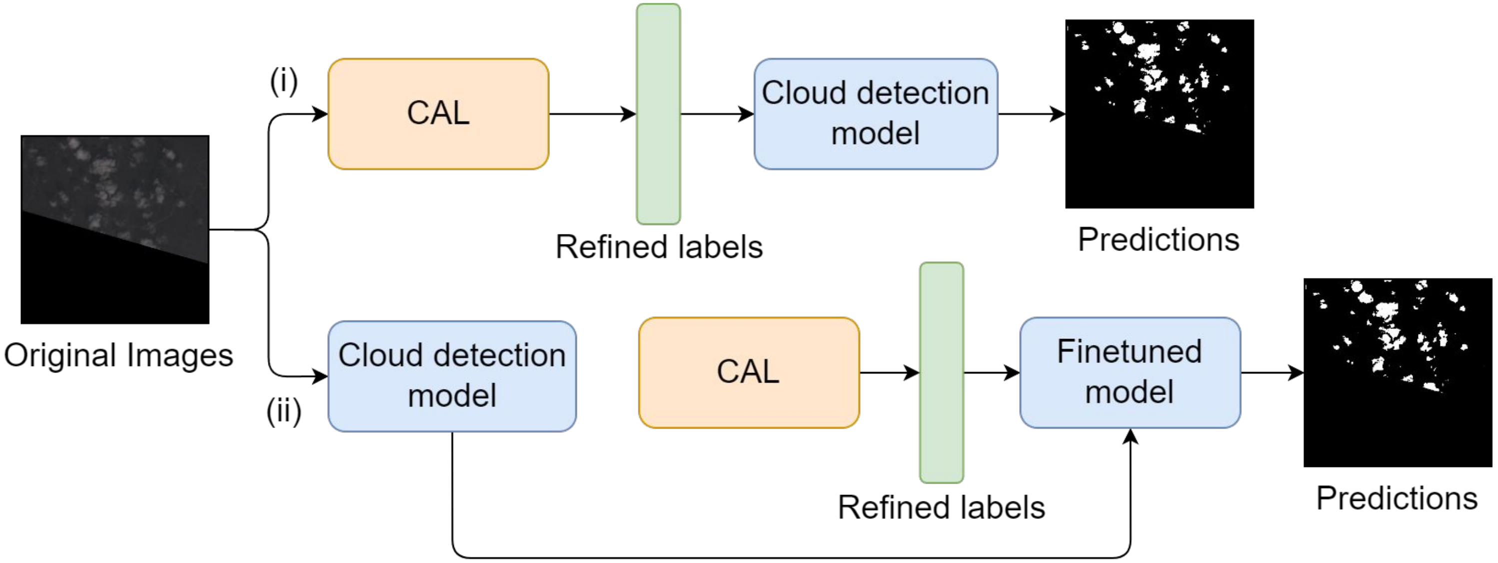
Algorithm 1: Cloud Adaptive Labeling

```
learnable_threshold = 60.0                                  // initial threshold
delta_x = 2.0                                                     // step size
best_loss = float('inf')
lower_bound = 45.0
update_frequency = 150

while not done do
    new_labels = binarize_image(images, threshold=learnable_threshold)
    outputs = model(images)
    loss = criterion(outputs, new_labels)
    best_loss = min(best_loss, loss)
    if (idx + 1) % update_frequency == 0 then
        if loss > best_loss then
            learnable_threshold -= delta_x              // Decrease the threshold
            learnable_threshold = max(lower_bound, learnable_threshold)
        else
            learnable_threshold += delta_x              // Increase the threshold
```

Integration with existing models

- **CAL** can be integrated with existing models without **design changes** or **learnable params**
- CAL can be applied directly to any existing model **without retraining.**
- However, retraining with CAL has demonstrated enhanced results.



Advantages of CAL

- Enhances performance
- Eliminates the need for prior estimation of an appropriate threshold
- Model-agnostic. Plug-and-play
- Faster convergence
- Stabilizes performance metrics

Results

Table 1: Performance comparison on the 38-Cloud dataset.

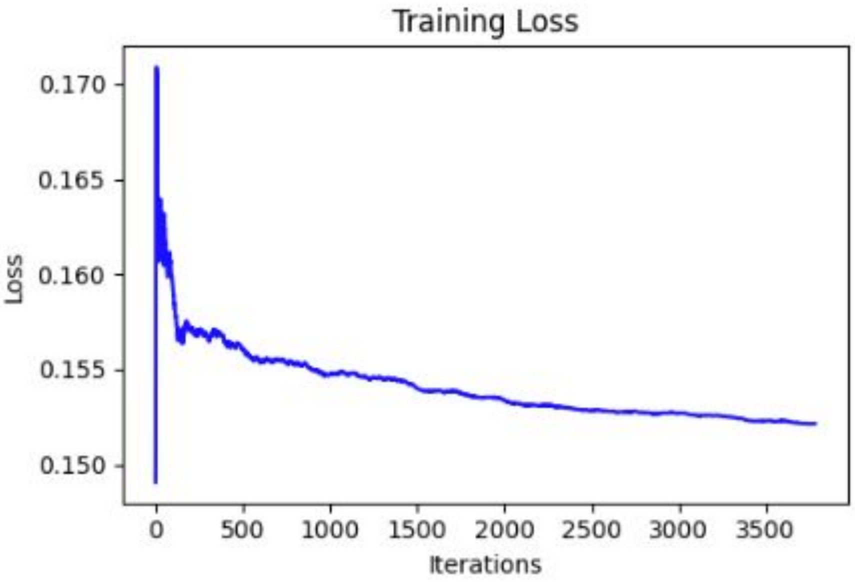
Method	mIoU	Precision	Recall	F1-Score	OA
CD-CTFM [5]	84.13	91.09	89.22	90.15	95.45
CD-AttDLV3+ [17]	81.24	88.85	87.58	88.21	94.49
CloudAttU [25]	84.73	90.62	89.95	90.28	95.92
CloudFCN [18]	83.31	88.81	89.61	89.21	95.66
CD-Net [16]	89.70	94.30	94.70	94.50	95.40
LWCDnet [4]	89.90	95.10	94.30	94.70	95.30
FCN [24]	60.40	80.69	63.02	69.07	92.38
DeeplabV3 [23]	44.02	86.25	89.53	86.73	96.07
U-Net [22]	73.69	74.31	97.17	82.10	91.20
FCN + **CAL**	83.94	91.52	87.73	89.26	96.74
DeeplabV3 + **CAL**	83.95	85.95	94.17	89.38	96.77
U-Net + **CAL**	**93.15**	**96.37**	**96.67**	**96.44**	**98.61**

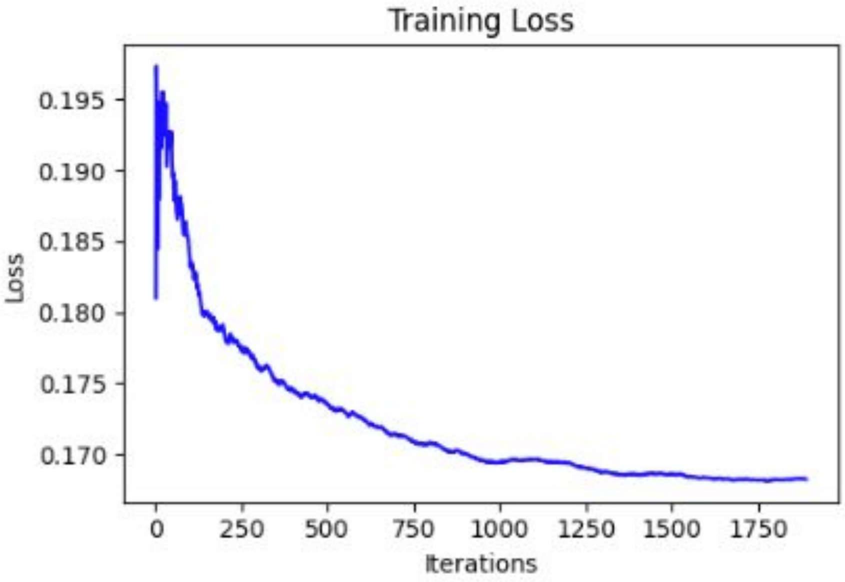
Results

Table 2: Performance comparisons of models with and without CAL.

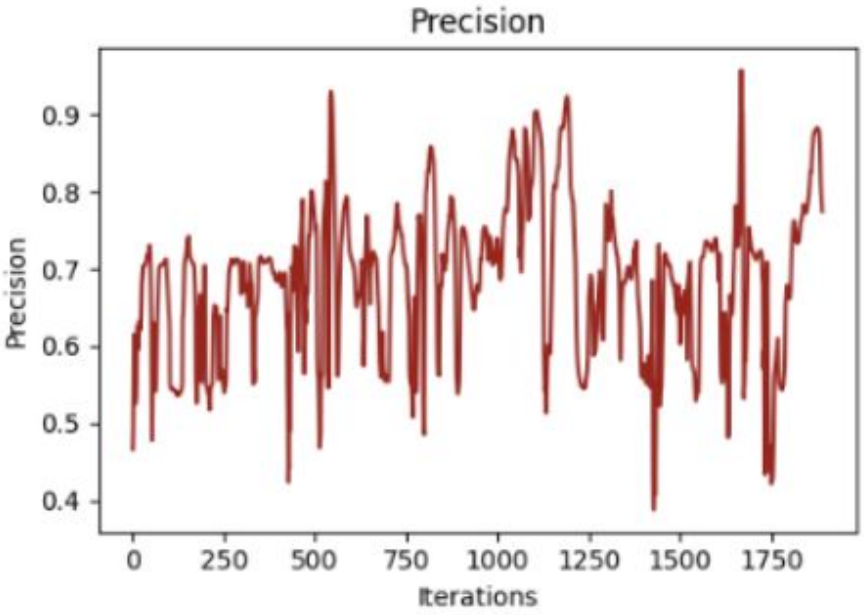
Methods	mIoU	Precision	Recall	F1-Score	OA
FCN	60.40	80.69	63.02	69.07	92.38
FCN + CAL (ours)	83.94	91.52	87.73	89.26	96.74
U-Net	73.69	74.31	97.17	82.10	91.20
U-Net + CAL (ours)	93.15	96.37	96.67	96.44	98.61
DeeplabV3	44.02	86.25	89.53	86.72	96.07
DeeplabV3 + CAL (ours)	83.95	85.95	94.18	89.39	96.77

Results

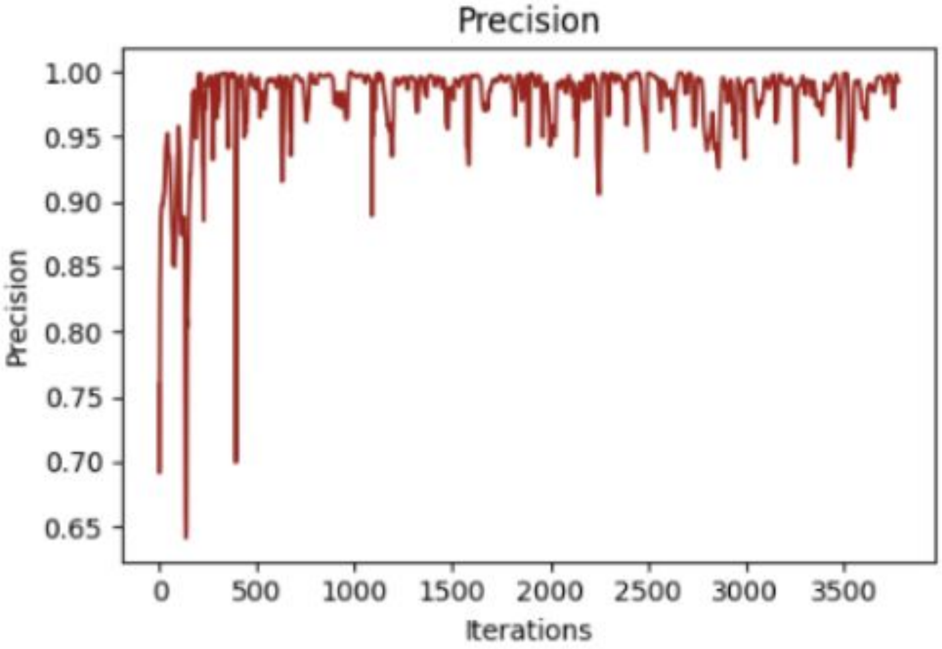

Training Loss
0.170
0.165
0.160
0.155
0.150
Loss
0
500
1000
1500
2000
2500
3000
3500
Iterations


Training Loss
0.195
0.190
0.185
0.180
0.175
0.170
Loss
0
250
500
750
1000
1250
1500
1750
Iterations

Results



Precision
0.9
0.8
0.7
0.6
0.5
0.4
Precision
0
250
500
750
1000
1250
1500
1750
Iterations



Precision
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
Precision
0
500
1000
1500
2000
2500
3000
3500
Iterations

Thank you

Contact jay.gala78@nmims.edu.in for any queries.

Code available at